VOT 74084

DESIGN AND DEVELOPMENT OF AN INTELLIGENT SECURITY LAYER FOR WEB-BASED APPLICATIONS

(REKABENTUK DAN PEMBANGUNAN PARAS KESELAMATAN PINTAR UNTUK APLIKASI BERASASKAN WEB)

ABDUL HANAN ABDULLAH
MOHD AIZAINI MAAROF
MOHD YAZID IDRIS
ABDUL SAMAD HAJI ISMAIL
CAHYO CRYSDIAN

FAKULTI SAINS KOMPUTER DAN SISTEM MAKLUMAT
UNIVERSITI TEKNOLOGI MALAYSIA

2005

ABSTRACT

Methods to activate firewall mechanism have been introduced in this research. The purpose is to build stronger protection for the intranet from the threats of Internet. The foundation of the work is the threat reduction strategies that are derived from formalizing and identifying the interaction between internal users and external parties. Internet access model is developed to facilitate this task. Mechanism of active firewall are divided into two main process i.e. initialisation and runtime process. The former process deals with the mechanism to start up and bring the active firewall into a point of its operation. Three approaches are introduced, namely open condition, close condition and lattice-based method. The open condition and close condition set the firewall into its extreme condition i.e. to open all available communication line or to close all connection respectively, while the lattice-based method affords to bring firewall into its optimum level to protect the intranet by establishing Internet connection based on the predetermined security level. In the runtime process, three methods are introduced as well i.e. adaptively updating security policy using fuzzy reasoning, detecting suspicious process using distributed agent-based module, and zero-based approach to have minimal network services at runtime. Besides analysing each method using its own parameters such as processing time, accuracy and speed for organizing canals, global evaluations were also held to investigate the protection can be delivered to the intranet. In this evaluation, security analysis and comparative study is held, in which each initialisation and runtime process are combined and analysed using three parameters that are created based on RFC 2979 i.e. probability of available network services, probability of exposed line, and denial of services. Results of this study deliver the combination of lattice-based and agent-based module become the best method for activating firewall.

ABSTRAK

Dalam penyelidikan ini, kaedah untuk mengaktifkan mekanisma dinding api telah diperkenalkan untuk membina perlindungan yang lebih kukuh daripada ancaman Internet. Asas kepada kajian ini ialah strategi pengurangan ancaman yang diambil daripada penformalan dan pengenalpastian hubungan antara pengguna dalaman dan luaran. Model capaian Internet dibangunkan untuk memudahkan tugas ini. Mekanisma bagi dinding api aktif dibahagikan kepada dua proses utama iaitu proses permulaan dan masa larian. Proses permulaan bertindak menguruskan mekanisma untuk memula dan mengaktifkan dinding api bagi membolehkan ia beroperasi. Tiga pendekatan telah diperkenalkan iaitu keadaan terbuka, keadaan tertutup dan kaedah berasaskan kekisi. Dalam keadaan terbuka dan keadaan tertutup, dinding api ditetapkan kepada keadaan ekstrimnya iaitu untuk membuka semua talian komunikasi yang ada atau menutup semua talian tersebut. Sementara itu, kaedah berasaskan kekisi berperanan untuk menetapkan dinding api kepada tahap optimumnya untuk melindungi intranet dengan membina hubungan Internet berdasarkan peringkat keselamatan yang telah ditetapkan. Dalam proses masa larian pula, tiga kaedah diperkenalkan iaitu penyesuaian pengemaskinian polisi keselamatan menggunakan *fuzzy reasoning*, mengenalpasti proses yang meragukan menggunakan modul berasaskan agen teragih dan pendekatan berasaskan sifar untuk memperolehi perkhidmatan rangkaian yang minimum pada masa larian. Selain menganalisa setiap kaedah yang dicadangkan menggunakan parameter-parameternya seperti masa pemprosesan, ketepatan dan kelajuan untuk menguruskan saluran, kajian keseluruhan juga dijalankan untuk menyelidik keselamatan kepada intranet. Dalam penilaian ini, setiap proses permulaan dan masa larian adalah digabungkan dan dianalisa menggunakan tiga parameter yang dibina daripada RFC 2979, iaitu kebarangkalian perkhidmatan rangkain yang ada, kebarangkalian talian yang terdedah dan penafian perkhidmatan. Hasil daripada kajian ini menunjukkan gabungan antara modul berasaskan kekisi dan modul berasaskan agen menjadi kaedah terbaik untuk mengaktifkan dinding api.

ACKNOWLEDGEMENTS

We would like to thank and express deepest gratitude and appreciation to MOSTI secretariat for providing a research grant for this wonderful project. Our appreciations are also addressed to Research Management Center, Universiti Teknologi Malaysia for organising and supporting the official works and documentation of this project, and also to the Faculty of Computer Science and Information System, Universiti Teknologi Malaysia for hosting the research activities by providing a comfortable laboratory.

VOT 74084

DESIGN AND DEVELOPMENT OF AN INTELLIGENT SECURITY LAYER FOR WEB-BASED APPLICATIONS

(REKABENTUK DAN PEMBANGUNAN PARAS KESELAMATAN PINTAR UNTUK APLIKASI BERASASKAN WEB)

ABDUL HANAN ABDULLAH
MOHD AIZAINI MAAROF
MOHD YAZID IDRIS
ABDUL SAMAD HAJI ISMAIL
CAHYO CRYSDIAN

FAKULTI SAINS KOMPUTER DAN SISTEM MAKLUMAT
UNIVERSITI TEKNOLOGI MALAYSIA

2005

ABSTRACT

Methods to activate firewall mechanism have been introduced in this research. The purpose is to build stronger protection for the intranet from the threats of Internet. The foundation of the work is the threat reduction strategies that are derived from formalizing and identifying the interaction between internal users and external parties. Internet access model is developed to facilitate this task. Mechanism of active firewall are divided into two main process i.e. initialisation and runtime process. The former process deals with the mechanism to start up and bring the active firewall into a point of its operation. Three approaches are introduced, namely open condition, close condition and lattice-based method. The open condition and close condition set the firewall into its extreme condition i.e. to open all available communication line or to close all connection respectively, while the lattice-based method affords to bring firewall into its optimum level to protect the intranet by establishing Internet connection based on the predetermined security level. In the runtime process, three methods are introduced as well i.e. adaptively updating security policy using fuzzy reasoning, detecting suspicious process using distributed agent-based module, and zero-based approach to have minimal network services at runtime. Besides analysing each method using its own parameters such as processing time, accuracy and speed for organizing canals, global evaluations were also held to investigate the protection can be delivered to the intranet. In this evaluation, security analysis and comparative study is held, in which each initialisation and runtime process are combined and analysed using three parameters that are created based on RFC 2979 i.e. probability of available network services, probability of exposed line, and denial of services. Results of this study deliver the combination of lattice-based and agent-based module become the best method for activating firewall.

ABSTRAK

Dalam penyelidikan ini, kaedah untuk mengaktifkan mekanisma dinding api telah diperkenalkan untuk membina perlindungan yang lebih kukuh daripada ancaman Internet. Asas kepada kajian ini ialah strategi pengurangan ancaman yang diambil daripada penformalan dan pengenalpastian hubungan antara pengguna dalaman dan luaran. Model capaian Internet dibangunkan untuk memudahkan tugas ini. Mekanisma bagi dinding api aktif dibahagikan kepada dua proses utama iaitu proses permulaan dan masa larian. Proses permulaan bertindak menguruskan mekanisma untuk memula dan mengaktifkan dinding api bagi membolehkan ia beroperasi. Tiga pendekatan telah diperkenalkan iaitu keadaan terbuka, keadaan tertutup dan kaedah berasaskan kekisi. Dalam keadaan terbuka dan keadaan tertutup, dinding api ditetapkan kepada keadaan ekstrimnya iaitu untuk membuka semua talian komunikasi yang ada atau menutup semua talian tersebut. Sementara itu, kaedah berasaskan kekisi berperanan untuk menetapkan dinding api kepada tahap optimumnya untuk melindungi intranet dengan membina hubungan Internet berdasarkan peringkat keselamatan yang telah ditetapkan. Dalam proses masa larian pula, tiga kaedah diperkenalkan iaitu penyesuaian pengemaskinian polisi keselamatan menggunakan *fuzzy reasoning*, mengenalpasti proses yang meragukan menggunakan modul berasaskan agen teragih dan pendekatan berasaskan sifar untuk memperolehi perkhidmatan rangkaian yang minimum pada masa larian. Selain menganalisa setiap kaedah yang dicadangkan menggunakan parameter-parameternya seperti masa pemprosesan, ketepatan dan kelajuan untuk menguruskan saluran, kajian keseluruhan juga dijalankan untuk menyelidik keselamatan kepada intranet. Dalam penilaian ini, setiap proses permulaan dan masa larian adalah digabungkan dan dianalisa menggunakan tiga parameter yang dibina daripada RFC 2979, iaitu kebarangkalian perkhidmatan rangkain yang ada, kebarangkalian talian yang terdedah dan penafian perkhidmatan. Hasil daripada kajian ini menunjukkan gabungan antara modul berasaskan kekisi dan modul berasaskan agen menjadi kaedah terbaik untuk mengaktifkan dinding api.

ACKNOWLEDGEMENTS

We would like to thank and express deepest gratitude and appreciation to MOSTI secretariat for providing a research grant for this wonderful project. Our appreciations are also addressed to Research Management Center, Universiti Teknologi Malaysia for organising and supporting the official works and documentation of this project, and also to the Faculty of Computer Science and Information System, Universiti Teknologi Malaysia for hosting the research activities by providing a comfortable laboratory.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

C,c	-	canals
$f(a)$	-	function of a
i,j	-	integer numbers
$lc(a)$	-	location of a
m	-	index of external parties
n	-	index of internal user
$nm(a)$	-	name of a
$P(a)$	-	probability of a
ref	-	reference
$rp(a)$	-	running process of a
$sz(a)$	-	size of a
t	-	time
$th(a)$	-	threat of a
x	-	index of experiment
$\forall a$	-	all a
$\exists a$	-	there exist a
$\wedge$	-	and
$\vee$	-	or
Δt	-	time duration
μ_a	-	fuzzy membership function of a
υ	-	mean average

LIST OF APPENDICES

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

C,c	-	canals
$f(a)$	-	function of a
i,j	-	integer numbers
$lc(a)$	-	location of a
m	-	index of external parties
n	-	index of internal user
$nm(a)$	-	name of a
$P(a)$	-	probability of a
ref	-	reference
$rp(a)$	-	running process of a
$sz(a)$	-	size of a
t	-	time
$th(a)$	-	threat of a
x	-	index of experiment
$\forall a$	-	all a
$\exists a$	-	there exist a
$\wedge$	-	and
$\vee$	-	or
Δt	-	time duration
μ_a	-	fuzzy membership function of a
υ	-	mean average

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Overview

Nowadays Internet become more and more important to many organisations due to the advantages delivered by the Internet to support and facilitate their business and activities. The needs for Internet access exist on broad and differ activities. They range from doing a simple and daily routine such as updating antivirus database, reading stock market index, and obtaining weather report, to a complex and critical task such as conducting e-commerce and bank transaction. In fact, to some organisations Internet has become their main tool to conduct the business. Besides its benefit, Internet is widely known to become the sources of many security incidents as well (Anagnostakis, 2003; Huang *et al.*, 2004; Lai, 2004). Therefore organisations having Internet connection needs to give more protection to their information system and internal network in order to reduce or even to eliminate the Internet threat. Commonly this strategy is implemented by installing firewall between the protected internal network or intranet and the outside network or Internet.

During the first decade of its discovery i.e. in 1980 to 1990, firewalls had gained so much popularity (Arbaugh, 2002). However, the effectiveness of firewall to enforce security has been called into question recently. In the last three years, there are dozens reports disclosing the security incidents happening in many business and private information systems originated from and even facilitated by Internet. Consider the following facts. In 29 September 2002, Bugbear Internet virus was first

spotted in Malaysia, and within 24 hours this virus was spreading in over 100 countries and infecting million computers (Cherry, 2002). In October 2003, Spammers were reported for stealing the customers email addresses from Orbitz, the online travel agency (Associated Press, 2003). In June 2004, mysterious Internet virus stealing credit card information designed by a group of Russians was detected spreading through hundreds or possibly thousands of infected websites (Wired, 2004; Pruitt, 2004). In October 2004, Purdue's computer system was cracked by hackers that successfully gained unauthorized access to its internal network (Associated Press, 2004). This attack forced all users to change their password. The security incidents above show that current firewall technology exhibit serious vulnerabilities that can easily be exploited to commit attacks. In this research, this issue is addressed, thus better intranet protection can be enforced.

1.2 Background of Study

The performance of firewall to establish intranet protection has been receiving a lot of critics recently. Indeed, some reports disclose the vulnerabilities of current firewall implementations. Arbaugh (2002; 2003) describes the flaws of firewall as being insensitive when it deals with active content of Internet such as ActiveX and Javascript, user mobility, and peer-to-peer technology. While Eschelbeck (2000) and Hunt and Verwoerd (2003) highlight the static behaviour of firewall as the cause of its weaknesses. The static character of firewall shows that current firewall implementations depend heavily on the configuration and security rules explicitly defined by network administrator. This configuration is created at start-up and maintained along the duration of the firewall, without considering the condition of the surrounding network. Thus it is not surprising that many firewall implementations cannot cope with the raising threats of Internet.

Meanwhile CERT surveys on the threat of Internet show that during the year 2002, 2003, and 2004, malicious code in terms of worms and automatic intrusion becomes the most serious threats endangering many organizations networks (CERT, 2002; CERT, 2003; CERT, 2004). These results are similar to the survey conducted

by Whitman (2003) and CSI/FBI Annual Computer Crime and Security Survey (Power, 2002) that produce deliberate software attacks and virus respectively as the top rank of Internet threat. In fact, the report of Zetter (2004) supports the above survey results. It discloses that 45% of executable files downloaded from Internet such as Kazaa contain malicious code in term of viruses, worms, and Trojan horses.

Confronting the above survey results with the critics on firewall performance discloses a fact that the technological improvements of Internet contents have been undermining the intranet protection provided by current firewall technology. Although this phenomenon can be easily discovered from many organization networks, however survey held by CERT on electronic crimes (CERT, 2004) shows that firewall are still considered as the most effective security tools for protecting the trusted intranet from the danger of Internet. The survey also shows that firewall become the most common technologies deployed to combat electronic crimes. Therefore it is crucial to upgrade the mechanism of firewall in order to enable firewall to cope with the raising threats of Internet.

Some research groups have been holding the effort to develop smart firewall, with the purpose is to enable firewall aware of the security condition of the surrounding network. The work of Eschelbeck (2000) in Network Associates to develop active security notably initiated the development of this field by introducing active firewall concept i.e. firewall collaborate with other security tool, such as IDS and anti virus, in order to recognize any intrusion and possible vulnerabilities of the protected network. This approach however draws some critics from the researchers, such as Kamara *et al.* (2003) who reports the appearance of denial of service in Gauntlet active firewall as the product of implementing this concept. Moreover Haixin *et al.* (2000) emphasizes a number of possible security problems might be driven by the firewall such as asymmetric routing and performance decreasing.

Referring to the survey results of Internet security that produce malicious code as the most serious threat, Arbaugh (2002) applies a different approach to afford intranet protection. By considering the lesson obtained from the incidents of Code Red worm, which the spreading of this code could not be prevented although the software patch for stopping it’s spreading and it’s action had been available, an

active security management was proposed. This concept puts greater responsibility on the management of the organization to manually managing its network security. However, by considering the rapid growth of malicious code that always increases from year to year (Kientzle and Elder, 2003), this approach would probably be more burdening the management and fall short in the implementation.

Considering the importance of the firewall to many organization networks, and learning from the past lessons for providing more secure systems, a well-defined security strategy that appropriately combats the Internet threat while in the same time facilitating the connection to the external parties is required. Thus a study on activating the mechanism of network firewall seems become a promising approach in dealing with this issue.

1.3 Objectives

As discussed in the research background, the reason for conducting this study is due to incapability of existing firewall methods to deal with the threat of Internet driven by the growth of Internet technology. The static behaviour of firewall is suspected causing this problem. In fact, based on CERT survey (CERT, 2004) firewalls are still required by most organization information systems to establish internal network protection. Thus the main objective of this study is to develop correct strategy for protecting intranet from the threats of Internet, in which the methods to activate and to improve the mechanism of firewall are observed. It is expected that this approach is capable to recognize any possible threats originating from Internet, and to restrict the Internet threats from entering the protected intranet. To accomplish this task, the study needs to achieve some other objectives as follows:

(i) Formulating the security strategy to combat the appearance of Internet threats. This strategy is built by identifying the possible security conditions caused by accessing Internet.

(ii) Developing a model of active firewall to host the implementation of the security strategy developed in point (i).

(iii) Formulating and developing active firewall methods as the implementation of each security strategy developed in point (i) in order to combat the Internet threats.

(iv) Conducting evaluation on each active firewall method in order to measure the applicability of network firewall to provide intranet protection. Security analysis on each method together with comparative study among the developed methods and to known firewall techniques are held in this study

1.4 Research Scopes

To properly conducting this study, the conditions limiting the research are set up as follows:

(i) It is assumed that all network packets passing through the firewall are un-encrypted. Therefore analysis on the content of network packet can be done without involving any mechanism to decrypt the data. This assumption greatly reduces the effort to deal with the content of the packets, thus the work can be focused on observing the mechanism to activate the firewall.

(ii) It is assumed that the speed to transfer data from Internet to intranet or vice versa is much faster than the speed required by the internal user to communicate with more than one external party, or jumping from one external party to the others. Thus, at any time the internal user would only be able to communicate with an external party.

(iii) With regard to the mechanism of network firewall, the security methods developed in this study are intended to guard the gate of intranet. Thus any mechanisms to secure the individual host residing inside the intranet, or the

individual users and applications, are not taken into account since this operation required a mechanism that is beyond the capability of network firewall.

1.5 Contributions

This study improves the mechanism of network firewall in order to provide better protection to the intranet by combating the threats of Internet, and in the same time to cope with the increasing technology of Internet. To accomplish this research, a model of Internet access that consists of model internal users and model external parties is developed. Based on these models, a set of security strategies to minimize the Internet threats is produced. Implementations of the developed security strategies are conducted by activating the mechanism of network firewall, since this device is considered capable to prevent the Internet threats from flowing into the protected intranet. Moreover based on the CERT survey (CERT, 2004), firewall is considered as the most deployed security tools to combat Internet threats. Thus the following results are delivered:

(i) The generic model of intranet users, the generic model of external parties, and the generic model of Internet access. From the model of Internet access, a set of security strategies to combat Internet threats is produced, in which it is created by identifying the security conditions of Internet access.

(ii) A model of active firewall utilizing the concept of canalisation is developed. This model is used for hosting the implementation of the developed security strategies.

(iii) Methods for initialising active firewall i.e. close condition, open condition, and lattice-based.

(iv) Methods for handling runtime process i.e. adaptive security rules update using fuzzy reasoning, suspicious process detection using agent-based module and zero-based configuration.

(v) Research evaluation on each developed active firewall method is presented. The evaluation consist of security analysis and comparative study among the developed methods and with the known firewall methods i.e. no firewall configuration, static and dynamic firewall. This evaluation is to measure the applicability of each method.

1.6 Organisation of Report

This report is organised as follow. Chapter 1, as has been presented, describes the background, objective, scope, and contributions of this study. Chapter 2 presents the results of studying the literatures in the effort of developing firewall technology. Chapter 3 discusses the concept of active firewall and formulating the strategy to combat Internet threats. A generic model of intranet users, a generic model of external parties, a model of Internet access, and active firewall model are presented in this chapter. Chapter 4 presents the development of the initialisation process of active firewall, while Chapter 5 to 7 presents the development of runtime process. Chapter 5 deals with adaptive update of security rules using fuzzy reasoning in which the content of network packet originating from external parties are evaluated. Chapter 6 affords to have a mechanism to monitor the activities of internal users by scrutinizing the running process of each internal host using distributed agent-based module. And Chapter 7 introduces zero configurations to minimize the available services at runtime. Chapter 8 conducts the evaluation on all of the methods proposed above using security analysis and comparative study with the known firewall methods. Finally, conclusion of this study is given in Chapter 9.

CHAPTER 1

INTRODUCTION

1.1 Overview

Nowadays Internet become more and more important to many organisations due to the advantages delivered by the Internet to support and facilitate their business and activities. The needs for Internet access exist on broad and differ activities. They range from doing a simple and daily routine such as updating antivirus database, reading stock market index, and obtaining weather report, to a complex and critical task such as conducting e-commerce and bank transaction. In fact, to some organisations Internet has become their main tool to conduct the business. Besides its benefit, Internet is widely known to become the sources of many security incidents as well (Anagnostakis, 2003; Huang *et al.*, 2004; Lai, 2004). Therefore organisations having Internet connection needs to give more protection to their information system and internal network in order to reduce or even to eliminate the Internet threat. Commonly this strategy is implemented by installing firewall between the protected internal network or intranet and the outside network or Internet.

During the first decade of its discovery i.e. in 1980 to 1990, firewalls had gained so much popularity (Arbaugh, 2002). However, the effectiveness of firewall to enforce security has been called into question recently. In the last three years, there are dozens reports disclosing the security incidents happening in many business and private information systems originated from and even facilitated by Internet. Consider the following facts. In 29 September 2002, Bugbear Internet virus was first

spotted in Malaysia, and within 24 hours this virus was spreading in over 100 countries and infecting million computers (Cherry, 2002). In October 2003, Spammers were reported for stealing the customers email addresses from Orbitz, the online travel agency (Associated Press, 2003). In June 2004, mysterious Internet virus stealing credit card information designed by a group of Russians was detected spreading through hundreds or possibly thousands of infected websites (Wired, 2004; Pruitt, 2004). In October 2004, Purdue's computer system was cracked by hackers that successfully gained unauthorized access to its internal network (Associated Press, 2004). This attack forced all users to change their password. The security incidents above show that current firewall technology exhibit serious vulnerabilities that can easily be exploited to commit attacks. In this research, this issue is addressed, thus better intranet protection can be enforced.

1.2 Background of Study

The performance of firewall to establish intranet protection has been receiving a lot of critics recently. Indeed, some reports disclose the vulnerabilities of current firewall implementations. Arbaugh (2002; 2003) describes the flaws of firewall as being insensitive when it deals with active content of Internet such as ActiveX and Javascript, user mobility, and peer-to-peer technology. While Eschelbeck (2000) and Hunt and Verwoerd (2003) highlight the static behaviour of firewall as the cause of its weaknesses. The static character of firewall shows that current firewall implementations depend heavily on the configuration and security rules explicitly defined by network administrator. This configuration is created at start-up and maintained along the duration of the firewall, without considering the condition of the surrounding network. Thus it is not surprising that many firewall implementations cannot cope with the raising threats of Internet.

Meanwhile CERT surveys on the threat of Internet show that during the year 2002, 2003, and 2004, malicious code in terms of worms and automatic intrusion becomes the most serious threats endangering many organizations networks (CERT, 2002; CERT, 2003; CERT, 2004). These results are similar to the survey conducted

by Whitman (2003) and CSI/FBI Annual Computer Crime and Security Survey (Power, 2002) that produce deliberate software attacks and virus respectively as the top rank of Internet threat. In fact, the report of Zetter (2004) supports the above survey results. It discloses that 45% of executable files downloaded from Internet such as Kazaa contain malicious code in term of viruses, worms, and Trojan horses.

Confronting the above survey results with the critics on firewall performance discloses a fact that the technological improvements of Internet contents have been undermining the intranet protection provided by current firewall technology. Although this phenomenon can be easily discovered from many organization networks, however survey held by CERT on electronic crimes (CERT, 2004) shows that firewall are still considered as the most effective security tools for protecting the trusted intranet from the danger of Internet. The survey also shows that firewall become the most common technologies deployed to combat electronic crimes. Therefore it is crucial to upgrade the mechanism of firewall in order to enable firewall to cope with the raising threats of Internet.

Some research groups have been holding the effort to develop smart firewall, with the purpose is to enable firewall aware of the security condition of the surrounding network. The work of Eschelbeck (2000) in Network Associates to develop active security notably initiated the development of this field by introducing active firewall concept i.e. firewall collaborate with other security tool, such as IDS and anti virus, in order to recognize any intrusion and possible vulnerabilities of the protected network. This approach however draws some critics from the researchers, such as Kamara *et al.* (2003) who reports the appearance of denial of service in Gauntlet active firewall as the product of implementing this concept. Moreover Haixin *et al.* (2000) emphasizes a number of possible security problems might be driven by the firewall such as asymmetric routing and performance decreasing.

Referring to the survey results of Internet security that produce malicious code as the most serious threat, Arbaugh (2002) applies a different approach to afford intranet protection. By considering the lesson obtained from the incidents of Code Red worm, which the spreading of this code could not be prevented although the software patch for stopping it’s spreading and it’s action had been available, an

active security management was proposed. This concept puts greater responsibility on the management of the organization to manually managing its network security. However, by considering the rapid growth of malicious code that always increases from year to year (Kientzle and Elder, 2003), this approach would probably be more burdening the management and fall short in the implementation.

Considering the importance of the firewall to many organization networks, and learning from the past lessons for providing more secure systems, a well-defined security strategy that appropriately combats the Internet threat while in the same time facilitating the connection to the external parties is required. Thus a study on activating the mechanism of network firewall seems become a promising approach in dealing with this issue.

1.3 Objectives

As discussed in the research background, the reason for conducting this study is due to incapability of existing firewall methods to deal with the threat of Internet driven by the growth of Internet technology. The static behaviour of firewall is suspected causing this problem. In fact, based on CERT survey (CERT, 2004) firewalls are still required by most organization information systems to establish internal network protection. Thus the main objective of this study is to develop correct strategy for protecting intranet from the threats of Internet, in which the methods to activate and to improve the mechanism of firewall are observed. It is expected that this approach is capable to recognize any possible threats originating from Internet, and to restrict the Internet threats from entering the protected intranet. To accomplish this task, the study needs to achieve some other objectives as follows:

(i) Formulating the security strategy to combat the appearance of Internet threats. This strategy is built by identifying the possible security conditions caused by accessing Internet.

(ii) Developing a model of active firewall to host the implementation of the security strategy developed in point (i).

(iii) Formulating and developing active firewall methods as the implementation of each security strategy developed in point (i) in order to combat the Internet threats.

(iv) Conducting evaluation on each active firewall method in order to measure the applicability of network firewall to provide intranet protection. Security analysis on each method together with comparative study among the developed methods and to known firewall techniques are held in this study

1.4 Research Scopes

To properly conducting this study, the conditions limiting the research are set up as follows:

(i) It is assumed that all network packets passing through the firewall are un-encrypted. Therefore analysis on the content of network packet can be done without involving any mechanism to decrypt the data. This assumption greatly reduces the effort to deal with the content of the packets, thus the work can be focused on observing the mechanism to activate the firewall.

(ii) It is assumed that the speed to transfer data from Internet to intranet or vice versa is much faster than the speed required by the internal user to communicate with more than one external party, or jumping from one external party to the others. Thus, at any time the internal user would only be able to communicate with an external party.

(iii) With regard to the mechanism of network firewall, the security methods developed in this study are intended to guard the gate of intranet. Thus any mechanisms to secure the individual host residing inside the intranet, or the

individual users and applications, are not taken into account since this operation required a mechanism that is beyond the capability of network firewall.

1.5 Contributions

This study improves the mechanism of network firewall in order to provide better protection to the intranet by combating the threats of Internet, and in the same time to cope with the increasing technology of Internet. To accomplish this research, a model of Internet access that consists of model internal users and model external parties is developed. Based on these models, a set of security strategies to minimize the Internet threats is produced. Implementations of the developed security strategies are conducted by activating the mechanism of network firewall, since this device is considered capable to prevent the Internet threats from flowing into the protected intranet. Moreover based on the CERT survey (CERT, 2004), firewall is considered as the most deployed security tools to combat Internet threats. Thus the following results are delivered:

(i) The generic model of intranet users, the generic model of external parties, and the generic model of Internet access. From the model of Internet access, a set of security strategies to combat Internet threats is produced, in which it is created by identifying the security conditions of Internet access.

(ii) A model of active firewall utilizing the concept of canalisation is developed. This model is used for hosting the implementation of the developed security strategies.

(iii) Methods for initialising active firewall i.e. close condition, open condition, and lattice-based.

(iv) Methods for handling runtime process i.e. adaptive security rules update using fuzzy reasoning, suspicious process detection using agent-based module and zero-based configuration.

(v) Research evaluation on each developed active firewall method is presented. The evaluation consist of security analysis and comparative study among the developed methods and with the known firewall methods i.e. no firewall configuration, static and dynamic firewall. This evaluation is to measure the applicability of each method.

1.6 Organisation of Report

This report is organised as follow. Chapter 1, as has been presented, describes the background, objective, scope, and contributions of this study. Chapter 2 presents the results of studying the literatures in the effort of developing firewall technology. Chapter 3 discusses the concept of active firewall and formulating the strategy to combat Internet threats. A generic model of intranet users, a generic model of external parties, a model of Internet access, and active firewall model are presented in this chapter. Chapter 4 presents the development of the initialisation process of active firewall, while Chapter 5 to 7 presents the development of runtime process. Chapter 5 deals with adaptive update of security rules using fuzzy reasoning in which the content of network packet originating from external parties are evaluated. Chapter 6 affords to have a mechanism to monitor the activities of internal users by scrutinizing the running process of each internal host using distributed agent-based module. And Chapter 7 introduces zero configurations to minimize the available services at runtime. Chapter 8 conducts the evaluation on all of the methods proposed above using security analysis and comparative study with the known firewall methods. Finally, conclusion of this study is given in Chapter 9.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Firewall is defined as a device to isolate the protected internal network from the untrusted external parties in the networking world (Garfinkel and Spafford, 1997). This device has gain so much popularity in the last decade for protecting many organisations network, and become a part of organisations security strategy even today. Motivated by this condition, research on developing firewall have been reactivated in the last half-decade. Many types of firewall mechanisms have been discovered during this period. This chapter aims to literally study the latest development of network firewalls. Report produced from the last several years are explored. This approach becomes a fundamental step towards disclosing the issue in the development of firewall technology. Knowledge obtained from this stage is used for formulating a new method to provide secure internal network with regard to Internet connection.

2.2 Internet Threat

Survey of Whitman (2003) and CERT (2004) discover a fact that Internet connection is frequently exploited. These incidents are purposely committed in order to breach the security of many organizations networks. Some efforts to study Internet

threats (CERT, 2002; CERT 2003; Whitman, 2003; Kienzle and Elder, 2003; Anagnostakis, 2003; Huang *et al*., 2004; CERT, 2004) found that vary attacks can be committed from the Internet. They span from the action of hacker and eavesdropper, information stealing, to the spreading of virus, worm, backdoor, etc. In order to get thorough figure of these attacks, here Internet attacks are classified into two categories i.e. interactive attack and malicious code. The former is characterized by interactive activities held by the offenders to obtain unauthorized access or even to gain control over a network or internal hosts. The attacks in this class include network and host scanning and penetration, hackers, remote buffer overflows, eavesdropping, etc. The latter category of attack is always started by the injection of malicious code into the internal network, and then it is followed by the automatic execution of the code that compromises the internal network. Currently there are broad types of malicious code can be discovered, among them are computer viruses and worm, Trojan software, back door, program causing buffer over flows and spy software.

While firewalls and other network security methods such as VPN and encryption are very effective to deter the action of interactive attack, the action of malicious code have been causing a great deal of losses (Christodorescu and Jha, 2003; Lai *et al.*, 2004). According to the survey of CERT, during the year 2002, 2003 and 2004, malicious code becomes the top threat compromising many organizations information system (CERT, 2002; CERT, 2003; CERT, 2004). The factors facilitating malicious code to gain so successful record in attacking organization network is due to the absence of preventive security methods to prevent this type of program from reaching the internal network, besides smart injection of the code to spread and replicate the contiguous program or to directly attacking the networking systems. Currently the available security methods such as intrusion detection and antivirus are able to respond only if the malicious code has been injected or the attack is appearing inside the internal network. The vendors of the security software commonly assume that the security software is quickly enough to respond on the intrusion. However these types of security tools mostly suffer from high runtime overhead (Anagnostakis, 2003; Toth and Kruegel, 2002). Problem of late patching the software increase the vulnerabilities of these method as well (Lai *et al.*, 2004; Arbaugh, 2002). Meanwhile, the available firewall technology has been

known to have no capability to sense the security conditions of the surrounding environment (Arbaugh, 2003). Therefore it is not surprising that many proposed system employing the integration between firewall and intrusion detection system and even antivirus software (Venkatesan and Bhattacharya, 1997; Eschelbeck, 2000; Hwang and Gangadharan, 2001) fall short in providing secure network environment.

2.3 Recent Development of Firewall Technology

As widely known, the purpose of having firewall is to protect the intranet from the threats of external networks. It is achieved by preventing the threats from getting into the protected network and compromising the organization information system and data (Ogletree, 2000). The mechanism of firewall to handle this task is to allow or to block traffic flowing from the intranet to outside or vice versa. This way, any sensitive information of the organization can be kept inside and prevented not to flow outside while any untrusted content that may jeopardizing internal system and data can be prevented from flowing inside. However, due to the growth of Internet technology, the task of firewall becomes more difficult. As noted by Arbaugh (2002; 2003), there are three factors that may undermine the protection of firewalls i.e. active content of Internet, mobile users and peer-to-peer technology. Therefore security breaches that exploit these factors would probably be committed successfully (Bellovin, 1999; Arbaugh, 2002; Kienzle and Elder, 2003). Hence the efforts to develop firewall mostly focuses on fixing these problems.

2.3.1 Distributed Firewalls

Concept of distributed firewalls was originally proposed by Bellovin (1999) to deal with mobile users. In Bellovin proposition, distributed firewall are defined as a firewall system that contain a set of individual firewalls installed in each host of internal network, which every individual firewall enforces security policy defined centrally by the administrator. This system is claimed capable to solve the problem

of insider attacks that compromise the intranet protected by traditional firewall. An effort to implement Bellovin concept into a prototype system was conducted by Ioannidis *et al.* (2000). However this concept draws some critiques as well. Although Bellovin (1999) proves that imposing access control as individual host leads to fine-grained security process, communication between individual hosts leads to a high runtime overhead (Hwang and Gangadharan, 2001). The use of encryption in IPSEC also creates a problem for facilitating network-based intrusion as reported by Xian *et al.* (2002). This problem is due to insensitive mechanism of distributed firewall to the existence of malicious host that may commit a network-based attack by communicating with other host (Payne and Markham, 2001). The last problem is also caused by the mechanism of distributed firewall to leave network boundary concept as applied by traditional firewall. Other weakness of distributed firewall as noted by Payne and Markham (2001) is the vulnerability of untrusted operating system since it run on top of operating system in every host in the intranet. Issue of tamper resistance is raised from this condition, as it is difficult to control the widely deployed individual firewalls.

Development of microfirewall as reported by Hwang and Gangadharan (2001) notably solves the problem of malicious host and reduced runtime overhead suffered by Bellovin concept. Here microfirewall were implemented as the functional modules acting as packet filtering rules constructed at the kernel space. A single microfirewall is installed in each host of the protected network. Architecture developed by Hwang and Gangadharan is to combine microfirewall, policy manager and the gate firewall to establish security in an intranet. The purpose of putting gateway firewall here is to have network boundary for isolating the intranet from the outside network. This approach is expected to eliminate malicious host as mentioned previously. Meanwhile, the employed policy manager is used for updating security policy in each microfirewall unit. Mobile agents, CORBA, and RMI middleware are used optionally in this task. This strategy however creates a trade off between speeds and robustness. Mobile agents that are capable to provide robust mechanism as they can easily be created, suspended, terminated and reboot dynamically, only deliver low speed and even vulnerable to be attacked by other agents or hosts.

The work of Payne and Markham (2001) aims to solve tamper resistance problems caused by untrusted operating system. Distributed embedded firewall developed in this effort utilizes embedded computing. The firewall is built on top of network interface card to avoid the problem of untrusted host operating system. Although this method is claimed solving the problem of untrusted host operating system and protect the intranet from malicious hosts using gateway firewall that guards network perimeter, firewall developed using this method cannot achieve dynamic security policy since the embedded method implemented on the interface card, thus it tends to have static configuration. Moreover, communication among hosts for applying security policy can also lead to high runtime overhead.

Xian *et al.* (2002) intends to improve Bellovin concept by revisiting the development of distributed microfirewall system. Domain type enforce is used in this project to enforce mandatory access control to the firewall. The purpose is to keep microfirewall being tampered by unwitting insiders or malicious host. This effort however cannot prevent network from the possibility of malicious host as gateway-firewall is removed from the implementation. Thus no security perimeter is defined to protect the internal network.

2.3.2 Adaptive Firewalls

Research on adaptive firewall is intended for understanding and dealing with the changing conditions of network security as well as the Internet threats. Although research on adaptive firewall have been running for sometimes, the author cannot determine who and when started the development of this method. However the oldest effort found in the literature for developing adaptive firewall is the work of Venkatesan and Bhattacharya (1997). Firewall developed in this effort is capable to adapt the security policy with the changes of the network threat endangering the internal system. To measure the threat, users activities is captured and analysed at any point of time, and then trusted level of each user defined priorly is adjusted. Whenever a threat is discovered from the log of user activities causes user trusted level to be decreased, otherwise it is increased. And to monitor the appearance of

intrusive action, existing intrusion detection such as expert system and statistically anomaly detection are employed. The method used in this system leads to create a security hole as it may trust the user too soon, or raise even denial of service if the employed intrusion detection system suffer from false positive. Moreover, the application of time period creates an opportunity that malicious users can be trusted. And as widely known, statistical anomaly has a pitfall that an intruder can train the system to believe on the suspicious activities, and the expert system only deals with the known intrusion. Therefore a new suspicious behaviour of user will not be detected by the system.

The development of adaptive packet filters by Reumann *et al.* (2001) notably solves the problem of security holes caused by malicious users. The approach used in the method is to allow alternative security rules to be loaded into a filter controller. The rules here are ordered by the increasing restrictiveness, which the least restrictive does not filter the incoming traffic while the most restrictive drops all packets. Users of the network are assigned specific security rule to access the outside network. This way any potentially malicious users can be identified prior to the intrusion, and therefore specific access policy can be enforced. This approach however reduces network access performance since firewall must check each incoming packet against its rule-base, and the mechanism of switching among the rule as well. The performance is even dropped when the number of rules becomes large.

Zou *et al.* (2003) afford to build an intelligent firewall based on adaptive security method. Fuzzy logic is employed to handle security classification based on the source and destination of the packet. This approach besides presenting an advantage to bring soft computing to the security world, it also causes performance decreasing due to algorithm complexity to process each flowing packet using fuzzy algorithm. To cope with this problem, Zou *et al.* simplify the analysis using a set of predetermined security levels for the source and destination of network packets. This approach results in producing shallow packet analysis.

The problem of performance overhead is also suffered by adaptive firewall of Verwoerd and Hunt (2002). Although this effort simplifies the mechanism to develop

adaptive security using packet filtering, performance decreasing due to processing intensive and complex calculations cannot be avoided, especially for computing network bandwidth. This problem indeed becomes the open issue in the research of adaptive firewall.

2.3.3 Hardware-Based Firewalls

The development of hardware-based firewall is motivated by slow processing speed of software-based system. The approach used in this research is to hold network packet processing in the hardware. The strategy to speed up the process is due to elimination of many software-based procedures. A number of efforts, such as the work of Kayssi *et al.* (2000) and Lee *et al.* (2002), afford to implement this approach by developing an FGPA (Field Programmable Gate Array) firewall. They make use the reprogramability of FPGA chips to process network packets. Different types of security rules are implemented. These works however carry a number of drawbacks leading to inefficient firewall implementation. Most of hardware-based firewalls experience difficulties in mapping security rules into the chips. Meanwhile, only limited security rules can be deployed in the chip due to hardware constraint. And as widely known, every reprogramable chip has its operational lifetime. Therefore exploiting its reprogramability will probably shorten its lifetime. Moreover, some complexities are raised by the effort to build dynamic security. Firewall developed by Kayssi *et al.* (2000) requires to power off the system for updating security rules. It causes performance decreasing since network transaction can be aborted when the system is down.

The work of Lee *et al.* (2002) to overcome the problem of security rule reconfiguration, by employing a high level programming language namely Ponder, also create a drawback in holding rule modification. This effort creates more algorithm complexity since it requires the framework to integrate the mechanism of hardware and software. And the effort of Lockwood *et al.* (2003) notably creates the extension of FGPA-based firewall to detect malicious content of Internet traffic. This work implements an algorithm to process the flowing packet in the hardware.

However this system cannot avoid hardware-based constraint, thus only limited filtering rules can be implemented.

2.3.4 Active Firewalls

The term of active firewalls was originally discovered from the result of developing security tools in two research-domains i.e. active network (Alexander *et al.*, 1999) and regular network (Eschelbeck, 2000). In the domain of active network, active firewalls actually have similar function as regular firewall devices, however they are developed to serve the mechanism of active network. It is worth to note that up to this time no serious active firewall development in the domain of active network is reported. It is due to the definition and the implementation of the platform of active network that are still progressing. The implementation of active firewall under active network such as presented by Alexander *et al.* (1999), Silva *et al.* (2001) and Hicks *et al.* (2003) are merely to prove that the proposed active networks capable to cope with networking devices.

On the contrary, researches for developing firewall in regular network have been delivering some useful methods by building active firewall. The basic notion of this method is to have active mechanism to detect and combat the threats, thus any attacks endangering the protected network can be prevented. The effort of Eschelbeck (2000), notably initiates the development of active firewall concept. In this work, active firewall is built by collaborating dynamic firewall with other security components to respond on the changing threat. This system works by actively detecting any intrusion using antivirus and intrusion detection. The active firewall proposed by Eschelbeck is also equipped with the function of vulnerability scanner to detect the existing host vulnerabilities. Although many types of security software are employed in this method, however this approach has no capabilities to measure the threat originating from the Internet. Therefore, it is still uncertain whether active firewall proposed by Eschelbeck capable to react on the changing Internet threat since it has to wait for an intrusion to update its security policy. Formalization of this method does not include any experimental result to justify the

claim nor any measurement to prove that active firewall has preventive action to secure network.

Lehtonen (2003) presented a different approach to build activate firewall. Although it is motivated by the programmability of active network and the idea of Eschelbeck firewall, the work is intended to serve the wireless network. Using a packet filtering functionality, security rules are built into a tree-like structure. Each network packet passing through the firewall is directed to a suitable leaf and branches nodes of the active firewall to enter or leaf the intranet. As clearly shown from this mechanism, this work fails to develop the mechanism for recognizing network threat. The function of firewall here is only to organize security policy, a similar method as presented by regular packet filtering firewall.

Different approach to build active firewall is presented by Hunt and Verwoerd (2003) for developing reactive firewall. Utilizing packet-filtering rules, the proposed method affords to provide safe network environment by identifying the requirements of internal network in term of bandwidth reservation and network transaction. The computation to specify the bandwidth usage is conducted based on the flowing traffic, which the source and destination of each packet are identified and compared against a predetermined security rules. Besides its complex computation that creates performance overhead, this method does not consider the condition of external network that potentially become the sources of many type of Internet threats.

2.4 Firewall Validation Standard

Due to the importance of firewall in the Internet community, some standards have been created for the purpose of validating firewall technologies. It is important to fulfil the requirements defined in these standards for creating a new firewall technique, since it warrants the usability of the developed firewall. This section aims to describe these standards, so the appropriate validation mechanism can be identified and be referred in validating the firewall developed in this research.

Basically there are two aspects influencing the behaviour of firewall devices, namely performance and security. Internet community uses a different standard for dealing with each of these aspects (Hickman *et al.*, 2003). For the purpose of validating firewall performance, some benchmarking terminologies have been presented and well described in RFC 2647 (Newman, 1999). This standard discusses the definition used in the benchmarking test such as allowed traffic, authentication, bit forwarding rate, connection, good put etc. The methodology for conducting benchmarking is presented in RFC 3511 (Hickman *et al.*, 2003). In this standard, the set up for testing firewall and the parameters to be tested are described. However as noted by Hickman *et al.*, the aspect of security is beyond the scope of this standard.

Meanwhile for the purpose of validating firewall security, the behaviour of and requirements for Internet firewalls are standardize in RFC 2979 (Freed, 2000). In this standard, two aspects, namely security and usability, are considered become the main function of firewall. However there is tradeoff between both aspects. Therefore a transparency rule is defined in this standard as the requirements for improving the security of firewall. This rule states that the introduction of firewall in the networking system must not cause unintended failures of the legitimate and standards compliant usage that would work were the firewall not present. Thus any security improvement will not prevent people from performing a useful works.

2.5 Discussion

Firewall developments are classified into four methods i.e. distributed, adaptive, hardware-based, and active firewalls. The first three methods have had a clear definition and formulation, therefore they become attractive to the researchers for improving firewall mechanism. Currently a number of reports have been presenting the achievements in enhancing firewall based on these methods. However, some issues are raised as well. Common pitfall suffered by the firewalls built using these methods are performance decreasing due to the increased of algorithm complexity and the processing load. Besides, these methods also suffer from specific drawbacks, such as the possible existence of malicious host in the distributed firewall

and hardware constraint of hardware-based firewall. Those drawbacks, of course, become the barrier for the acceptance of firewall development.

And as mentioned earlier in Session 2.2, the factor that has more influence in network security is the appearance of Internet threat in which some surveys (CERT, 2002; CERT, 2003; CERT, 2004; Whitman, 2003; Anagnostakis, 2003; Kientzle and Elder, 2003) conclude that malicious code has the bigger portion for causing this threat. Therefore a mechanism to keep the bad stuff of Internet from coming into the protected intranet is required. The method based on distributed, adaptive, and hardware-based firewalls apparently show lack performance to stop the action of malicious code. However, adaptive firewall that is combined with intrusion detection or antivirus tools may reduce this threat. This strategy is significantly influenced by the performance of the employed security tools, such as intrusion detection system that may suffer from false positive and false negative, or antivirus software that can be compromised due to late updating virus database. Moreover communication method between firewall and other security tools potentially create performance decreasing that causes security holes in the network. By considering the issues above, active firewall concept introduced by Eschelbeck (2000) has the promising mechanism to provide security for the intranet. The main idea behind this concept is to equip firewall with active mechanism to detect, recognize and take an action against the Internet threat. This method however suffers from similar problem as adaptive firewall combined with other security tools, since the mechanism applied in this effort relies on the collaboration with other methods. Meanwhile, other works such as presented by Lehtonen (2003) and Hunt and Verwoerd (2003) fail to accommodate the requirements to deal with the changing pattern of network threats. Thus improvement for developing active firewall is still required.

2.6 Summary

The development of firewall technology that consists of distributed, adaptive, hardware-based, and active firewall has been presented. The purpose is to discover the direction of the research in the field of network firewall and to highlight the issue produced from these efforts. Referring to the survey results that produce malicious code as the top threat of Internet (CERT, 2002; CERT, 2003; CERT, 2004), only the adaptive and active firewalls that are capable to deter this threat. However if we compare adaptive and active firewall, it shows that adaptive method tends to have passive mechanism that wait for the incidents to happen before changing the security rules, thus it might be too late to react. Active firewall has more promising approach in protecting intranet since this method has active mechanism to search any indication of security threats.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Firewall is defined as a device to isolate the protected internal network from the untrusted external parties in the networking world (Garfinkel and Spafford, 1997). This device has gain so much popularity in the last decade for protecting many organisations network, and become a part of organisations security strategy even today. Motivated by this condition, research on developing firewall have been reactivated in the last half-decade. Many types of firewall mechanisms have been discovered during this period. This chapter aims to literally study the latest development of network firewalls. Report produced from the last several years are explored. This approach becomes a fundamental step towards disclosing the issue in the development of firewall technology. Knowledge obtained from this stage is used for formulating a new method to provide secure internal network with regard to Internet connection.

2.2 Internet Threat

Survey of Whitman (2003) and CERT (2004) discover a fact that Internet connection is frequently exploited. These incidents are purposely committed in order to breach the security of many organizations networks. Some efforts to study Internet

threats (CERT, 2002; CERT 2003; Whitman, 2003; Kienzle and Elder, 2003; Anagnostakis, 2003; Huang *et al*., 2004; CERT, 2004) found that vary attacks can be committed from the Internet. They span from the action of hacker and eavesdropper, information stealing, to the spreading of virus, worm, backdoor, etc. In order to get thorough figure of these attacks, here Internet attacks are classified into two categories i.e. interactive attack and malicious code. The former is characterized by interactive activities held by the offenders to obtain unauthorized access or even to gain control over a network or internal hosts. The attacks in this class include network and host scanning and penetration, hackers, remote buffer overflows, eavesdropping, etc. The latter category of attack is always started by the injection of malicious code into the internal network, and then it is followed by the automatic execution of the code that compromises the internal network. Currently there are broad types of malicious code can be discovered, among them are computer viruses and worm, Trojan software, back door, program causing buffer over flows and spy software.

While firewalls and other network security methods such as VPN and encryption are very effective to deter the action of interactive attack, the action of malicious code have been causing a great deal of losses (Christodorescu and Jha, 2003; Lai *et al.*, 2004). According to the survey of CERT, during the year 2002, 2003 and 2004, malicious code becomes the top threat compromising many organizations information system (CERT, 2002; CERT, 2003; CERT, 2004). The factors facilitating malicious code to gain so successful record in attacking organization network is due to the absence of preventive security methods to prevent this type of program from reaching the internal network, besides smart injection of the code to spread and replicate the contiguous program or to directly attacking the networking systems. Currently the available security methods such as intrusion detection and antivirus are able to respond only if the malicious code has been injected or the attack is appearing inside the internal network. The vendors of the security software commonly assume that the security software is quickly enough to respond on the intrusion. However these types of security tools mostly suffer from high runtime overhead (Anagnostakis, 2003; Toth and Kruegel, 2002). Problem of late patching the software increase the vulnerabilities of these method as well (Lai *et al.*, 2004; Arbaugh, 2002). Meanwhile, the available firewall technology has been

known to have no capability to sense the security conditions of the surrounding environment (Arbaugh, 2003). Therefore it is not surprising that many proposed system employing the integration between firewall and intrusion detection system and even antivirus software (Venkatesan and Bhattacharya, 1997; Eschelbeck, 2000; Hwang and Gangadharan, 2001) fall short in providing secure network environment.

2.3 Recent Development of Firewall Technology

As widely known, the purpose of having firewall is to protect the intranet from the threats of external networks. It is achieved by preventing the threats from getting into the protected network and compromising the organization information system and data (Ogletree, 2000). The mechanism of firewall to handle this task is to allow or to block traffic flowing from the intranet to outside or vice versa. This way, any sensitive information of the organization can be kept inside and prevented not to flow outside while any untrusted content that may jeopardizing internal system and data can be prevented from flowing inside. However, due to the growth of Internet technology, the task of firewall becomes more difficult. As noted by Arbaugh (2002; 2003), there are three factors that may undermine the protection of firewalls i.e. active content of Internet, mobile users and peer-to-peer technology. Therefore security breaches that exploit these factors would probably be committed successfully (Bellovin, 1999; Arbaugh, 2002; Kienzle and Elder, 2003). Hence the efforts to develop firewall mostly focuses on fixing these problems.

2.3.1 Distributed Firewalls

Concept of distributed firewalls was originally proposed by Bellovin (1999) to deal with mobile users. In Bellovin proposition, distributed firewall are defined as a firewall system that contain a set of individual firewalls installed in each host of internal network, which every individual firewall enforces security policy defined centrally by the administrator. This system is claimed capable to solve the problem

of insider attacks that compromise the intranet protected by traditional firewall. An effort to implement Bellovin concept into a prototype system was conducted by Ioannidis *et al.* (2000). However this concept draws some critiques as well. Although Bellovin (1999) proves that imposing access control as individual host leads to fine-grained security process, communication between individual hosts leads to a high runtime overhead (Hwang and Gangadharan, 2001). The use of encryption in IPSEC also creates a problem for facilitating network-based intrusion as reported by Xian *et al.* (2002). This problem is due to insensitive mechanism of distributed firewall to the existence of malicious host that may commit a network-based attack by communicating with other host (Payne and Markham, 2001). The last problem is also caused by the mechanism of distributed firewall to leave network boundary concept as applied by traditional firewall. Other weakness of distributed firewall as noted by Payne and Markham (2001) is the vulnerability of untrusted operating system since it run on top of operating system in every host in the intranet. Issue of tamper resistance is raised from this condition, as it is difficult to control the widely deployed individual firewalls.

Development of microfirewall as reported by Hwang and Gangadharan (2001) notably solves the problem of malicious host and reduced runtime overhead suffered by Bellovin concept. Here microfirewall were implemented as the functional modules acting as packet filtering rules constructed at the kernel space. A single microfirewall is installed in each host of the protected network. Architecture developed by Hwang and Gangadharan is to combine microfirewall, policy manager and the gate firewall to establish security in an intranet. The purpose of putting gateway firewall here is to have network boundary for isolating the intranet from the outside network. This approach is expected to eliminate malicious host as mentioned previously. Meanwhile, the employed policy manager is used for updating security policy in each microfirewall unit. Mobile agents, CORBA, and RMI middleware are used optionally in this task. This strategy however creates a trade off between speeds and robustness. Mobile agents that are capable to provide robust mechanism as they can easily be created, suspended, terminated and reboot dynamically, only deliver low speed and even vulnerable to be attacked by other agents or hosts.

The work of Payne and Markham (2001) aims to solve tamper resistance problems caused by untrusted operating system. Distributed embedded firewall developed in this effort utilizes embedded computing. The firewall is built on top of network interface card to avoid the problem of untrusted host operating system. Although this method is claimed solving the problem of untrusted host operating system and protect the intranet from malicious hosts using gateway firewall that guards network perimeter, firewall developed using this method cannot achieve dynamic security policy since the embedded method implemented on the interface card, thus it tends to have static configuration. Moreover, communication among hosts for applying security policy can also lead to high runtime overhead.

Xian *et al.* (2002) intends to improve Bellovin concept by revisiting the development of distributed microfirewall system. Domain type enforce is used in this project to enforce mandatory access control to the firewall. The purpose is to keep microfirewall being tampered by unwitting insiders or malicious host. This effort however cannot prevent network from the possibility of malicious host as gateway-firewall is removed from the implementation. Thus no security perimeter is defined to protect the internal network.

2.3.2 Adaptive Firewalls

Research on adaptive firewall is intended for understanding and dealing with the changing conditions of network security as well as the Internet threats. Although research on adaptive firewall have been running for sometimes, the author cannot determine who and when started the development of this method. However the oldest effort found in the literature for developing adaptive firewall is the work of Venkatesan and Bhattacharya (1997). Firewall developed in this effort is capable to adapt the security policy with the changes of the network threat endangering the internal system. To measure the threat, users activities is captured and analysed at any point of time, and then trusted level of each user defined priorly is adjusted. Whenever a threat is discovered from the log of user activities causes user trusted level to be decreased, otherwise it is increased. And to monitor the appearance of

intrusive action, existing intrusion detection such as expert system and statistically anomaly detection are employed. The method used in this system leads to create a security hole as it may trust the user too soon, or raise even denial of service if the employed intrusion detection system suffer from false positive. Moreover, the application of time period creates an opportunity that malicious users can be trusted. And as widely known, statistical anomaly has a pitfall that an intruder can train the system to believe on the suspicious activities, and the expert system only deals with the known intrusion. Therefore a new suspicious behaviour of user will not be detected by the system.

The development of adaptive packet filters by Reumann *et al.* (2001) notably solves the problem of security holes caused by malicious users. The approach used in the method is to allow alternative security rules to be loaded into a filter controller. The rules here are ordered by the increasing restrictiveness, which the least restrictive does not filter the incoming traffic while the most restrictive drops all packets. Users of the network are assigned specific security rule to access the outside network. This way any potentially malicious users can be identified prior to the intrusion, and therefore specific access policy can be enforced. This approach however reduces network access performance since firewall must check each incoming packet against its rule-base, and the mechanism of switching among the rule as well. The performance is even dropped when the number of rules becomes large.

Zou *et al.* (2003) afford to build an intelligent firewall based on adaptive security method. Fuzzy logic is employed to handle security classification based on the source and destination of the packet. This approach besides presenting an advantage to bring soft computing to the security world, it also causes performance decreasing due to algorithm complexity to process each flowing packet using fuzzy algorithm. To cope with this problem, Zou *et al.* simplify the analysis using a set of predetermined security levels for the source and destination of network packets. This approach results in producing shallow packet analysis.

The problem of performance overhead is also suffered by adaptive firewall of Verwoerd and Hunt (2002). Although this effort simplifies the mechanism to develop

adaptive security using packet filtering, performance decreasing due to processing intensive and complex calculations cannot be avoided, especially for computing network bandwidth. This problem indeed becomes the open issue in the research of adaptive firewall.

2.3.3 Hardware-Based Firewalls

The development of hardware-based firewall is motivated by slow processing speed of software-based system. The approach used in this research is to hold network packet processing in the hardware. The strategy to speed up the process is due to elimination of many software-based procedures. A number of efforts, such as the work of Kayssi *et al.* (2000) and Lee *et al.* (2002), afford to implement this approach by developing an FGPA (Field Programmable Gate Array) firewall. They make use the reprogramability of FPGA chips to process network packets. Different types of security rules are implemented. These works however carry a number of drawbacks leading to inefficient firewall implementation. Most of hardware-based firewalls experience difficulties in mapping security rules into the chips. Meanwhile, only limited security rules can be deployed in the chip due to hardware constraint. And as widely known, every reprogramable chip has its operational lifetime. Therefore exploiting its reprogramability will probably shorten its lifetime. Moreover, some complexities are raised by the effort to build dynamic security. Firewall developed by Kayssi *et al.* (2000) requires to power off the system for updating security rules. It causes performance decreasing since network transaction can be aborted when the system is down.

The work of Lee *et al.* (2002) to overcome the problem of security rule reconfiguration, by employing a high level programming language namely Ponder, also create a drawback in holding rule modification. This effort creates more algorithm complexity since it requires the framework to integrate the mechanism of hardware and software. And the effort of Lockwood *et al.* (2003) notably creates the extension of FGPA-based firewall to detect malicious content of Internet traffic. This work implements an algorithm to process the flowing packet in the hardware.

However this system cannot avoid hardware-based constraint, thus only limited filtering rules can be implemented.

2.3.4 Active Firewalls

The term of active firewalls was originally discovered from the result of developing security tools in two research-domains i.e. active network (Alexander *et al.*, 1999) and regular network (Eschelbeck, 2000). In the domain of active network, active firewalls actually have similar function as regular firewall devices, however they are developed to serve the mechanism of active network. It is worth to note that up to this time no serious active firewall development in the domain of active network is reported. It is due to the definition and the implementation of the platform of active network that are still progressing. The implementation of active firewall under active network such as presented by Alexander *et al.* (1999), Silva *et al.* (2001) and Hicks *et al.* (2003) are merely to prove that the proposed active networks capable to cope with networking devices.

On the contrary, researches for developing firewall in regular network have been delivering some useful methods by building active firewall. The basic notion of this method is to have active mechanism to detect and combat the threats, thus any attacks endangering the protected network can be prevented. The effort of Eschelbeck (2000), notably initiates the development of active firewall concept. In this work, active firewall is built by collaborating dynamic firewall with other security components to respond on the changing threat. This system works by actively detecting any intrusion using antivirus and intrusion detection. The active firewall proposed by Eschelbeck is also equipped with the function of vulnerability scanner to detect the existing host vulnerabilities. Although many types of security software are employed in this method, however this approach has no capabilities to measure the threat originating from the Internet. Therefore, it is still uncertain whether active firewall proposed by Eschelbeck capable to react on the changing Internet threat since it has to wait for an intrusion to update its security policy. Formalization of this method does not include any experimental result to justify the

claim nor any measurement to prove that active firewall has preventive action to secure network.

Lehtonen (2003) presented a different approach to build activate firewall. Although it is motivated by the programmability of active network and the idea of Eschelbeck firewall, the work is intended to serve the wireless network. Using a packet filtering functionality, security rules are built into a tree-like structure. Each network packet passing through the firewall is directed to a suitable leaf and branches nodes of the active firewall to enter or leaf the intranet. As clearly shown from this mechanism, this work fails to develop the mechanism for recognizing network threat. The function of firewall here is only to organize security policy, a similar method as presented by regular packet filtering firewall.

Different approach to build active firewall is presented by Hunt and Verwoerd (2003) for developing reactive firewall. Utilizing packet-filtering rules, the proposed method affords to provide safe network environment by identifying the requirements of internal network in term of bandwidth reservation and network transaction. The computation to specify the bandwidth usage is conducted based on the flowing traffic, which the source and destination of each packet are identified and compared against a predetermined security rules. Besides its complex computation that creates performance overhead, this method does not consider the condition of external network that potentially become the sources of many type of Internet threats.

2.4 Firewall Validation Standard

Due to the importance of firewall in the Internet community, some standards have been created for the purpose of validating firewall technologies. It is important to fulfil the requirements defined in these standards for creating a new firewall technique, since it warrants the usability of the developed firewall. This section aims to describe these standards, so the appropriate validation mechanism can be identified and be referred in validating the firewall developed in this research.

Basically there are two aspects influencing the behaviour of firewall devices, namely performance and security. Internet community uses a different standard for dealing with each of these aspects (Hickman *et al.*, 2003). For the purpose of validating firewall performance, some benchmarking terminologies have been presented and well described in RFC 2647 (Newman, 1999). This standard discusses the definition used in the benchmarking test such as allowed traffic, authentication, bit forwarding rate, connection, good put etc. The methodology for conducting benchmarking is presented in RFC 3511 (Hickman *et al.*, 2003). In this standard, the set up for testing firewall and the parameters to be tested are described. However as noted by Hickman *et al.*, the aspect of security is beyond the scope of this standard.

Meanwhile for the purpose of validating firewall security, the behaviour of and requirements for Internet firewalls are standardize in RFC 2979 (Freed, 2000). In this standard, two aspects, namely security and usability, are considered become the main function of firewall. However there is tradeoff between both aspects. Therefore a transparency rule is defined in this standard as the requirements for improving the security of firewall. This rule states that the introduction of firewall in the networking system must not cause unintended failures of the legitimate and standards compliant usage that would work were the firewall not present. Thus any security improvement will not prevent people from performing a useful works.

2.5 Discussion

Firewall developments are classified into four methods i.e. distributed, adaptive, hardware-based, and active firewalls. The first three methods have had a clear definition and formulation, therefore they become attractive to the researchers for improving firewall mechanism. Currently a number of reports have been presenting the achievements in enhancing firewall based on these methods. However, some issues are raised as well. Common pitfall suffered by the firewalls built using these methods are performance decreasing due to the increased of algorithm complexity and the processing load. Besides, these methods also suffer from specific drawbacks, such as the possible existence of malicious host in the distributed firewall

and hardware constraint of hardware-based firewall. Those drawbacks, of course, become the barrier for the acceptance of firewall development.

And as mentioned earlier in Session 2.2, the factor that has more influence in network security is the appearance of Internet threat in which some surveys (CERT, 2002; CERT, 2003; CERT, 2004; Whitman, 2003; Anagnostakis, 2003; Kientzle and Elder, 2003) conclude that malicious code has the bigger portion for causing this threat. Therefore a mechanism to keep the bad stuff of Internet from coming into the protected intranet is required. The method based on distributed, adaptive, and hardware-based firewalls apparently show lack performance to stop the action of malicious code. However, adaptive firewall that is combined with intrusion detection or antivirus tools may reduce this threat. This strategy is significantly influenced by the performance of the employed security tools, such as intrusion detection system that may suffer from false positive and false negative, or antivirus software that can be compromised due to late updating virus database. Moreover communication method between firewall and other security tools potentially create performance decreasing that causes security holes in the network. By considering the issues above, active firewall concept introduced by Eschelbeck (2000) has the promising mechanism to provide security for the intranet. The main idea behind this concept is to equip firewall with active mechanism to detect, recognize and take an action against the Internet threat. This method however suffers from similar problem as adaptive firewall combined with other security tools, since the mechanism applied in this effort relies on the collaboration with other methods. Meanwhile, other works such as presented by Lehtonen (2003) and Hunt and Verwoerd (2003) fail to accommodate the requirements to deal with the changing pattern of network threats. Thus improvement for developing active firewall is still required.

2.6 Summary

The development of firewall technology that consists of distributed, adaptive, hardware-based, and active firewall has been presented. The purpose is to discover the direction of the research in the field of network firewall and to highlight the issue produced from these efforts. Referring to the survey results that produce malicious code as the top threat of Internet (CERT, 2002; CERT, 2003; CERT, 2004), only the adaptive and active firewalls that are capable to deter this threat. However if we compare adaptive and active firewall, it shows that adaptive method tends to have passive mechanism that wait for the incidents to happen before changing the security rules, thus it might be too late to react. Active firewall has more promising approach in protecting intranet since this method has active mechanism to search any indication of security threats.

CHAPTER 3

ACTIVE FIREWALL MODEL

3.1 Introduction

Development of active firewall has been started by Eschelbeck (2000) to produce active security. The concept of active firewall notably deliver new paradigm in protecting organization internal network. Idea of this effort is to create firewall capable to respond actively to the raising and changing Internet threats. Although implementations of this method still carry some drawbacks and weaknesses as described in Chapter 2, the effort opens a new view to counter Internet threat. This chapter aims to continue the work on developing active firewall concept by identifying the requirements that shall be fulfilled in establishing secure network environment. The work includes the development of Internet access model that consist of the models of intranet users and external parties, the formulation of network security strategy, and the implementation of this strategy into a model of active firewall.

3.2 Definition of Active Firewall

Currently definition of active firewall has not been settled among the researchers. Yet few efforts have started proposing the mechanism of active firewall by describing how this device should behave. Eschelbeck (2000) defines active

firewall as active guards working in concert with other security components in the network to actively respond to the changing threats. Collaborative security approach is delivered from this definition. Meanwhile Hunt and Verwoerd (2003) tends to have adaptive method by defining active firewall as device capable to change or adapt its rules in the face of adverse situation. Both efforts created different approaches in implementing active mechanism into the firewall, however they agree that active firewall shall be able to respond on the adverse condition caused by the changing threat. It means active firewall shall develop intelligent mechanism to understand the condition of the surrounding network both internal and external, and then to detect and to recognize any possible threat that can be raised from unfavourable network conditions, and finally to change its configuration in order to counter the threats.

Based on this paradigm, here active firewall is defined as firewall aware of the conditions of its surrounding network and capable to identify and to develop security requirement for guarding the protected network. From this point, requirement for developing active firewall is described by identifying possible threat caused by accessing Internet and modelling the root cause of this condition, and then developing security strategy for providing intranet protection. The developed active firewall will follow this strategy in the implementation stage.

3.3 Modelling Internet Access

Formulating an Internet access model is a useful approach towards reviewing the interaction between the protected internal users and the Internet. The model provides a schematic point of view in developing security strategy for intranet protection. Using set theory, each possible condition raised from accessing Internet is sought and examined. This approach delivers three separate models i.e. intranet users model, external parties model and Internet access model in which the last model become the product of integrating both former ones. Description of each model follows.

3.3.1 Intranet Users Model

Model of intranet users is implemented by developing the interaction between the possible safety conditions experienced by each user. Let *U* becomes a domain of intranet users that consist of two safety regions namely protected and unprotected. These regions correspond to the protection and unprotection index denoted by *p* and *up* respectively. Assuming that an intranet user is an entity, which each user $u \in U$ is defined as a tuple $< p, up >$, hence

$$p + up = 1 \tag{3.1}$$

Equation (3.1) must be fulfilled for each $u =< p, up >$. Illustration of this model is depicted in Figure 3.1.

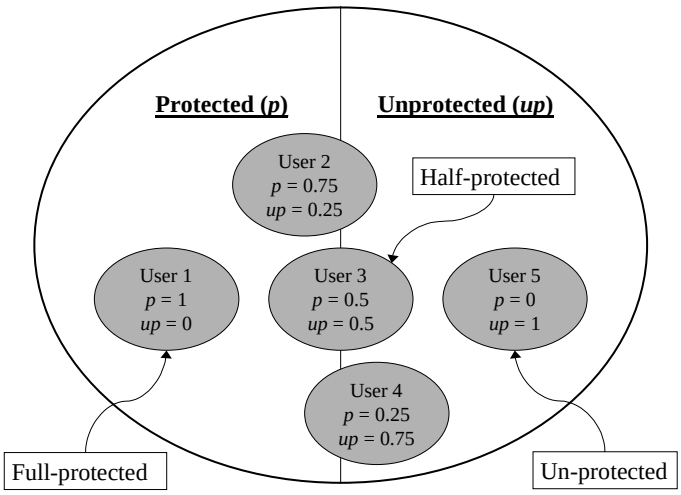


Figure 3.1: Intranet users model

To measure the safety factor of an intranet, the total protection index and unprotection index are computed as follow.

$$P = \sum_i p_{u_i} \tag{3.2}$$

$$UP = \sum_i up_{u_i} \tag{3.3}$$

Thus safety factor is obtained by correlating total protection and unprotection index as follow.

$$S = \frac{P}{UP + P} \tag{3.4}$$

The safety factor here is used to measure the security of an intranet for providing a safe environment for its users. Here the value of safety factor ranges from $S = 0$ means the intranet provide lowest protection to the users, to $S = 1$ means users are fully protected by the intranet. To determine the value of protection and unprotection index, the security methods proposed by Eschelbeck (2000) in studying the intranet protection are used as the input parameters, together with the survey result of CERT (2004) in developing effective network security. The former effort proposes the collaboration of firewall, intrusion detection, vulnerability assessment and antivirus to protect an intranet. Analysis of integrating these methods by Davies (2000) show the added advantages can be delivered i.e. a proactive approach is produced by vulnerability assessment, and a reactive approach is delivered by intrusion detection. Meanwhile CERT (2004) shows the most effective security methods as follows firewall, encryption in transmitting data, encryption in storage, manual patch management, regular security audits, and the recording of user activities. Based on those aspects, protection of intranet user is computed by measuring the availability of the following factors:

(i) Firewall
(ii) Intrusion detection
(iii) Vulnerability assessment
(iv) Antivirus
(v) Encryption in transmitting data
(vi) Encryption in storage
(vii) Manual patching and update management
(viii) Regular security audits
(ix) User activities recording

To simplify the model, it is assumed that the factors above contribute to the same portion of user protection, with total protection derived from the integration of those factors is 100%. Therefore the absence of a single security method for protecting the individual user in the intranet causes the reduction of protection index by factor

$$x = \frac{100\%}{m} \tag{3.5}$$

with m denotes the number of identified security methods used to protect intranet users as listed above. Since $m = 9$, then $x = 11.11\%$. In other words, the unprotected index will increase by factor 11.11% for the absence of a single security method. Illustration of this approach is depicted in Figure 3.2. It is worth to note that the effort for developing the user model as explained in this subsection enables the empirical measurement of intranet security of the organization network, which the protection index and security hole can quantitatively be calculated using Equation (3.2) to (3.5).

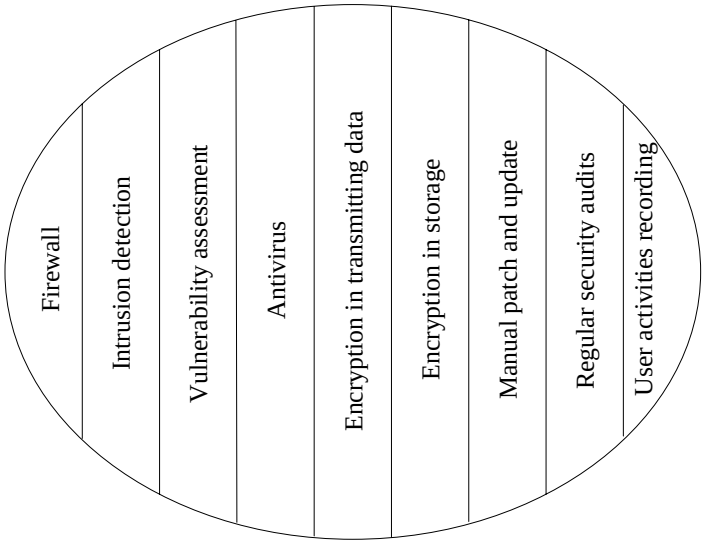


Figure 3.2: Illustration for computing protection and unprotection index

3.3.2 External Parties Model

External parties are modelled using the similar way as intranet users. Instead of using protection and unprotection index, this model utilizes trusted and untrusted index denoted by *t* and *ut* respectively, in which the following relation

$$t + ut = 1 \tag{3.6}$$

must be fulfilled for each external party by assuming that a single external party is an entity. And let *O* become a domain of external parties, an external party $o \in O$ is defined as a tuple $< t, ut >$ satisfying Equation (3.6). Illustration of this model is given in Figure 3.3.

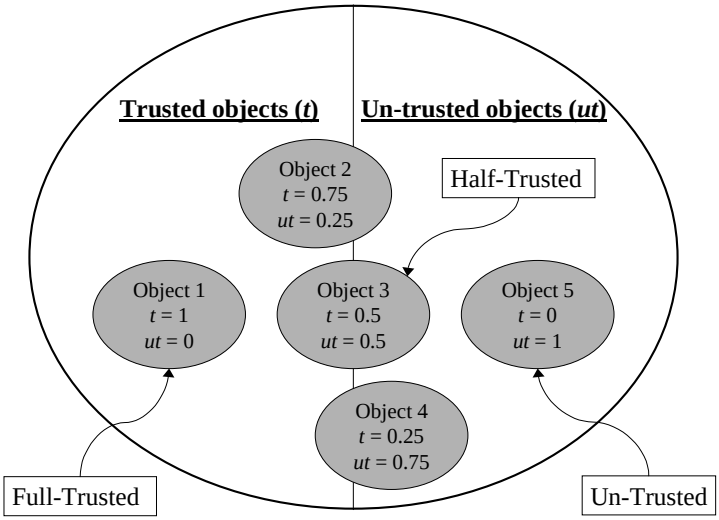


Figure 3.3: Model of external parties

To compute the total trusted and untrusted index, the following calculations are held

$$T = \sum_i t_{0_i} \tag{3.7}$$

$$UT = \sum_i ut_{o_i} \tag{3.8}$$

Thus the safety factor of the domain *O* is computed as follows

$$S = \frac{T}{UT + T} \tag{3.9}$$

To determine the trusted and untrusted index of an individual external party, categories of Internet threat defined by Seo *et al.* (2004) are employed in this model. It is due to complete presentation of the sources of web attack by Seo *et al.* in which explanation of those attack can also be found partially from the works of Arbaugh (2003), Kienzle and Elder (2003), and Hernandez (2001). According to Seo *et al.* (2004), threats originated from external parties are classified into twelve items i.e. code scanning, cookie poisoning, hidden manipulation, forceful site browsing, third-party misconfiguration, known vulnerabilities, buffer overflow, debug option and back doors, parameter tampering, stealth commanding, cross site scripting, and application denial of service. Thus the trusted index of an external party is determined by measuring the presence of these threats. Assuming that each threat contribute to the same portion of trusted index, then the absence of a threat will increase the trusted index by factor

$$y = \frac{100\%}{n} \tag{3.10}$$

with *n* denotes the number of identified threats. Referring to Seo *et al.* (2004), it produces $n = 12$, and $y = 8.33\%$. It means the presence of each threat above reduces trusted index by 8.33 %. Illustration of this approach is shown in Figure 3.4. After computing the trusted and untrusted index, each external party can be mapped into the external parties model as given in Figure 3.3 to define the safety factor of permitted Internet access using Equation (3.7) to (3.9).

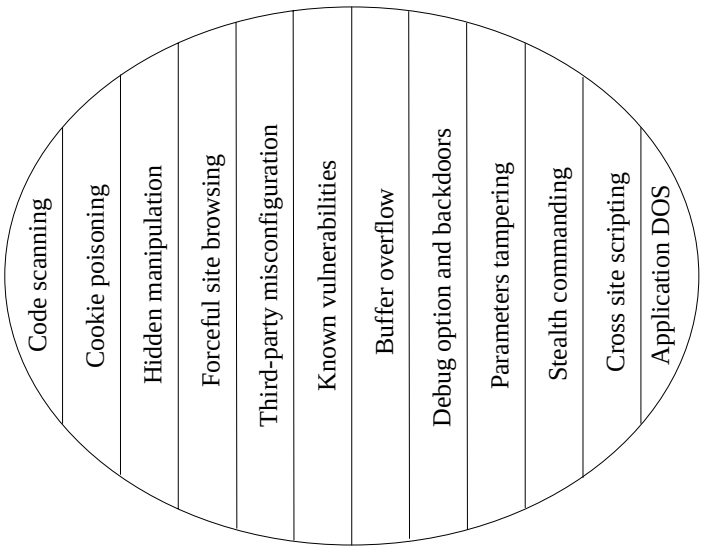


Figure 3.4: The category of Internet threats to compute the trusted index

3.3.3 Internet Access Model

Model of Internet access is built by intersecting the model of intranet users against the model of external parties. It is illustrated in Figure 3.5. The new model creates some security regions as the product of the interaction between the protected and unprotected index of intranet users model with the trusted and untrusted index of the model of external parties. The new security regions correspond to the safety condition for accessing intranet. Formulation of each region follows.

Let $SC = \{safe, controlled, threat\}$ become the domain of security condition, the interaction between the intranet users *U* and the external parties *O* can be expressed as

$$SC_{U \cap O} = \begin{cases} safe & \leftarrow U_p \cap O_t \\ controlled & \leftarrow U_p \cap O_{ut} \vee U_{up} \cap O_t \\ threat & \leftarrow U_{up} \cap O_{ut} \end{cases} \quad (3.11)$$

Equation (3.11) states that the threat condition is produced from the interaction between unprotected user and untrusted external parties. Therefore the efforts to minimize the interaction between unprotected users and untrusted external parties are required in order to develop more secure environment for accessing Internet. These efforts are formulated into the security strategies described in the next section.

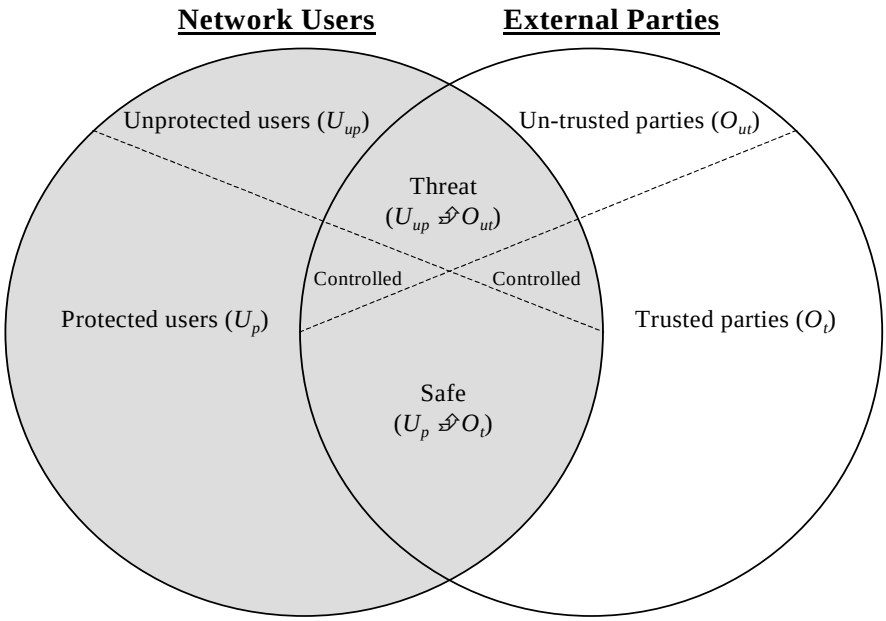


Figure 3.5: Model of Internet access

3.4 Threat Reduction Strategies

Knowledge obtained from developing Internet access model clearly shows the root causing threat condition i.e. the interaction between unprotected users and untrusted external parties. Hence the strategies to minimize, or if possible to eliminate, the threat condition will significantly deliver more secure Internet access. Based on Equation (3.11), the following approaches can be used to reduce the threats:

- Minimizing U_{up} (unprotected users)

  It is implemented by fulfilling all factors to produce maximum protection index as described in Section 3.3.1. This approach intends to bring all security methods to protect intranet users, thus safe or controlled condition can be achieved. However the cost will be too expensive to execute this strategy since many security tools are required. This strategy may introduce performance overhead as well. Other approach to run this strategy is to equip all intranet users with the distributed detectors such as antivirus or intrusion detection system, hence any condition leading to the threats can be detected and passed to firewall. This way firewall can take an action to secure the internal users e.g. to drop the communication line of the victim machines. The last approach is considered more applicable rather than installing all available security tools on each intranet machines that causes performance decreasing.

- Minimizing O_{ut} (untrusted external parties)

  Similar to the first approach, this strategy intends to fulfil all factors in order to provide maximum trusted index. It is conducted by restricting the access to the external parties that contain the threats as described in Section 3.3.2. Since various Internet contents can be found at present, this approach potentially reduces the flexibility of Internet access. If this problem cannot be avoided, accessing Internet will not be as easy as before executing this strategy. Hence a correct mechanism to reduce untrusted external parties without burdening user flexibility too much is desired in implementing this strategy.

- Minimizing $U_{up} \cap O_{ut}$ (the interaction between unprotected users and untrusted external parties)

  This approach is implemented by regulating the interaction between unprotected users and untrusted external parties. Compared to both previous strategies, this approach seems delivering a promising mechanism to provide

better network protection since it is potentially capable to avoid performance decreasing due to minimizing U_{up} and the problem of lack flexibility caused by minimizing O_{ut}. The advantages provided by this strategy are twofold. First, it is not necessary to install many security tools in the intranet or internal machines, although bigger protection index is desired. And second, it does not require filtering all network traffic to prevent various Internet threat. However content analysis on the flowing network traffic is still necessary for the purpose of recognizing the security conditions in accessing the external parties. Hence a soft computing or artificial intelligence can be employed to handle this task. In implementing this strategy, the developed content analysis algorithm is used for supporting the access control method for regulating the interaction between internal users and external parties.

3.5 Active Firewall Model

This section present the development of active firewall model in order to host the implementation of the developed threat reduction strategies defined in Section 3.4. Let active firewall control a set of reconfigureable canals $C = \{c_1, c_2, \Lambda\ \Lambda\ , c_k\}$ connecting an intranet that consists of *k* internal machines to the Internet, and let n denotes the integer number and c_n becomes a set of canals corresponding to the particular connection of an internal machine to an external party, in which the connection status is given by $c_n = \{1,0\}$ with $c_n = 1$ represent a connected line while $c_n = 0$ means a close condition. To manage the Internet connection, active firewall enables and disables c_n as visualized in Figure 3.6. This way, using particular method to compute and detect the Internet threats, the strategies developed in Section 3.4 can be implemented for managing the Internet connection.

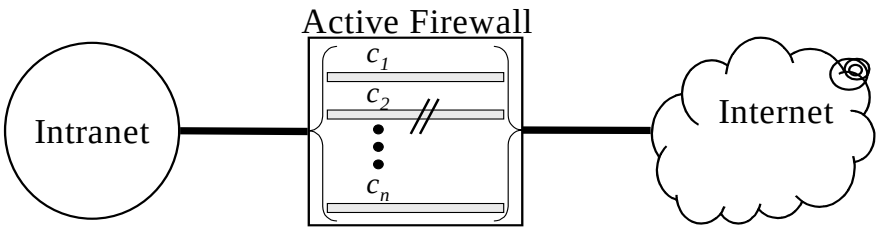


Figure 3.6: Mechanism of active firewall to control the canals

In this research, active firewall is implemented by developing the technique to handle initialisation and runtime process. Here initialisation refers to the mechanism to start the firewall up from its off condition, while the runtime refers to the operation of firewall at all time after the firewall is started up to a time before it is shutdown. The active firewall process is depicted in Figure 3.7.

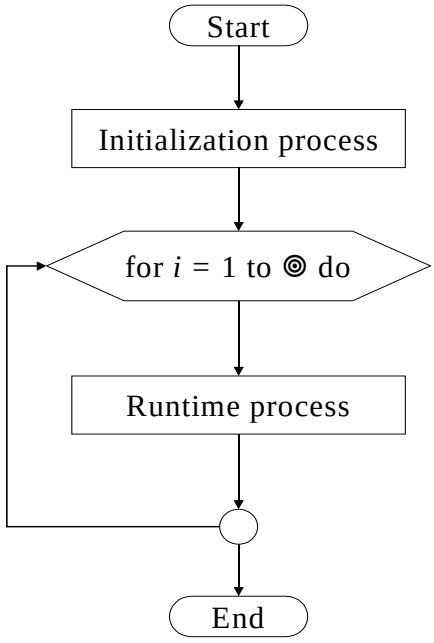


Figure 3.7: Active firewall process

3.6 Summary

This chapter discusses the concept and proposes a model of active firewall. The concept of active firewall is defined as the firewall aware of the security condition of its surrounding network, actively detecting the appearance of security threats, and capable to take an action to stop the threats. Meanwhile the model of active firewall is designed using a concept of canalisation of the connections of intranet to Internet or vice versa. This model is used as the platform for implementing the security strategies to reduce Internet threats. In this chapter, the development of security strategies to combat Internet threats is presented as well. This effort is held by developing a generic model of intranet users that consist of protected and unprotected index, and a generic model of external parties that consists of trusted and untrusted index. By intersecting both models, a model of Internet access is built, in which three security conditions are obtained namely safe, controlled and threats. The security strategy defined in this chapter deals with the last condition of Internet access i.e. reducing the threats condition. Thus the implementation of security strategy in active firewall will also aim to reduce the threat condition, in which it is achieved by running one of this activities i.e. minimizing the unprotected internal users, minimizing the untrusted external parties, or minimizing the interaction between unprotected users with untrusted external parties.

CHAPTER 3

ACTIVE FIREWALL MODEL

3.1 Introduction

Development of active firewall has been started by Eschelbeck (2000) to produce active security. The concept of active firewall notably deliver new paradigm in protecting organization internal network. Idea of this effort is to create firewall capable to respond actively to the raising and changing Internet threats. Although implementations of this method still carry some drawbacks and weaknesses as described in Chapter 2, the effort opens a new view to counter Internet threat. This chapter aims to continue the work on developing active firewall concept by identifying the requirements that shall be fulfilled in establishing secure network environment. The work includes the development of Internet access model that consist of the models of intranet users and external parties, the formulation of network security strategy, and the implementation of this strategy into a model of active firewall.

3.2 Definition of Active Firewall

Currently definition of active firewall has not been settled among the researchers. Yet few efforts have started proposing the mechanism of active firewall by describing how this device should behave. Eschelbeck (2000) defines active

firewall as active guards working in concert with other security components in the network to actively respond to the changing threats. Collaborative security approach is delivered from this definition. Meanwhile Hunt and Verwoerd (2003) tends to have adaptive method by defining active firewall as device capable to change or adapt its rules in the face of adverse situation. Both efforts created different approaches in implementing active mechanism into the firewall, however they agree that active firewall shall be able to respond on the adverse condition caused by the changing threat. It means active firewall shall develop intelligent mechanism to understand the condition of the surrounding network both internal and external, and then to detect and to recognize any possible threat that can be raised from unfavourable network conditions, and finally to change its configuration in order to counter the threats.

Based on this paradigm, here active firewall is defined as firewall aware of the conditions of its surrounding network and capable to identify and to develop security requirement for guarding the protected network. From this point, requirement for developing active firewall is described by identifying possible threat caused by accessing Internet and modelling the root cause of this condition, and then developing security strategy for providing intranet protection. The developed active firewall will follow this strategy in the implementation stage.

3.3 Modelling Internet Access

Formulating an Internet access model is a useful approach towards reviewing the interaction between the protected internal users and the Internet. The model provides a schematic point of view in developing security strategy for intranet protection. Using set theory, each possible condition raised from accessing Internet is sought and examined. This approach delivers three separate models i.e. intranet users model, external parties model and Internet access model in which the last model become the product of integrating both former ones. Description of each model follows.

3.3.1 Intranet Users Model

Model of intranet users is implemented by developing the interaction between the possible safety conditions experienced by each user. Let *U* becomes a domain of intranet users that consist of two safety regions namely protected and unprotected. These regions correspond to the protection and unprotection index denoted by *p* and *up* respectively. Assuming that an intranet user is an entity, which each user $u \in U$ is defined as a tuple $< p, up >$, hence

$$p + up = 1 \tag{3.1}$$

Equation (3.1) must be fulfilled for each $u =< p, up >$. Illustration of this model is depicted in Figure 3.1.

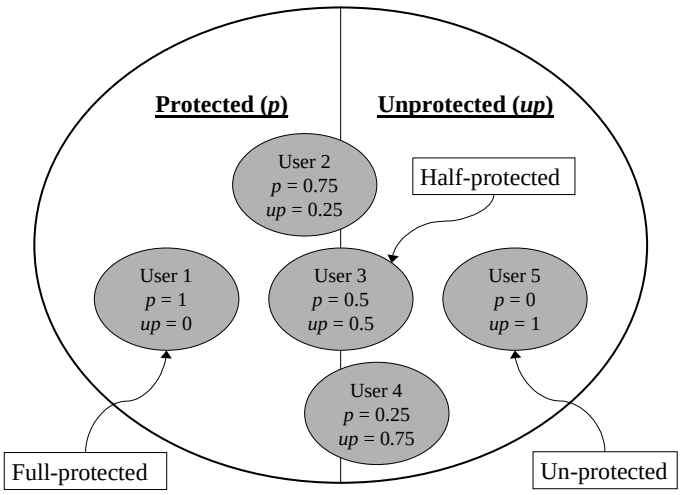


Figure 3.1: Intranet users model

To measure the safety factor of an intranet, the total protection index and unprotection index are computed as follow.

$$P = \sum_i p_{u_i} \tag{3.2}$$

$$UP = \sum_i up_{u_i} \tag{3.3}$$

Thus safety factor is obtained by correlating total protection and unprotection index as follow.

$$S = \frac{P}{UP + P} \tag{3.4}$$

The safety factor here is used to measure the security of an intranet for providing a safe environment for its users. Here the value of safety factor ranges from $S = 0$ means the intranet provide lowest protection to the users, to $S = 1$ means users are fully protected by the intranet. To determine the value of protection and unprotection index, the security methods proposed by Eschelbeck (2000) in studying the intranet protection are used as the input parameters, together with the survey result of CERT (2004) in developing effective network security. The former effort proposes the collaboration of firewall, intrusion detection, vulnerability assessment and antivirus to protect an intranet. Analysis of integrating these methods by Davies (2000) show the added advantages can be delivered i.e. a proactive approach is produced by vulnerability assessment, and a reactive approach is delivered by intrusion detection. Meanwhile CERT (2004) shows the most effective security methods as follows firewall, encryption in transmitting data, encryption in storage, manual patch management, regular security audits, and the recording of user activities. Based on those aspects, protection of intranet user is computed by measuring the availability of the following factors:

(i) Firewall
(ii) Intrusion detection
(iii) Vulnerability assessment
(iv) Antivirus
(v) Encryption in transmitting data
(vi) Encryption in storage
(vii) Manual patching and update management
(viii) Regular security audits
(ix) User activities recording

To simplify the model, it is assumed that the factors above contribute to the same portion of user protection, with total protection derived from the integration of those factors is 100%. Therefore the absence of a single security method for protecting the individual user in the intranet causes the reduction of protection index by factor

$$x = \frac{100\%}{m} \tag{3.5}$$

with m denotes the number of identified security methods used to protect intranet users as listed above. Since $m = 9$, then $x = 11.11\%$. In other words, the unprotected index will increase by factor 11.11% for the absence of a single security method. Illustration of this approach is depicted in Figure 3.2. It is worth to note that the effort for developing the user model as explained in this subsection enables the empirical measurement of intranet security of the organization network, which the protection index and security hole can quantitatively be calculated using Equation (3.2) to (3.5).

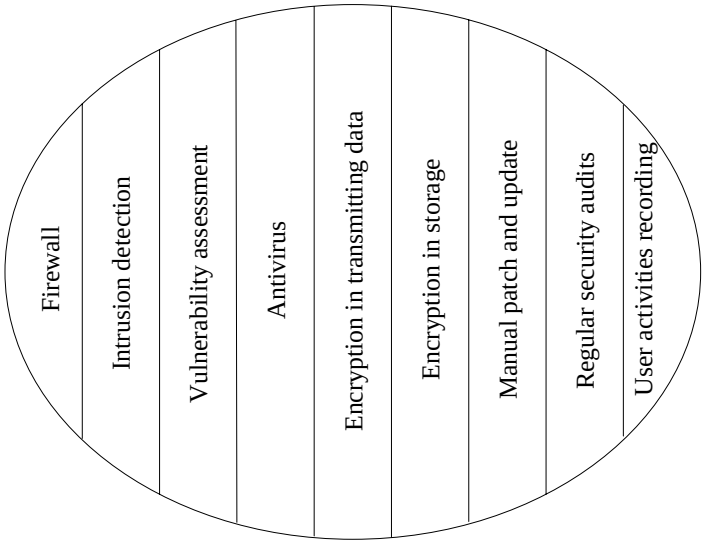


Figure 3.2: Illustration for computing protection and unprotection index

3.3.2 External Parties Model

External parties are modelled using the similar way as intranet users. Instead of using protection and unprotection index, this model utilizes trusted and untrusted index denoted by *t* and *ut* respectively, in which the following relation

$$t + ut = 1 \tag{3.6}$$

must be fulfilled for each external party by assuming that a single external party is an entity. And let *O* become a domain of external parties, an external party $o \in O$ is defined as a tuple $< t, ut >$ satisfying Equation (3.6). Illustration of this model is given in Figure 3.3.

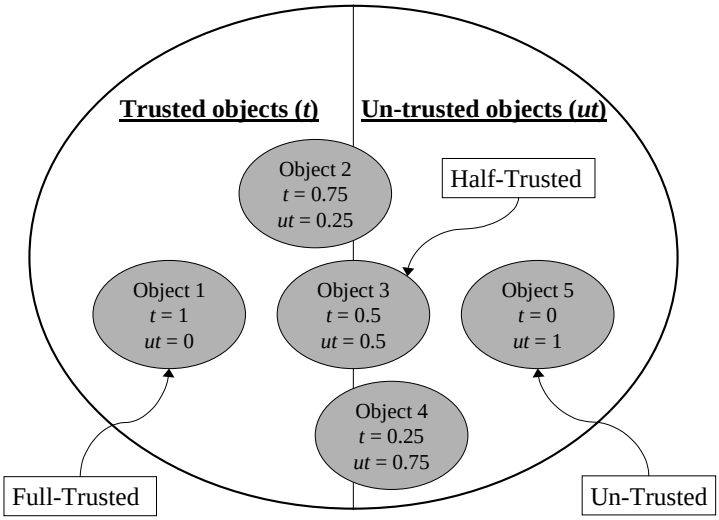


Figure 3.3: Model of external parties

To compute the total trusted and untrusted index, the following calculations are held

$$T = \sum_i t_{0_i} \tag{3.7}$$

$$UT = \sum_i ut_{o_i} \tag{3.8}$$

Thus the safety factor of the domain *O* is computed as follows

$$S = \frac{T}{UT + T} \tag{3.9}$$

To determine the trusted and untrusted index of an individual external party, categories of Internet threat defined by Seo *et al.* (2004) are employed in this model. It is due to complete presentation of the sources of web attack by Seo *et al.* in which explanation of those attack can also be found partially from the works of Arbaugh (2003), Kienzle and Elder (2003), and Hernandez (2001). According to Seo *et al.* (2004), threats originated from external parties are classified into twelve items i.e. code scanning, cookie poisoning, hidden manipulation, forceful site browsing, third-party misconfiguration, known vulnerabilities, buffer overflow, debug option and back doors, parameter tampering, stealth commanding, cross site scripting, and application denial of service. Thus the trusted index of an external party is determined by measuring the presence of these threats. Assuming that each threat contribute to the same portion of trusted index, then the absence of a threat will increase the trusted index by factor

$$y = \frac{100\%}{n} \tag{3.10}$$

with *n* denotes the number of identified threats. Referring to Seo *et al.* (2004), it produces $n = 12$, and $y = 8.33\%$. It means the presence of each threat above reduces trusted index by 8.33 %. Illustration of this approach is shown in Figure 3.4. After computing the trusted and untrusted index, each external party can be mapped into the external parties model as given in Figure 3.3 to define the safety factor of permitted Internet access using Equation (3.7) to (3.9).

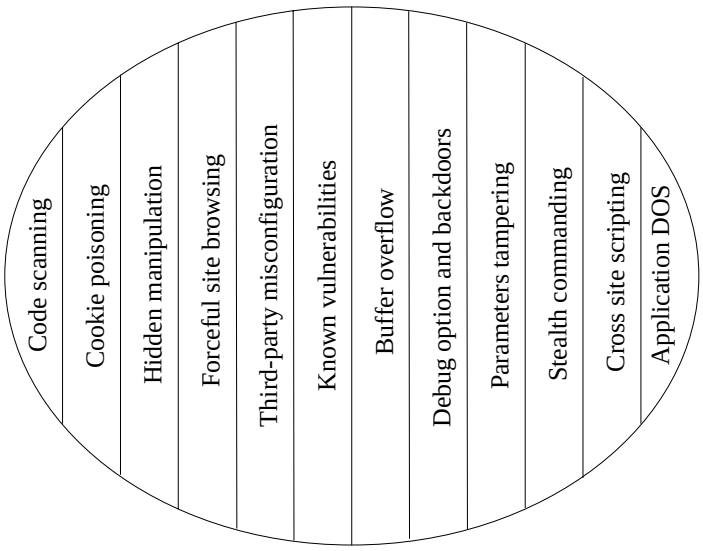


Figure 3.4: The category of Internet threats to compute the trusted index

3.3.3 Internet Access Model

Model of Internet access is built by intersecting the model of intranet users against the model of external parties. It is illustrated in Figure 3.5. The new model creates some security regions as the product of the interaction between the protected and unprotected index of intranet users model with the trusted and untrusted index of the model of external parties. The new security regions correspond to the safety condition for accessing intranet. Formulation of each region follows.

Let $SC = \{safe, controlled, threat\}$ become the domain of security condition, the interaction between the intranet users *U* and the external parties *O* can be expressed as

$$SC_{U \cap O} = \begin{cases} safe & \leftarrow U_p \cap O_t \\ controlled & \leftarrow U_p \cap O_{ut} \vee U_{up} \cap O_t \\ threat & \leftarrow U_{up} \cap O_{ut} \end{cases} \quad (3.11)$$

Equation (3.11) states that the threat condition is produced from the interaction between unprotected user and untrusted external parties. Therefore the efforts to minimize the interaction between unprotected users and untrusted external parties are required in order to develop more secure environment for accessing Internet. These efforts are formulated into the security strategies described in the next section.

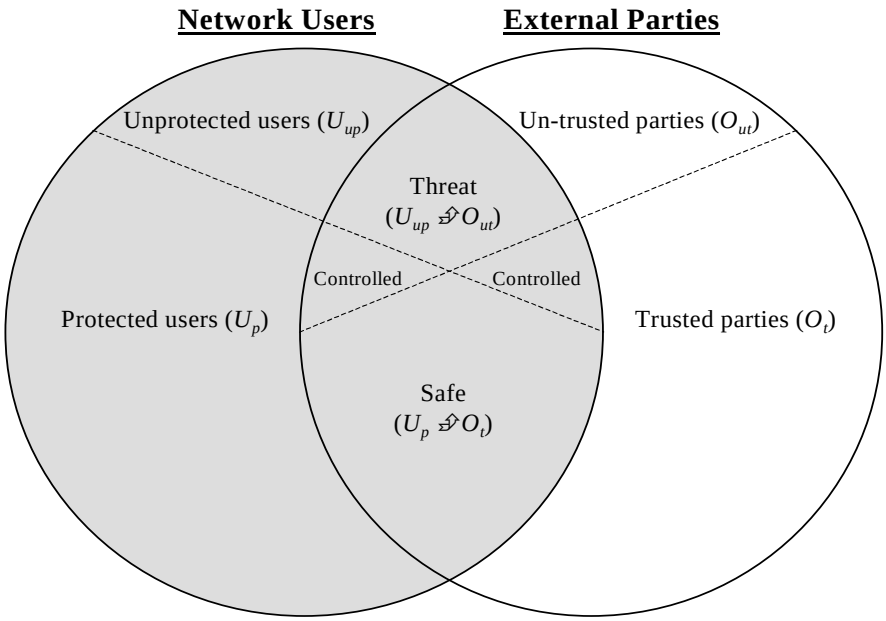


Figure 3.5: Model of Internet access

3.4 Threat Reduction Strategies

Knowledge obtained from developing Internet access model clearly shows the root causing threat condition i.e. the interaction between unprotected users and untrusted external parties. Hence the strategies to minimize, or if possible to eliminate, the threat condition will significantly deliver more secure Internet access. Based on Equation (3.11), the following approaches can be used to reduce the threats:

- Minimizing U_{up} (unprotected users)

  It is implemented by fulfilling all factors to produce maximum protection index as described in Section 3.3.1. This approach intends to bring all security methods to protect intranet users, thus safe or controlled condition can be achieved. However the cost will be too expensive to execute this strategy since many security tools are required. This strategy may introduce performance overhead as well. Other approach to run this strategy is to equip all intranet users with the distributed detectors such as antivirus or intrusion detection system, hence any condition leading to the threats can be detected and passed to firewall. This way firewall can take an action to secure the internal users e.g. to drop the communication line of the victim machines. The last approach is considered more applicable rather than installing all available security tools on each intranet machines that causes performance decreasing.

- Minimizing O_{ut} (untrusted external parties)

  Similar to the first approach, this strategy intends to fulfil all factors in order to provide maximum trusted index. It is conducted by restricting the access to the external parties that contain the threats as described in Section 3.3.2. Since various Internet contents can be found at present, this approach potentially reduces the flexibility of Internet access. If this problem cannot be avoided, accessing Internet will not be as easy as before executing this strategy. Hence a correct mechanism to reduce untrusted external parties without burdening user flexibility too much is desired in implementing this strategy.

- Minimizing $U_{up} \cap O_{ut}$ (the interaction between unprotected users and untrusted external parties)

  This approach is implemented by regulating the interaction between unprotected users and untrusted external parties. Compared to both previous strategies, this approach seems delivering a promising mechanism to provide

better network protection since it is potentially capable to avoid performance decreasing due to minimizing U_{up} and the problem of lack flexibility caused by minimizing O_{ut}. The advantages provided by this strategy are twofold. First, it is not necessary to install many security tools in the intranet or internal machines, although bigger protection index is desired. And second, it does not require filtering all network traffic to prevent various Internet threat. However content analysis on the flowing network traffic is still necessary for the purpose of recognizing the security conditions in accessing the external parties. Hence a soft computing or artificial intelligence can be employed to handle this task. In implementing this strategy, the developed content analysis algorithm is used for supporting the access control method for regulating the interaction between internal users and external parties.

3.5 Active Firewall Model

This section present the development of active firewall model in order to host the implementation of the developed threat reduction strategies defined in Section 3.4. Let active firewall control a set of reconfigureable canals $C = \{c_1, c_2, \Lambda\ \Lambda\ , c_k\}$ connecting an intranet that consists of *k* internal machines to the Internet, and let n denotes the integer number and c_n becomes a set of canals corresponding to the particular connection of an internal machine to an external party, in which the connection status is given by $c_n = \{1,0\}$ with $c_n = 1$ represent a connected line while $c_n = 0$ means a close condition. To manage the Internet connection, active firewall enables and disables c_n as visualized in Figure 3.6. This way, using particular method to compute and detect the Internet threats, the strategies developed in Section 3.4 can be implemented for managing the Internet connection.

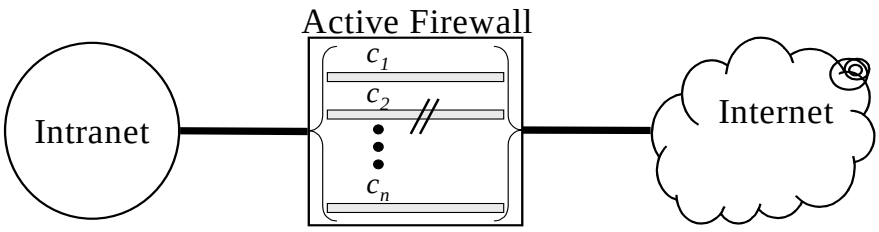


Figure 3.6: Mechanism of active firewall to control the canals

In this research, active firewall is implemented by developing the technique to handle initialisation and runtime process. Here initialisation refers to the mechanism to start the firewall up from its off condition, while the runtime refers to the operation of firewall at all time after the firewall is started up to a time before it is shutdown. The active firewall process is depicted in Figure 3.7.

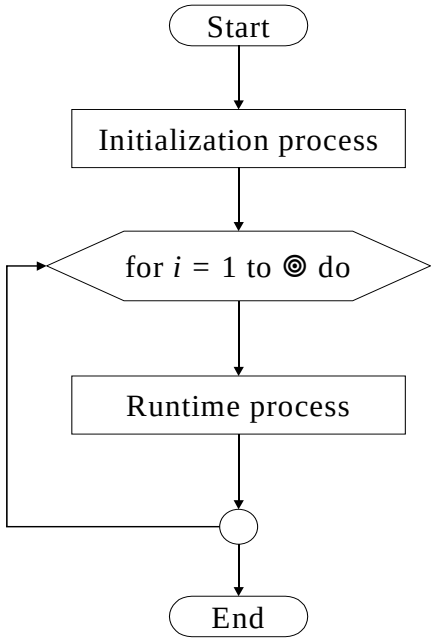


Figure 3.7: Active firewall process

3.6 Summary

This chapter discusses the concept and proposes a model of active firewall. The concept of active firewall is defined as the firewall aware of the security condition of its surrounding network, actively detecting the appearance of security threats, and capable to take an action to stop the threats. Meanwhile the model of active firewall is designed using a concept of canalisation of the connections of intranet to Internet or vice versa. This model is used as the platform for implementing the security strategies to reduce Internet threats. In this chapter, the development of security strategies to combat Internet threats is presented as well. This effort is held by developing a generic model of intranet users that consist of protected and unprotected index, and a generic model of external parties that consists of trusted and untrusted index. By intersecting both models, a model of Internet access is built, in which three security conditions are obtained namely safe, controlled and threats. The security strategy defined in this chapter deals with the last condition of Internet access i.e. reducing the threats condition. Thus the implementation of security strategy in active firewall will also aim to reduce the threat condition, in which it is achieved by running one of this activities i.e. minimizing the unprotected internal users, minimizing the untrusted external parties, or minimizing the interaction between unprotected users with untrusted external parties.

CHAPTER 4

INITIALISATION PROCESS

4.1 Introduction

Developing initialisation strategy is essential to set firewall into the suitable condition for runtime process. According to Ren *et al.* (2004), initialisation refers to the procedure of starting up and bringing the system to a point on its operation. To develop this procedure condition before and after the firewall turned up are identified.

4.2 Closed-Condition Approach

This approach is based on the experience of Ranum and Avolio (1994), which suggest to turning off system services at minimum to have more secure system. Thus in this effort, it intends to minimize the availability of services in the beginning of runtime operation. Formalization of close-condition approach is described as follow.

Let firewall controls a set of reconfigureable canals $C = \{c_1, c_2, \text{K}, c_n, \text{K}, c_k\}$ in which each canal c_n corresponds to the individual intranet machine n. Starting up the firewall means to bring the firewall from the off condition at time $t = t_0$ to the initial condition at time $t = t_i$, where firewall is ready to hold its runtime process.

Hence close-condition approach introduced here forces the firewall to close all available canal at time $t = t_i$, thus it can be formalized as follows

$$\forall c_n(t_i) = 0 \qquad (4.1)$$

A timeline visualization of this approach is given in Figure 4.1.

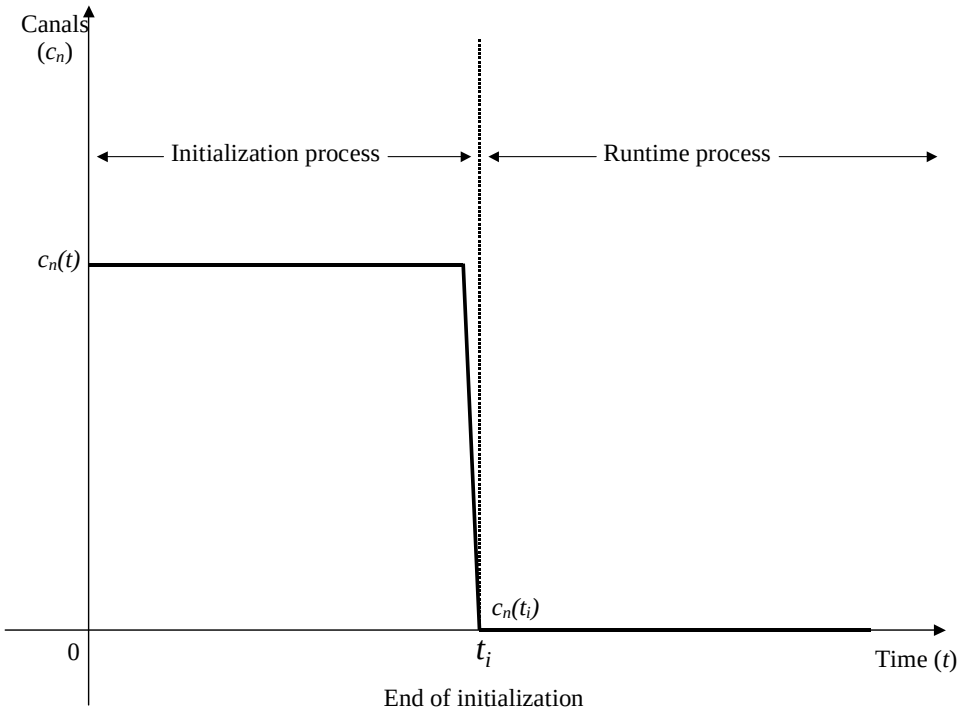


Figure 4.1: Timeline visualization of close-condition approach

It can be noted that close-condition approach enables firewall to always authenticate all the requested Internet access, thus strict Internet transactions are produced at runtime. Here traffic analyser and canal modification must be established to support the function of firewall to organize external network transaction, otherwise no internal machine can reach the Internet since no opened canals are available at runtime. Implementation of this approach is conducted by developing the following script:

```
# To have default DROP policy
   iptables -P INPUT ACCEPT
   iptables -P OUTPUT ACCEPT
   iptables -P FORWARD ACCEPT

# Delete and flush all rules
   iptables --flush
   iptables --table nat --flush
   iptables --delete-chain
   iptables --table nat --delete-chain

# Set up IP Forwarding and Masquerading
   iptables -T nat -A POSTROUTING --out-interface eth0 -j MASQUERADE
   iptables -A FORWARD --in-interface eth1 -j DROP
   echo 1 > /proc/sys/net/ipv4/ip_forward
```

4.3 Open-Condition Approach

Just like the mechanism of the close-condition approach, this method similarly projects the firewall into its edge-condition. Rather than closing all available canals as held by close-condition, here canals are opened after passing the initialisation stage. It can be formalized as follows

$$\forall c_n(t_i) = 1 \qquad (4.2)$$

Visualization of this approach is given in Figure 4.2. This method is implemented by executing the following script:

```
# To have default DROP policy
   iptables -P INPUT ACCEPT
   iptables -P OUTPUT ACCEPT
   iptables -P FORWARD ACCEPT

# Delete and flush.
   iptables --flush
   iptables --table nat --flush
   iptables --delete-chain
   iptables --table nat --delete-chain

# Set up IP Forwarding and Masquerading
   iptables -T nat -A POSTROUTING --out-interface eth0 -j MASQUERADE
   iptables -A FORWARD --in-interface eth1 -j ACCEPT
   echo 1 > /proc/sys/net/ipv4/ip_forward
```

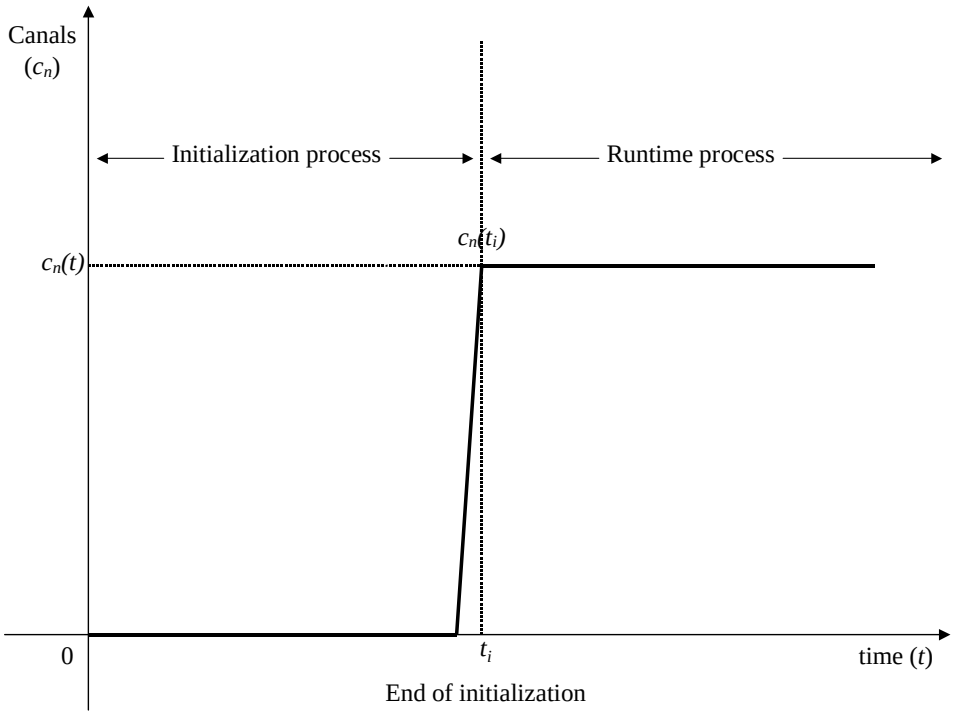


Figure 4.2: Timeline visualization of open-condition approach

4.4 Lattice-Based Method

Motivation for using lattice-based to handle initialisation process is due to the advantage provided by this method to enable the development of security policy based on information flow. Referring to the security strategy to reduce Internet threat defined in Chapter 3, this method can be classified as implementing the strategy to reduce the interaction between unprotected users and untrusted external parties. Review of this method and the formulation of the proposed initialisation process using lattice-based approach are described in the following subsection.

4.4.1 Review of Lattice-Based

Lattice-based is originally proposed by Denning (1976) to model the information flows implied by the given sets of access right in the system. This

method aims to determine the requirements to achieve secure information flows using a set of lattices. Due to its advantage, this model receives wide attentions from the security community. Some works notably afford to re-discuss lattice-based method such as conducted by Sandhu (1993) and Castano *et al.* (1994). According to Denning (1976), information flow policy is a triple $< SC, \rightarrow, \oplus >$ with *SC* denotes a set of security classes, $\rightarrow \subseteq SC \times SC$ is a binary can-flow relation on *SC*, and $\oplus : SC \times SC \rightarrow SC$ is a binary operation for class-combination. Denning also states that the set of security classes *SC* is finite, the can flow relation $\rightarrow$ is a partial order on *SC*, *SC* has a lower bound with respect to $\rightarrow$, and the join operator $\oplus$ is a least upper bound operator (Sandhu, 1993).

Based on the definition above, for $SC = \{U, C, S\}$ with $U \rightarrow U$, $U \rightarrow C$, $U \rightarrow S$, $C \rightarrow S$ and $U \oplus U = U$, $U \oplus C = C$, $U \oplus S = S$, $C \oplus S = S$, the information flow can be visualized using a Hasse diagram as depicted in Figure 4.3. Here the information is flowed upwards from *U* to *S*. It presents a dominance relation $S > C > U$.

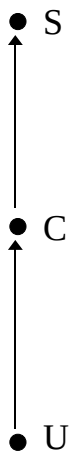


Figure 4.3: The information flow of *S*, *C*, *U*

4.4.2 Formulating Initialisation Process using Lattice-Based

The initialisation process employing lattice-based method is formulated as follows. Let $u_m^i \in U$ becomes an intranet user *m* with the protection level *i*, and let $o_n^j \in O$ becomes an external party *n* with the predetermined safety level *j*. Access request of user u_m^i to external party o_n^j is granted by the firewall if only if $i \geq j$. This policy can be represented by the following equation:

$$\text{Access: } u_m^i \rightarrow o_n^j \Leftrightarrow i \geq j \qquad (4.3)$$

Equation (4.3) states that the protection level of a network user must be greater or at least equal to the safety level of an external party for the requirement of firewall for granting access of a user to Internet. It becomes the basic mechanism to develop lattice-based initialisation process. Implementation is held by designing three protection levels of the internal users i.e. protected, half-protected and unprotected, and three safety level of external parties i.e. trusted, half-trusted and un-trusted. Interactions of each protection level of internal users and the safety level of external parties are presented in Figure 4.4.

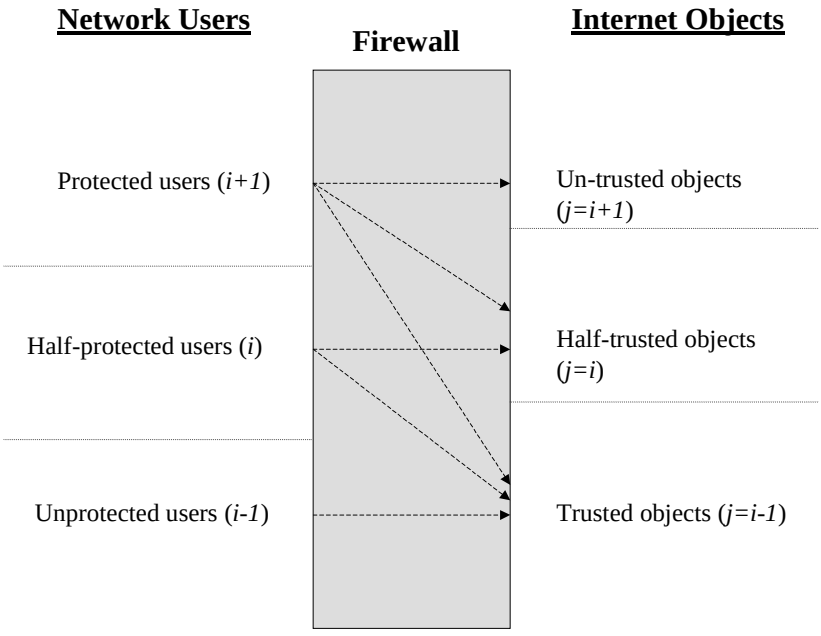


Figure 4.4: Mechanism of lattice-based initialisation

4.4.3 Implementation

Lattice-based initialisation process is implemented by defining the protection level of internal users as follows:

- Protection level 1 (unprotected) is owned by a group of internal machines having IP address 24.4.76.0/24.
- Protection level 2 (half-protected) is owned by a group of internal machines having IP address 24.4.75.0/24.
- Protection level 3 (protected) is owned by a group of internal machines having IP address 24.4.74.0/24.

Meanwhile criteria of the trusted levels to classify Internet objects are subjectively and manually defined by the administrator, in which the definition of these criteria depend on the type and function of the organization. Since this research was held in the university research lab, thus the criteria developed here follow the priorities for facilitating the research. Here it is implemented as follows:

- Trusted level 1 is the trusted group of external parties. The example of this group is the outside parties publishing education-oriented paper works such as IEEE, ACM and Elsevier.
- Trusted level 2 is the half-trusted group of external parties. The example is the outside parties delivering public-oriented news such as Readers Digest, BBC and Kompas.
- Trusted level 3 is the un-trusted group of external parties. The example is the outside parties providing search engine function such as Google, Altavista and Infoseek.

The interaction between the protection levels of intranet users and the trusted levels of external parties is implemented by developing the algorithm as depicted in Figure 4.5. The implemented script of this algorithm is given in Appendix A1.

```
Procedure Lattice-Based Initialisation Process (P, T)
'Input parameter P is the table of protection-level of internal users
'Input parameter T is the table of trusted-level of Internet objects
Var i, j: integer
Begin
        For i = 1 to P.length do
            For j = 1 to T.length do
                If P(i) ≥ T(j) then
                    Activate canal P(i) ↔ T(j)
End
```

Figure 4.5: Algorithm of lattice-based initialisation process

4.5 Experiment

Experiments for evaluating the proposed initialisation methods are presented in this section. Set up of the experiment is shown in Figure 4.6. The developed initialisation programs were run in the firewall machine having specification as Intel Pentium 4 1.6GHz, 128 Kbytes RAM and 40Gbytes local disk. Procedures for conducting experiments are as follows:

- Run each initialisation program in the firewall machine
- Record the security policy produced by each method
- Measure time consumption of each method, and for lattice-based method measurement is done by increasing the number of security rules.

Experimental results are presented as follows. The security policies produced by the initialisation methods are shown in Figure 4.7, 4.8 and 4.9 for close-condition approach, open-condition and lattice-based method respectively. These figures show successful implementation of initialisation methods to set the desired condition to start the runtime operation. Referring to Figure 4.7, close-condition approach requires a mechanism to open up the canal in the runtime to enable Internet access. Meanwhile, Figure 4.8 discloses the necessity of open-condition approach to have security threat detection to inform firewall at the time intrusion is happening, thus firewall can take an action to close the connection. Successful Internet access

restrictions by enforcing security rules are produced by lattice-based initialisation algorithm, in which the results is shown in Figure 4.9. In this figure, security rules for each trusted level of external parties are enforced for each corresponding protection level of internal users. Other results of security rules enforced by lattice-based initialisation for different rules number are also given in Appendix A2.

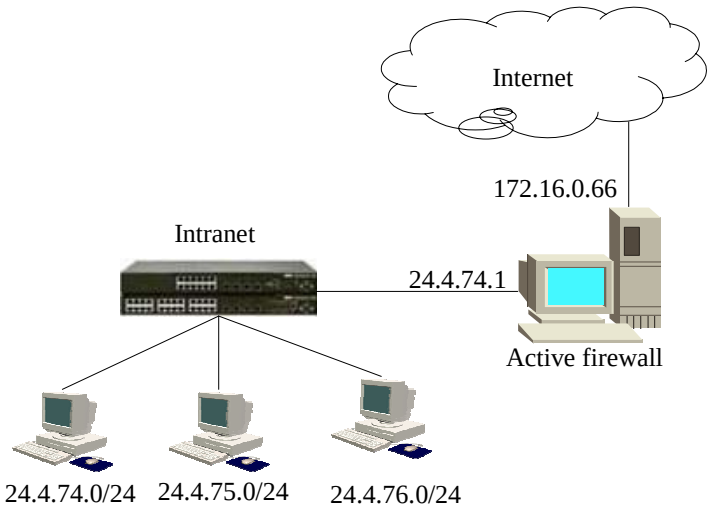


Figure 4.6: Set up of the experiment

```
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  anywhere             anywhere

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Figure 4.7: Security policy produced by close-condition approach

```
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Figure 4.8: Security policy produced by open-condition approach

```
Chain INPUT (policy DROP)
target     prot opt source               destination

Chain FORWARD (policy DROP)
target     prot opt source               destination
ACCEPT     tcp  --  24.4.74.0/24         209.202.220.97
ACCEPT     tcp  --  209.202.220.97       24.4.74.0/24
ACCEPT     tcp  --  24.4.74.0/24         212.58.240.120
ACCEPT     tcp  --  212.58.240.120       24.4.74.0/24
ACCEPT     tcp  --  24.4.74.0/24         129.35.76.177
ACCEPT     tcp  --  129.35.76.177        24.4.74.0/24
ACCEPT     tcp  --  24.4.75.0/24         212.58.240.120
ACCEPT     tcp  --  212.58.240.120       24.4.75.0/24
ACCEPT     tcp  --  24.4.75.0/24         129.35.76.177
ACCEPT     tcp  --  129.35.76.177        24.4.75.0/24
ACCEPT     tcp  --  24.4.76.0/24         129.35.76.177
ACCEPT     tcp  --  129.35.76.177        24.4.76.0/24

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Figure 4.9: Security policy produced by lattice-based method

Results of measuring time consumption are presented in Table 4.1, Table 4.2, and Table 4.3 for close-condition, open-condition and lattice-based respectively. Graph presentation of these data is given in Figure 4.10.

Table 4.1: Close-Condition

No.	Start Time (time)	End Time (time)	Elapsed Time (seconds)
1	45.46575342	45.46575	0
2	50.89473684	50.89474	0
3	53.68493151	53.68493	0
4	56.6344086	56.63441	0
5	0.276923077	0.276923	0
6	11.69135802	11.69136	0
7	25.01886792	25.01887	0
8	29.79487179	29.79487	0
9	33.27272727	33.27273	0
10	36.90697674	36.90698	0
11	43.97647059	43.97647	0
Average			0
Std dev			0

Table 4.2: Open-Condition

No.	Start Time (time)	End Time (time)	Elapsed Time (seconds)
1	50.85	50.85	0
2	55.66666667	55.66666667	0
3	59.13846154	59.13846154	0
4	3	3	0
5	8.962962963	8.962962963	0
6	11.81081081	11.81081081	0
7	14.88541667	14.88541667	0
8	19.15151515	19.15151515	0
9	24	24	0
10	28.88541667	28.88541667	0
Average			0
Std dev			0

The graph in Figure 4.10 shows close-condition and open-condition take very small time consumption, i.e. $\Delta t \cong 0$. This condition is due to simple process executed by these methods. Meanwhile lattice-based method takes longer time consumption to complete its process, in which referring to the algorithm of lattice initialisation depicted in Figure 4.5, the processing time can be computed using the following formula.

$$\Delta t = np \times nt \qquad (4.4)$$

with *np* refers to the number of the predefined protection level, and *nt* is the number of external parties. Based on the implementation stage presented in Section 4.4.3, *np*=3, while *nt* is a variable. In this experiment, *nt* ranges from 1 to 5, as shown in Table 4.3. Comparing the processing time obtained using Equation (4.4) against the experimental results in Table 4.3 produces the data as shown in Table 4.4. As can be seen from Table 4.4, similar data are produced from both computation and empirical measurements. It is evident that the consumed time of lattice-based initialisation is linear to the number of rules being used in the initialisation process. This phenomenon is also shown by the graph of processing time in Figure 4.10.

Table 4.3: Lattice-Based

No.	1 rule Elapsed Time (seconds)	2 rule Elapsed Time (seconds)	3 rule Elapsed Time (seconds)	5 rule Elapsed Time (seconds)
1	0.6814904	1.5802469	2.6428571	5.1186299
2	0.6214158	1.703871	2.5263158	5.2997619
3	0.7367484	1.4949101	2.4084507	5.0096154
4	0.8723404	1.5282682	2.8947368	5.0076923
5	0.7195908	1.8014027	2.3032828	4.8507099
6	0.9420635	1.6369048	2.6516921	4.870269
7	0.925	1.5957035	2.4325397	4.9344262
8	0.9927596	1.7791925	2.6591199	5.1429078
9	0.8322119	1.8268008	2.6634409	4.1143376
10	0.9759317	1.8813559	2.4420842	4.6762452
11	0.7679475	1.6916667	2.6603774	4.3934938
12	1	1.8135593	2.4083333	5.0192308
13	0.6699929	1.8355099	3.1706798	4.7044289
14	0.9791667	1.8612981	2.6735019	4.8090909
15	0.9285714	1.7813559	2.5395702	4.7811265
16	0.6203736	1.862069	2.2600733	4.794772
17	0.8326118	1.4545455	2.3977117	4.554569
18	0.3134921	1.475641	2.6285609	4.5821386
19	0.5954156	1.8666667	2.9523185	4.8124171
20	1.0333333	1.7692308	3.2174899	5.1855006
21	0.3508696	1.6206897	2.9264347	4.625
22	1.0317693	1.7669837	2.9341282	5.3122587
23	0.4592889	1.798951	2.6079658	4.5419192
24	1.008547	1.64	2.7075893	4.9803922
25	0.8884381	1.6666667	2.4350193	4.7929167
26	0.8454545	1.578125	2.2633929	4.9356187
27	0.6772727	1.6470457	2.7837349	4.8607804
28	0.8038847	1.475891	2.717033	4.9393939
29	0.7664577	1.8852459	2.9859155	4.3073752
30	1.0585271	1.837146	2.4051587	5.008
Average	0.7976989	1.7052315	2.6433328	4.8321673
Std dev	0.1988975	0.1371058	0.2583105	0.2776129

Table 4.4: Time consumptions of lattice-based initialisation

Number of Rules	Theoretical (seconds)	Empirical (seconds)
1	0.9	0.798
2	1.8	1.705
3	2.7	2.643
5	4.5	4.832

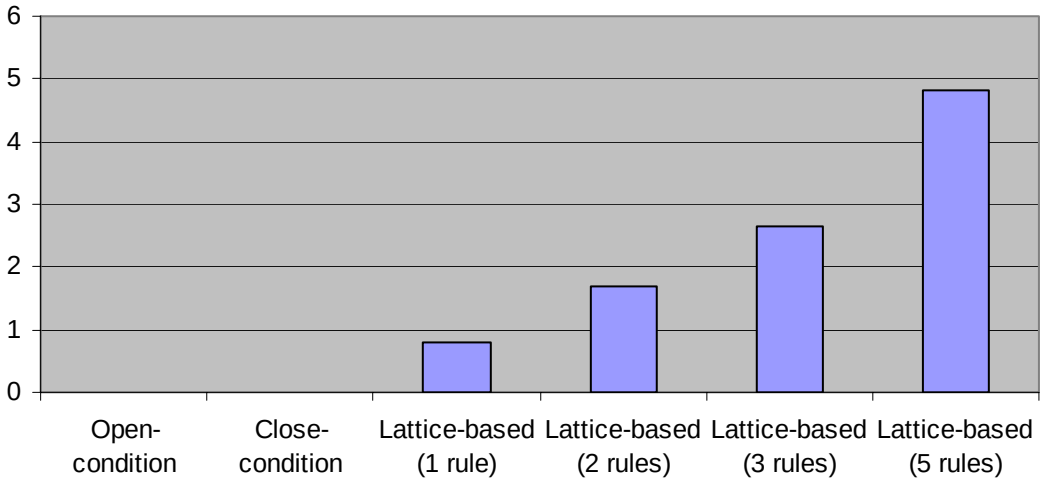


Figure 4.10: Graph presentation of time consumption

4.6 Summary

Three methods for handling initialisation process have been presented in this chapter. They consist of close-condition, open-condition, and lattice-based method. Close-condition drop all canals as the result of initialisation, while open-condition open all canals. Lattice-based takes a different approach by opening only the canals having equalled or higher protection level than the trusted level of external parties. Experiment show that very small processing time is consumed by close and open-condition. Meanwhile, lattice-based produces a linear processing time against the number of security rules.

CHAPTER 4

INITIALISATION PROCESS

4.1 Introduction

Developing initialisation strategy is essential to set firewall into the suitable condition for runtime process. According to Ren *et al.* (2004), initialisation refers to the procedure of starting up and bringing the system to a point on its operation. To develop this procedure condition before and after the firewall turned up are identified.

4.2 Closed-Condition Approach

This approach is based on the experience of Ranum and Avolio (1994), which suggest to turning off system services at minimum to have more secure system. Thus in this effort, it intends to minimize the availability of services in the beginning of runtime operation. Formalization of close-condition approach is described as follow.

Let firewall controls a set of reconfigureable canals $C = \{c_1, c_2, \text{K}, c_n, \text{K}, c_k\}$ in which each canal c_n corresponds to the individual intranet machine n. Starting up the firewall means to bring the firewall from the off condition at time $t = t_0$ to the initial condition at time $t = t_i$, where firewall is ready to hold its runtime process.

Hence close-condition approach introduced here forces the firewall to close all available canal at time $t = t_i$, thus it can be formalized as follows

$$\forall c_n(t_i) = 0 \qquad (4.1)$$

A timeline visualization of this approach is given in Figure 4.1.

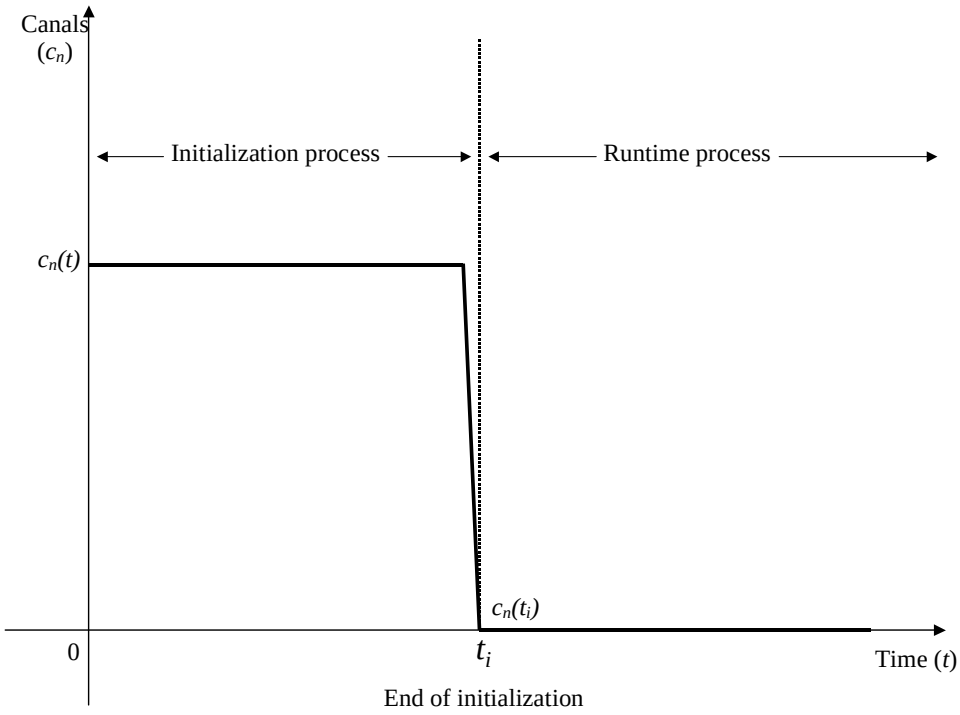


Figure 4.1: Timeline visualization of close-condition approach

It can be noted that close-condition approach enables firewall to always authenticate all the requested Internet access, thus strict Internet transactions are produced at runtime. Here traffic analyser and canal modification must be established to support the function of firewall to organize external network transaction, otherwise no internal machine can reach the Internet since no opened canals are available at runtime. Implementation of this approach is conducted by developing the following script:

```
# To have default DROP policy
   iptables -P INPUT ACCEPT
   iptables -P OUTPUT ACCEPT
   iptables -P FORWARD ACCEPT

# Delete and flush all rules
   iptables --flush
   iptables --table nat --flush
   iptables --delete-chain
   iptables --table nat --delete-chain

# Set up IP Forwarding and Masquerading
   iptables -T nat -A POSTROUTING --out-interface eth0 -j MASQUERADE
   iptables -A FORWARD --in-interface eth1 -j DROP
   echo 1 > /proc/sys/net/ipv4/ip_forward
```

4.3 Open-Condition Approach

Just like the mechanism of the close-condition approach, this method similarly projects the firewall into its edge-condition. Rather than closing all available canals as held by close-condition, here canals are opened after passing the initialisation stage. It can be formalized as follows

$$\forall c_n(t_i) = 1 \qquad (4.2)$$

Visualization of this approach is given in Figure 4.2. This method is implemented by executing the following script:

```
# To have default DROP policy
   iptables -P INPUT ACCEPT
   iptables -P OUTPUT ACCEPT
   iptables -P FORWARD ACCEPT

# Delete and flush.
   iptables --flush
   iptables --table nat --flush
   iptables --delete-chain
   iptables --table nat --delete-chain

# Set up IP Forwarding and Masquerading
   iptables -T nat -A POSTROUTING --out-interface eth0 -j MASQUERADE
   iptables -A FORWARD --in-interface eth1 -j ACCEPT
   echo 1 > /proc/sys/net/ipv4/ip_forward
```

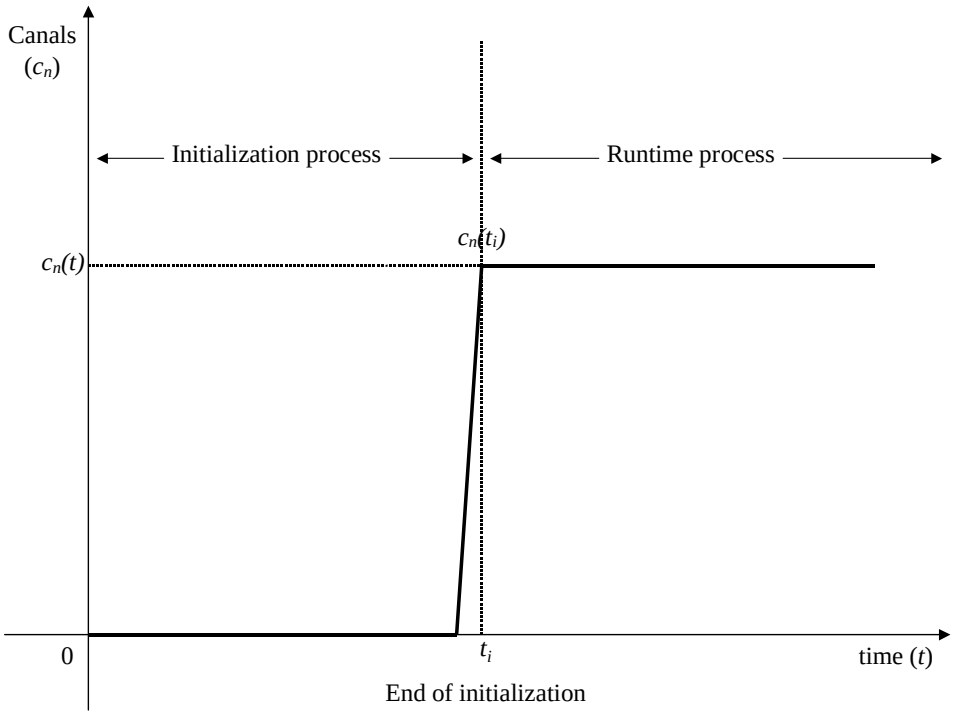


Figure 4.2: Timeline visualization of open-condition approach

4.4 Lattice-Based Method

Motivation for using lattice-based to handle initialisation process is due to the advantage provided by this method to enable the development of security policy based on information flow. Referring to the security strategy to reduce Internet threat defined in Chapter 3, this method can be classified as implementing the strategy to reduce the interaction between unprotected users and untrusted external parties. Review of this method and the formulation of the proposed initialisation process using lattice-based approach are described in the following subsection.

4.4.1 Review of Lattice-Based

Lattice-based is originally proposed by Denning (1976) to model the information flows implied by the given sets of access right in the system. This

method aims to determine the requirements to achieve secure information flows using a set of lattices. Due to its advantage, this model receives wide attentions from the security community. Some works notably afford to re-discuss lattice-based method such as conducted by Sandhu (1993) and Castano *et al.* (1994). According to Denning (1976), information flow policy is a triple $< SC, \rightarrow, \oplus >$ with *SC* denotes a set of security classes, $\rightarrow \subseteq SC \times SC$ is a binary can-flow relation on *SC*, and $\oplus : SC \times SC \rightarrow SC$ is a binary operation for class-combination. Denning also states that the set of security classes *SC* is finite, the can flow relation $\rightarrow$ is a partial order on *SC*, *SC* has a lower bound with respect to $\rightarrow$, and the join operator $\oplus$ is a least upper bound operator (Sandhu, 1993).

Based on the definition above, for $SC = \{U, C, S\}$ with $U \rightarrow U$, $U \rightarrow C$, $U \rightarrow S$, $C \rightarrow S$ and $U \oplus U = U$, $U \oplus C = C$, $U \oplus S = S$, $C \oplus S = S$, the information flow can be visualized using a Hasse diagram as depicted in Figure 4.3. Here the information is flowed upwards from *U* to *S*. It presents a dominance relation $S > C > U$.

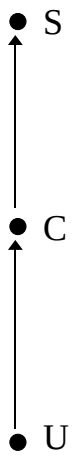


Figure 4.3: The information flow of *S*, *C*, *U*

4.4.2 Formulating Initialisation Process using Lattice-Based

The initialisation process employing lattice-based method is formulated as follows. Let $u_m^i \in U$ becomes an intranet user *m* with the protection level *i*, and let $o_n^j \in O$ becomes an external party *n* with the predetermined safety level *j*. Access request of user u_m^i to external party o_n^j is granted by the firewall if only if $i \geq j$. This policy can be represented by the following equation:

$$\text{Access: } u_m^i \rightarrow o_n^j \Leftrightarrow i \geq j \qquad (4.3)$$

Equation (4.3) states that the protection level of a network user must be greater or at least equal to the safety level of an external party for the requirement of firewall for granting access of a user to Internet. It becomes the basic mechanism to develop lattice-based initialisation process. Implementation is held by designing three protection levels of the internal users i.e. protected, half-protected and unprotected, and three safety level of external parties i.e. trusted, half-trusted and un-trusted. Interactions of each protection level of internal users and the safety level of external parties are presented in Figure 4.4.

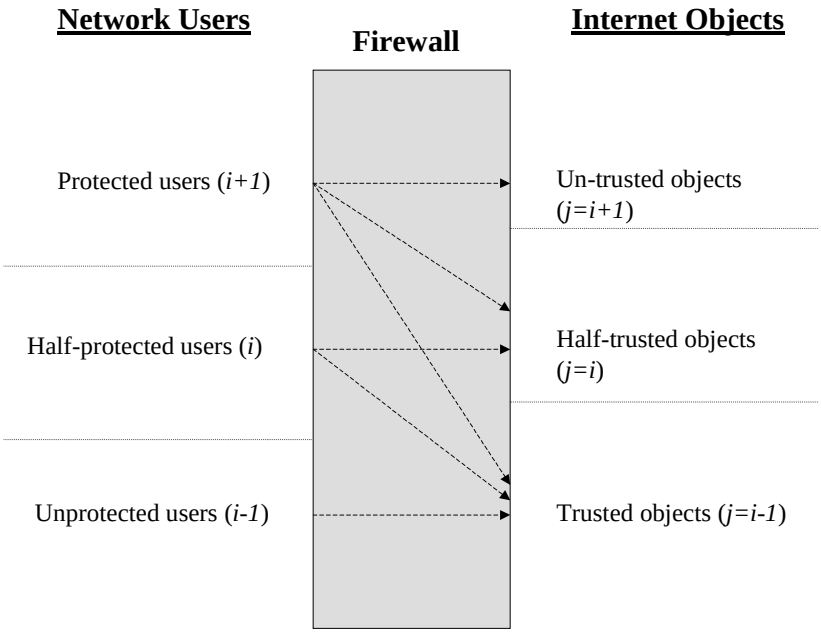


Figure 4.4: Mechanism of lattice-based initialisation

4.4.3 Implementation

Lattice-based initialisation process is implemented by defining the protection level of internal users as follows:

- Protection level 1 (unprotected) is owned by a group of internal machines having IP address 24.4.76.0/24.
- Protection level 2 (half-protected) is owned by a group of internal machines having IP address 24.4.75.0/24.
- Protection level 3 (protected) is owned by a group of internal machines having IP address 24.4.74.0/24.

Meanwhile criteria of the trusted levels to classify Internet objects are subjectively and manually defined by the administrator, in which the definition of these criteria depend on the type and function of the organization. Since this research was held in the university research lab, thus the criteria developed here follow the priorities for facilitating the research. Here it is implemented as follows:

- Trusted level 1 is the trusted group of external parties. The example of this group is the outside parties publishing education-oriented paper works such as IEEE, ACM and Elsevier.
- Trusted level 2 is the half-trusted group of external parties. The example is the outside parties delivering public-oriented news such as Readers Digest, BBC and Kompas.
- Trusted level 3 is the un-trusted group of external parties. The example is the outside parties providing search engine function such as Google, Altavista and Infoseek.

The interaction between the protection levels of intranet users and the trusted levels of external parties is implemented by developing the algorithm as depicted in Figure 4.5. The implemented script of this algorithm is given in Appendix A1.

```
Procedure Lattice-Based Initialisation Process (P, T)
'Input parameter P is the table of protection-level of internal users
'Input parameter T is the table of trusted-level of Internet objects
Var i, j: integer
Begin
        For i = 1 to P.length do
            For j = 1 to T.length do
                If P(i) ≥ T(j) then
                    Activate canal P(i) ↔ T(j)
End
```

Figure 4.5: Algorithm of lattice-based initialisation process

4.5 Experiment

Experiments for evaluating the proposed initialisation methods are presented in this section. Set up of the experiment is shown in Figure 4.6. The developed initialisation programs were run in the firewall machine having specification as Intel Pentium 4 1.6GHz, 128 Kbytes RAM and 40Gbytes local disk. Procedures for conducting experiments are as follows:

- Run each initialisation program in the firewall machine
- Record the security policy produced by each method
- Measure time consumption of each method, and for lattice-based method measurement is done by increasing the number of security rules.

Experimental results are presented as follows. The security policies produced by the initialisation methods are shown in Figure 4.7, 4.8 and 4.9 for close-condition approach, open-condition and lattice-based method respectively. These figures show successful implementation of initialisation methods to set the desired condition to start the runtime operation. Referring to Figure 4.7, close-condition approach requires a mechanism to open up the canal in the runtime to enable Internet access. Meanwhile, Figure 4.8 discloses the necessity of open-condition approach to have security threat detection to inform firewall at the time intrusion is happening, thus firewall can take an action to close the connection. Successful Internet access

restrictions by enforcing security rules are produced by lattice-based initialisation algorithm, in which the results is shown in Figure 4.9. In this figure, security rules for each trusted level of external parties are enforced for each corresponding protection level of internal users. Other results of security rules enforced by lattice-based initialisation for different rules number are also given in Appendix A2.

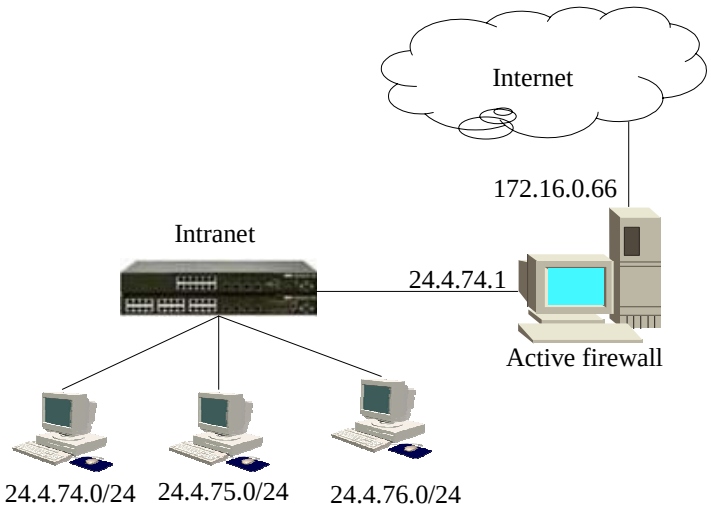


Figure 4.6: Set up of the experiment

```
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  anywhere             anywhere

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Figure 4.7: Security policy produced by close-condition approach

```
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Figure 4.8: Security policy produced by open-condition approach

```
Chain INPUT (policy DROP)
target     prot opt source               destination

Chain FORWARD (policy DROP)
target     prot opt source               destination
ACCEPT     tcp  --  24.4.74.0/24         209.202.220.97
ACCEPT     tcp  --  209.202.220.97       24.4.74.0/24
ACCEPT     tcp  --  24.4.74.0/24         212.58.240.120
ACCEPT     tcp  --  212.58.240.120       24.4.74.0/24
ACCEPT     tcp  --  24.4.74.0/24         129.35.76.177
ACCEPT     tcp  --  129.35.76.177        24.4.74.0/24
ACCEPT     tcp  --  24.4.75.0/24         212.58.240.120
ACCEPT     tcp  --  212.58.240.120       24.4.75.0/24
ACCEPT     tcp  --  24.4.75.0/24         129.35.76.177
ACCEPT     tcp  --  129.35.76.177        24.4.75.0/24
ACCEPT     tcp  --  24.4.76.0/24         129.35.76.177
ACCEPT     tcp  --  129.35.76.177        24.4.76.0/24

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Figure 4.9: Security policy produced by lattice-based method

Results of measuring time consumption are presented in Table 4.1, Table 4.2, and Table 4.3 for close-condition, open-condition and lattice-based respectively. Graph presentation of these data is given in Figure 4.10.

Table 4.1: Close-Condition

No.	Start Time (time)	End Time (time)	Elapsed Time (seconds)
1	45.46575342	45.46575	0
2	50.89473684	50.89474	0
3	53.68493151	53.68493	0
4	56.6344086	56.63441	0
5	0.276923077	0.276923	0
6	11.69135802	11.69136	0
7	25.01886792	25.01887	0
8	29.79487179	29.79487	0
9	33.27272727	33.27273	0
10	36.90697674	36.90698	0
11	43.97647059	43.97647	0
Average			0
Std dev			0

Table 4.2: Open-Condition

No.	Start Time (time)	End Time (time)	Elapsed Time (seconds)
1	50.85	50.85	0
2	55.66666667	55.66666667	0
3	59.13846154	59.13846154	0
4	3	3	0
5	8.962962963	8.962962963	0
6	11.81081081	11.81081081	0
7	14.88541667	14.88541667	0
8	19.15151515	19.15151515	0
9	24	24	0
10	28.88541667	28.88541667	0
Average			0
Std dev			0

The graph in Figure 4.10 shows close-condition and open-condition take very small time consumption, i.e. $\Delta t \cong 0$. This condition is due to simple process executed by these methods. Meanwhile lattice-based method takes longer time consumption to complete its process, in which referring to the algorithm of lattice initialisation depicted in Figure 4.5, the processing time can be computed using the following formula.

$$\Delta t = np \times nt \qquad (4.4)$$

with *np* refers to the number of the predefined protection level, and *nt* is the number of external parties. Based on the implementation stage presented in Section 4.4.3, *np*=3, while *nt* is a variable. In this experiment, *nt* ranges from 1 to 5, as shown in Table 4.3. Comparing the processing time obtained using Equation (4.4) against the experimental results in Table 4.3 produces the data as shown in Table 4.4. As can be seen from Table 4.4, similar data are produced from both computation and empirical measurements. It is evident that the consumed time of lattice-based initialisation is linear to the number of rules being used in the initialisation process. This phenomenon is also shown by the graph of processing time in Figure 4.10.

Table 4.3: Lattice-Based

No.	1 rule Elapsed Time (seconds)	2 rule Elapsed Time (seconds)	3 rule Elapsed Time (seconds)	5 rule Elapsed Time (seconds)
1	0.6814904	1.5802469	2.6428571	5.1186299
2	0.6214158	1.703871	2.5263158	5.2997619
3	0.7367484	1.4949101	2.4084507	5.0096154
4	0.8723404	1.5282682	2.8947368	5.0076923
5	0.7195908	1.8014027	2.3032828	4.8507099
6	0.9420635	1.6369048	2.6516921	4.870269
7	0.925	1.5957035	2.4325397	4.9344262
8	0.9927596	1.7791925	2.6591199	5.1429078
9	0.8322119	1.8268008	2.6634409	4.1143376
10	0.9759317	1.8813559	2.4420842	4.6762452
11	0.7679475	1.6916667	2.6603774	4.3934938
12	1	1.8135593	2.4083333	5.0192308
13	0.6699929	1.8355099	3.1706798	4.7044289
14	0.9791667	1.8612981	2.6735019	4.8090909
15	0.9285714	1.7813559	2.5395702	4.7811265
16	0.6203736	1.862069	2.2600733	4.794772
17	0.8326118	1.4545455	2.3977117	4.554569
18	0.3134921	1.475641	2.6285609	4.5821386
19	0.5954156	1.8666667	2.9523185	4.8124171
20	1.0333333	1.7692308	3.2174899	5.1855006
21	0.3508696	1.6206897	2.9264347	4.625
22	1.0317693	1.7669837	2.9341282	5.3122587
23	0.4592889	1.798951	2.6079658	4.5419192
24	1.008547	1.64	2.7075893	4.9803922
25	0.8884381	1.6666667	2.4350193	4.7929167
26	0.8454545	1.578125	2.2633929	4.9356187
27	0.6772727	1.6470457	2.7837349	4.8607804
28	0.8038847	1.475891	2.717033	4.9393939
29	0.7664577	1.8852459	2.9859155	4.3073752
30	1.0585271	1.837146	2.4051587	5.008
Average	0.7976989	1.7052315	2.6433328	4.8321673
Std dev	0.1988975	0.1371058	0.2583105	0.2776129

Table 4.4: Time consumptions of lattice-based initialisation

Number of Rules	Theoretical (seconds)	Empirical (seconds)
1	0.9	0.798
2	1.8	1.705
3	2.7	2.643
5	4.5	4.832

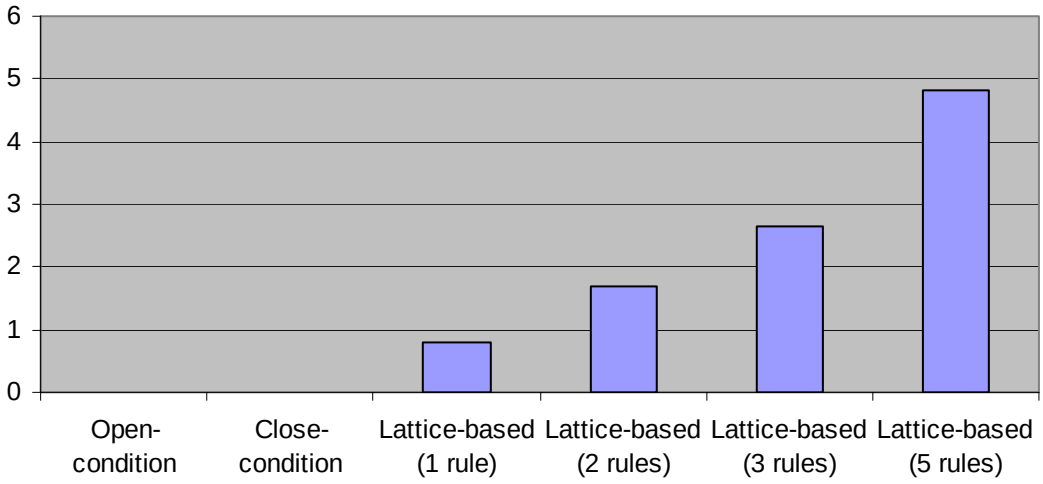


Figure 4.10: Graph presentation of time consumption

4.6 Summary

Three methods for handling initialisation process have been presented in this chapter. They consist of close-condition, open-condition, and lattice-based method. Close-condition drop all canals as the result of initialisation, while open-condition open all canals. Lattice-based takes a different approach by opening only the canals having equalled or higher protection level than the trusted level of external parties. Experiment show that very small processing time is consumed by close and open-condition. Meanwhile, lattice-based produces a linear processing time against the number of security rules.

CHAPTER 5

RUNTIME PROCESS: MINIMIZING THE INTERACTION BETWEEN UNPROTECTED USERS AND UNTRUSTED EXTERNAL PARTIES

5.1 Introduction

This chapter presents the development of runtime process for minimizing the interaction between unprotected users and untrusted external parties as defined in the security strategy in Chapter 3. The methods proposed in this stage are to follow initialisation process presented in Chapter 4. With *t* represent the current time and t_i denote the ending of initialisation process, the operations of firewall at $t > t_i$ are discussed. To carry on this task, a mechanism to identify which users are unprotected and which external parties are untrusted, is developed as well as the mechanism to measure the degree of risk produced from each Internet transaction. Since this method requires quantifying the abstract parameters such as safety and risk of an Internet transaction, thus fuzzy reasoning is employed to handle this task. Referring to the work of (Negnevitsky, 2002) that shows successful application of fuzzy reasoning for dealing with abstract parameter, and Labuschagne and Eloff (1998) and Kim *et al.* (2004) that show the applicability of fuzzy logic in handling real-time analysis on network traffic, fuzzy reasoning is utilized to observe the content of network traffic passing through the firewall. The developed strategy is to define some parameters indicating or causing the appearance of the threats and to formulate their membership function contributing to the risk for accessing Internet. The

advantage of this method is due to continues observation of network traffic, therefore any arising threats originated from Internet transaction in any time can be detected, thus the active firewall can take an action to secure internal network.

5.2 Review of Fuzzy Logic in Network Security

Since its introduction by Zadeh (1965), fuzzy logic has been applied to many systems with the purpose is to inject some degree of intelligence (Klir and Yuan, 1995; Yen *et al.*, 1995). Today, the implementations of fuzzy logic can be found in many disciplines such as finance sector, traffic control and automobile, information system, to medical science. In the domain of network security, fuzzy logic has successfully presented methods to describe the parameters that are difficult to quantify such as the safety and the risk of the network. The work of Ru and Eloff (1996) notably introduced the risk analysis method using fuzzy logic, and was followed by Labuschagne and Eloff (1998) to develop real time risk analysis for securing network. The last effort aims to support firewall to perform identification and authentication on specific users by measuring the global risk of every transaction. Besides observing the header of IP and TCP packet, no traffic content are taken into account by this method, therefore only shallow analysis can be held. Meanwhile, the work of Zou *et al.* (2002) for developing fuzzy adaptive security algorithm for intelligent firewall, consider only the membership function of the security level that are manually determined based on the source and destination of the packet. Although the proposed method is claimed resolving the conflict between security and speed, however the approach applied in the developed algorithm simplifies the analysis of the flowing packet using the predetermined security level. Therefore the algorithm has minimal capability to identify the threat on the flowing traffic. This condition also leads to the dependency to manually set the security levels. Moreover, it seems inapplicable to manually assign security level to every external party.

The effort of Guan *et al.* (2004) to develop intrusion detection systems and Kim *et al.* (2004) to develop network forensic afford to employ fuzzy reasoning to

conduct thorough analysis on network traffic in order to determine the pattern of network intrusion. However the proposed methods consider only the header of network packet. Thus it will be difficult to identify the threat contained in the flowing data. In this research, fuzzy reasoning is used to support the mechanism of active firewall to determine the security level of the flowing packet. Unlike the works of Labuschagne and Eloff (1998) that rely only on the packet header or Zou *et al.* (2002) that manually determine the security level of the packet, here automatic threat analysis is held on the content of the flowing packet. Based on this analysis, security policy of the firewall is adaptively updated during the runtime process. Formulation of this method is described in the next section.

5.3 Network Traffic Analysis

Preliminary works to manually analyse the content of network traffic were held to disclose the factors influencing Internet threat. This effort observed some parameters that might indicate the appearance of the threats as follow, the existence of executable file (*E*), the effort to force user for reading the external information (*F*), the number of advertisements (*A*), the number of external servers involved in a single Internet transaction (*M*), the number of active script (*S*) and the number of cookies (*C*). To verify the influence of these parameters for causing the risk of accessing Internet, a set of network traffics generated from some well-known Internet parties were collected and scrutinized. Here the usage of well-known Internet parties is important in order to facilitate the justification of the safety of accessing Internet i.e. to determine whether the external party is a normal party or causing threat. Thus, the above parameters that lead to the threat could be identified. Results of the observation are presented in Table 5.1.

By proportioning the mean υ of the existence of each predetermined parameter of normal parties against the threat parties as shown in the last row of Table 5.1, and also following the rule of influence presented in Negnevitsky (2002), data presented in Table 5.1 show some phenomenon as follow. The existence of executable file and forced information carried by the external parties absolutely

causes the threat to the internal network users. The number of advertisements slightly indicates the threat, and the numbers of external servers has less influence to the appearance of threat. Meanwhile the number of cookies and active scripts cannot be used to show the appearance of Internet threat since the proportion of the mean of these parameters from normal and threat parties produce the values close to or greater than one. It means these parameters are indifferent both in normal and threat parties, it is shown by large numbers of active scripts and cookies can be found in both parties. These phenomenons are obviously presented in Figure 5.1 below. Therefore, in this research only first four-listed parameters above are used in analysing Internet threats, since the number of cookies and active scripts cannot be employed to detect the Internet threats.

Table 5.1: Measuring the factors influencing the threat

Name of External Party	**Status**	***E***	***F***	***A***	***M***	***C***	***S***
www.ukm.my	safe	0	0	0	0	2	2
www.acm.org	safe	0	0	0	0	3	1
www.utm.my	safe	0	0	0	0	3	0
www.mfa.go.th	safe	0	0	0	0	3	9
www.elsevier.com	safe	0	0	2	1	4	43
www.ieee.com	safe	0	0	1	3	7	15
www.xxx.com	threat	0	0	2	4	4	2
www.babes.com	threat	0	0	2	7	4	6
www.sex.com	threat	0	4	3	9	2	0
www.porn.com	threat	4	90	9	3	6	33
υ of safe parties		0	0	1.5	2	3.67	11.67
υ of threat parties		1	23.5	4	5.75	4	10.25
υ of safe / υ of threat		0	0	0.375	0.348	0.918	1.139

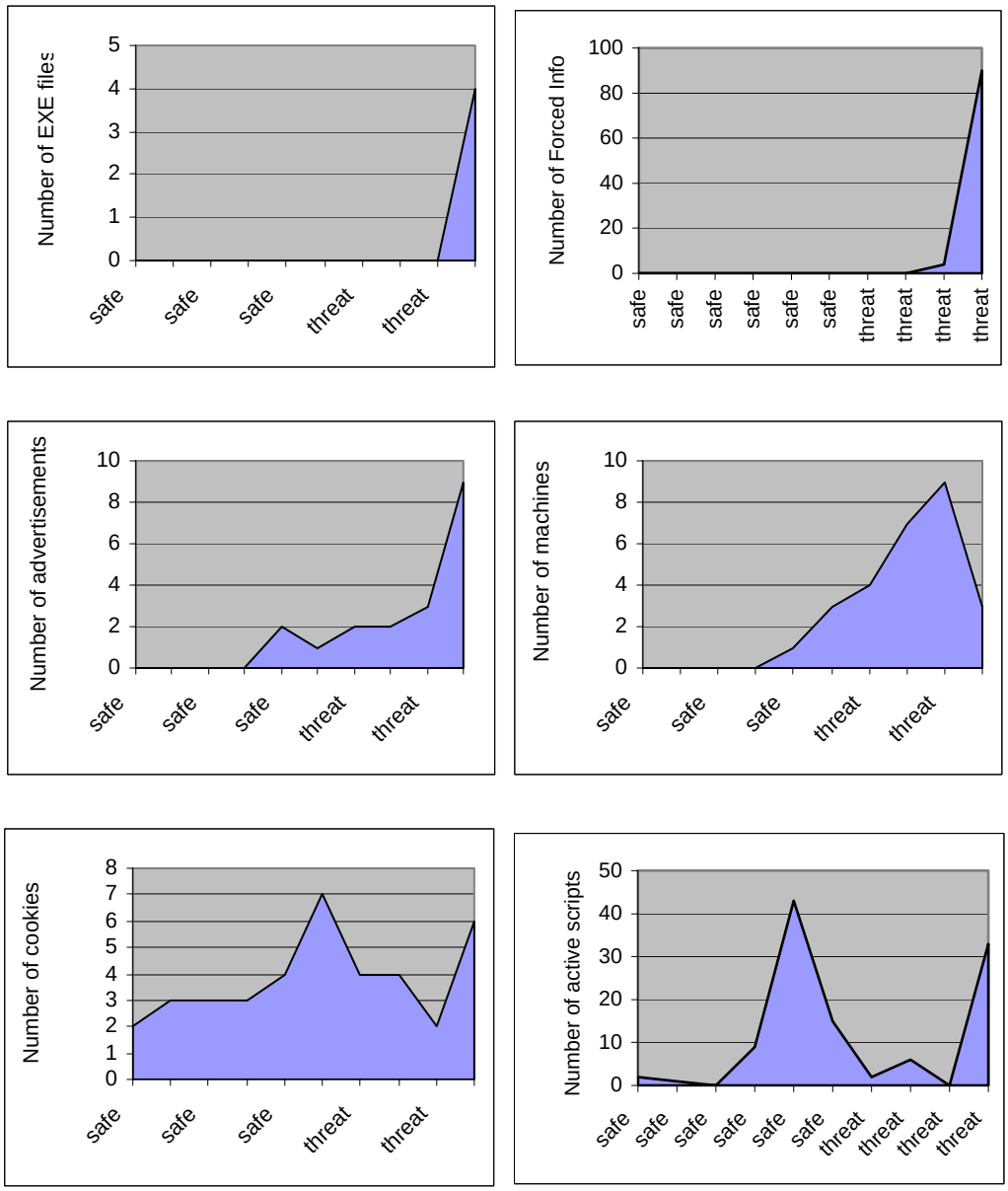


Figure 5.1: Effects of some defined parameters to indicate threats

5.4 Formulating Security Rules Update using Fuzzy Reasoning

Let *Y* be the universe of discourse with elements *y* that denotes the threat of the network traffic passing through the firewall. And let *E*, *F*, *A*, and *M* be the sets of components representing the existence of the executable file, the effort to force internal user to read the information from external parties, the number of advertisements, and the number of the external server involved in a single Internet

transaction respectively, hence fuzzy set *E*, *F*, *A*, and *M* of universe *Y* are defined by the functions $\mu_E(y)$, $\mu_F(y)$, $\mu_A(y)$, and $\mu_M(y)$. Those functions are called the membership function of set *E*, *F*, *A*, and *M* respectively, and are described as follow.

$$\begin{aligned} E &= \{y, \mu_E(y)\} \quad y \in Y, \mu_E(y) : Y \to [0,1] \\ F &= \{y, \mu_F(y)\} \quad y \in Y, \mu_F(y) : Y \to [0,1] \\ A &= \{y, \mu_A(y)\} \quad y \in Y, \mu_A(y) : Y \to [0,1] \\ M &= \{y, \mu_M(y)\} \quad y \in Y, \mu_M(y) : Y \to [0,1] \end{aligned} \tag{5.1}$$

with

$$\mu_E(y) = \begin{cases} 1 \Leftrightarrow E = 0 \\ 0 \Leftrightarrow E > 0 \end{cases} \tag{5.2}$$

$$\mu_F(y) = \begin{cases} 1 \Leftrightarrow F = 0 \\ 0 \Leftrightarrow F > 0 \end{cases} \tag{5.3}$$

$$\mu_A(y) = 1 - \frac{1}{A^{1.7}} \tag{5.4}$$

$$\mu_M(y) = 1 - \frac{1}{2\sqrt{M}} \tag{5.5}$$

The graphs visualizing the membership function of the parameters defined in Equation (5.2) to (5.5), are depicted in Figure 5.2 to 5.5. These graphs become the input of the fuzzy-based security rules update to determine the security level of every external party.

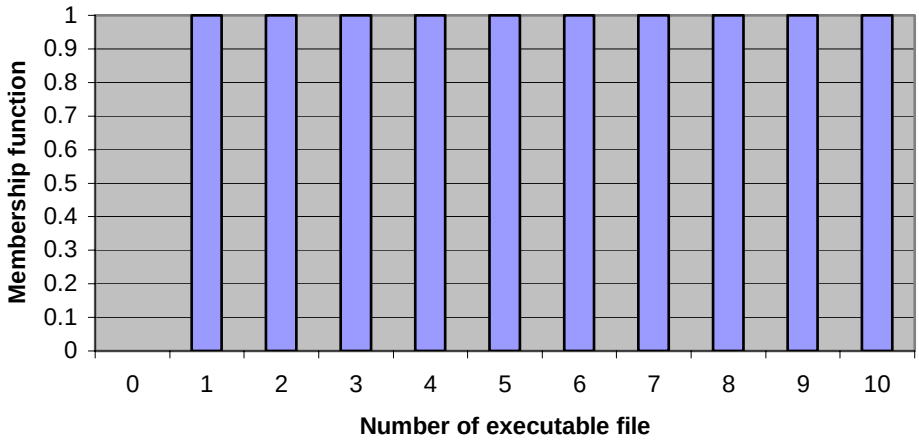


Figure 5.2: The membership function of executable files

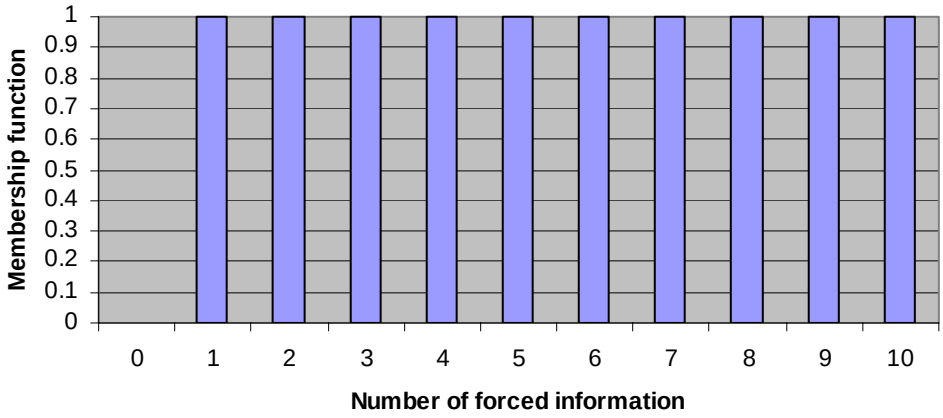


Figure 5.3: The membership function of forced information

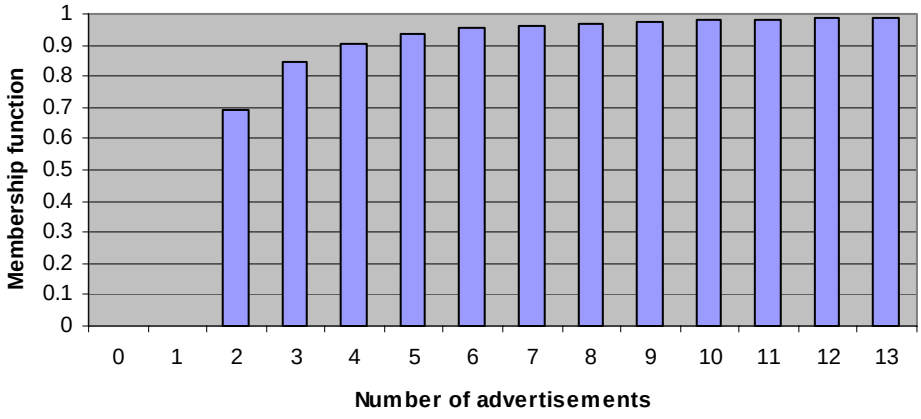


Figure 5.4: The membership function of the number of advertisements

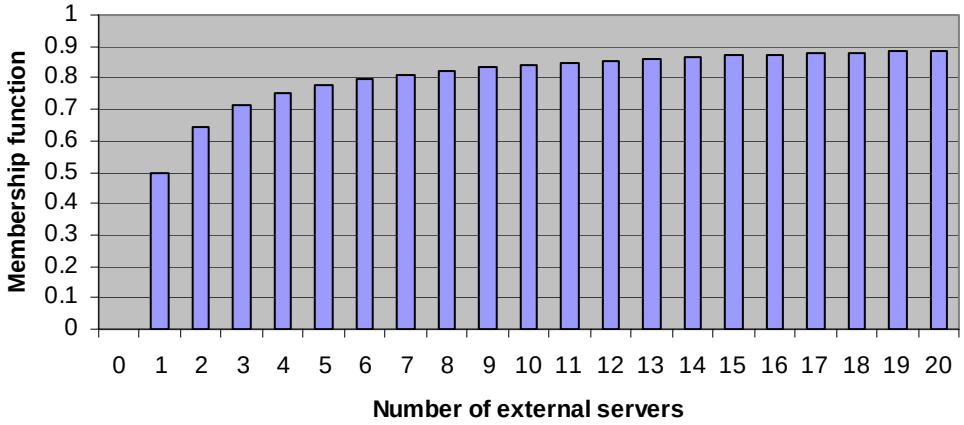


Figure 5.5: The membership function of the number of external machines

To obtain the total threat produced by the membership function of the factors defined above, the equation below is computed.

$$\begin{aligned} x &= \mu_E(y) \,\mathrm{Y}\, \mu_F(y) \,\mathrm{Y}\, \mu_A(y) \,\mathrm{Y}\, \mu_M(y) \\ &= \max\left[\mu_E(y), \mu_F(y), \mu_A(y), \mu_M(y)\right] \end{aligned} \tag{5.6}$$

with x denote the total threat. Let X becomes the universe of discourse of the threat in which $x \in X$. And let *LR*, *MR*, and *HR* become the subsets of X representing the low, medium, and high risk respectively, hence

$$\begin{aligned} LR &= \{x, \mu_{LR}(x)\} \qquad \mu_{LR}(x) : X \to [0,1] \\ MR &= \{x, \mu_{MR}(x)\} \qquad \mu_{MR}(x) : X \to [0,1] \\ HR &= \{x, \mu_{HR}(x)\} \qquad \mu_{HR}(x) : X \to [0,1] \end{aligned} \tag{5.7}$$

with

$$\begin{aligned} \mu_{LR}(x) &= \begin{cases} 1-2x & for \quad 0 \le x \le 0{,}5 \\ 0 & otherwise \end{cases} \\ \mu_{MR}(x) &= \begin{cases} 2x & for \quad 0 \le x \le 0{,}5 \\ 2-2x & for \quad 0{,}5 \le x \le 1 \end{cases} \\ \mu_{HR}(x) &= \begin{cases} 2x-1 & for \quad 0{,}5 \le x \le 1 \\ 0 & otherwise \end{cases} \end{aligned} \tag{5.8}$$

And finally the total risk of the external parties can be computed as follows.

$$\begin{aligned} Risk(x) &= \mu_{LR}(x) \,\mathrm{Y}\, \mu_{MR}(x) \,\mathrm{Y}\, \mu_{HR}(x) \\ &= \max\left[\mu_{LR}(x), \mu_{MR}(x), \mu_{HR}(x)\right] \end{aligned} \tag{5.9}$$

The value of total risk determine which security level shall be assigned to the external party originating the packet of interest. Membership function of the risk is depicted in Figure 5.6. To implement the total risk of a transaction into the security rules of firewall, the following computation is executed:

$$\forall o_m \in O, Risk(x) = \begin{cases} LR: \; o_m(x) \rightarrow O_{j-1} \\ \mathrm{MR}: o_m(x) \rightarrow O_j \\ \mathrm{HR}: o_m(x) \rightarrow O_{j+1} \end{cases} \tag{5.10}$$

with $o_m \in O$ denote a particular external party involved in the transaction, and j is the index of trusted level of external party as defined in Section 4.4.2.

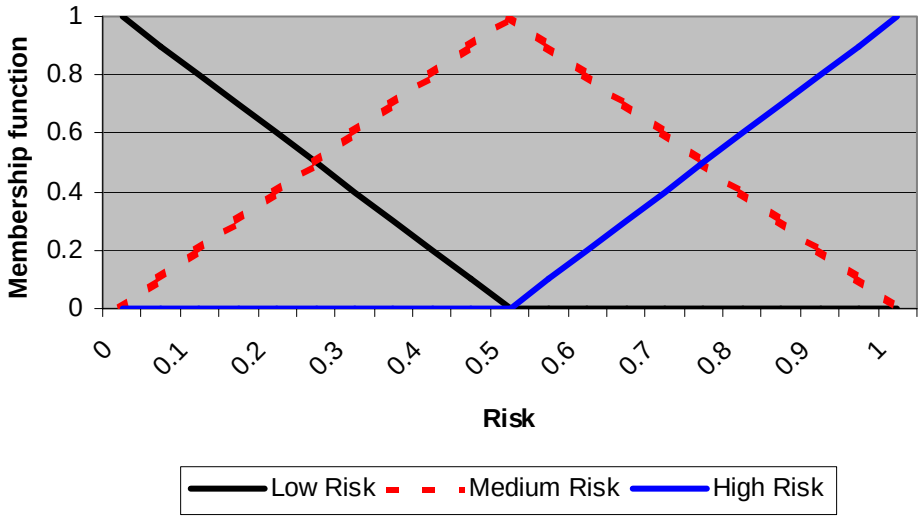


Figure 5.6: Membership function of the risk

5.5 Implementation

Implementation of the proposed security policy update using fuzzy reasoning was conducted by developing algorithm to assign security level to external parties, as depicted in Figure 5.7. The algorithm consists of two main processes i.e. the background process to collect the flowing packets that are stored in the traffic buffer, and the foreground process to analyse the content of the buffer and to compute the security level assign for each external parties. In order to collect the traffic, *ngrep* network monitor is used to capture and store the flowing packet passing through the firewall. The mechanism for collecting packet here is to fill up the traffic buffer with network packet during the running time of the foreground process, and then empty back the buffer when its content have been supplied to the foreground process for running in the next cycle. Result of this program is the security level assigned to each external party, which the security levels identify the risk carried by accessing those parties. Referring to the formulation in Section 5.4, here three security levels are defined namely high risk, medium risk, and low risk. This implementation also follows the risk membership function as depicted in Figure 5.6.

In the runtime process of active firewall, the developed security policy update based on fuzzy reasoning is run in the background although its algorithm consists of the foreground and background process. It is due to the processing intensive of this method, particularly in analysing the content of traffic buffer. Execution to update security policy in the active firewall is held next after completing each cycle of foreground process. Complete implementation of this method is given in Appendix B.

```
Procedure Fuzzy Reasoning
'TBUF  : array of traffic buffer
Begin
   Forced Info = {"exit=true", "exit=false","checkexit"}
   Adv = {"window.open","winopen (","popup ("}
   i = 1
   While i ≤ TBUF.size do
   Begin
        Line_content = ""
         j = 1
        Do Until (TBUF(i) = EOL) Or (i = TBUF.size)
           Line_content = Line_content + TBUF(i)
           If (TBUF(i) = " ") Then
           Begin
                Word(j) = Line_content
                Line_content = ""
                 j = j + 1
           End
           i = i + 1
        Loop
        For  j = 1  to Word.size do
        Begin
           If External_party = External(Word(j)) And External_party is new Then
                External _ party ⇒ Array _ Outsider
           If Internal_user = Internal(Word(j)) And Internal_user is new Then
                Internal _ user ⇒ Array _ Outsider.Internal
           If Word(j) = EXE File Then Array_Outsider.EXE = Array_Outsider.EXE + 1
           If Word(j) = Forced Info Then
                Array_Outsider.Forced Info = Array_Outsider.Forced Info + 1
           If Word(j) = Adv Then Array_Outsider.Adv = Array_Outsider.Adv + 1
        End
   End
   For  j = 1  to Array_Outsider.size do
   Begin
        For  k = 1  to Array_Outsider.size do
        Begin
           If j <> k Then
                If Array_Outsider(j).Internal = Array_Outsider(k).Internal Then
                   If (Array_Outsider(j).Time - Array_Outsider(k).Time) < 15 Then
                        Array_Outsider(j).Machine = Array_Outsider(j).Machine+1
           Risk_EXE = ComputeRiskEXE(Array_Outsider(j).EXE)
           Risk_ForcedInfo = ComputeRiskForcedInfo(Array_Outsider(j). Forced Info)
           Risk_Adv = ComputeRiskAdv(Array_Outsider(j).Adv)
           Risk_Machine = ComputeRiskMachine(Array_Outsider(j).Machine)
           Total_Risk = max[Risk_EXE, Risk_Machine, Risk_Script]
           Select Case Total_Risk
                Total_Risk ≤ 2.5 : Array_Outsider(k) ⇒ Low_Risk
                2.5 < Total_Risk ≤7.5 : Array_Outsider(k) ⇒ Medium_Risk
                Total_Risk > 2.5 : Array_Outsider(k) ⇒ High_Risk
           End Case
        End
   End
End
```

Figure 5.7: Algorithm to assign security level to external parties

5.6 Experiment

Experiments to evaluate the proposed fuzzy reasoning method were held on the same platform as used in Chapter 4. Some collections of network packet were used as the input of the system. These packets were captured from different sources of network transaction as listed in the first column of Table 5.2. Observations of the collected packets using fuzzy reasoning as formulated in Section 5.4 produce the result as presented in Table 5.3 and Table 5.4. These data disclose the aspects of the proposed method i.e. accuracy and processing time. The former parameter corresponds to the performance of fuzzy reasoning in identifying the risk of external parties, while the latter disclose the speed of this method to respond to the malicious external parties. Descriptions of each parameter follow.

5.6.1 Accuracy

Here accuracy is analysed using two well-known parameters i.e. false acceptance rate and false rejection rate. False acceptance refers to the error of accepting malicious party, while false rejection is the error of rejecting the secure or normal outsiders. Both parameters are computed by comparing the data of experimental result in Table 5.3 to the reference of the external parties in term of the list of safety properties as shown in Table 5.2. Here the reference is obtained by holding manual in depth analysis on the global content of each external party, combined with the observation on the reputation of each party. The last two properties i.e. the global content and the reputation, influence the selection for using popular external parties as the object of observation, since it facilitates the creation of the reference.

Table 5.2: List of reference

No	Name of External Party	Risk	Comment
1	www.17tahun.com	Threat	Pornography
2	www.acm.org	Normal	-
3	www.altavista.com	Normal	-
4	www.astro.my	Normal	-
5	www.babes.com	Threat	Pornography
6	www.cisco.com	Normal	-
7	www.cnn.com	Normal	-
8	www.elsevier.com	Normal	-
9	www.gatra.com	Normal	-
10	www.gawab.com	Normal	-
11	www.go.com	Normal	-
12	www.hotmail.com	Normal	-
13	www.hunsa.com	Threat	Web Portal
14	www.ieee.com	Normal	-
15	www.jaring.my	Normal	-
16	www.kompas.com	Normal	-
17	www.lycos.com	Normal	-
18	www.melayu.com	Normal	-
19	www.mfa.go.th	Normal	-
20	www.mtreexxx.net	Threat	Pornography
21	www.porn.com	Threat	Pornography
22	www.sanook.com	Threat	Web Portal
23	www.sciencedirect.com	Normal	-
24	www.sex.com	Threat	Pornography
25	www.thaiamateur.net	Threat	Pornography
26	www.thaigirls.net	Threat	Pornography
27	www.thestar.com	Normal	-
28	www.ukm.my	Normal	-
29	www.utm.my	Normal	-
30	www.xxx.com	Threat	Pornography

Referring to the experimental result and the reference, from ten external parties that are considered dangerous by the reference, two are accepted by fuzzy reasoning method. While from twenty external parties that are accepted as normal by reference, nine of them are considered malicious by the developed fuzzy reasoning. This distinction show the different point of view in evaluating the risk factor of the external parties, which the reference uses high level consideration, while the proposed fuzzy reasoning method conducts low level evaluation based on the predetermined parameters influencing threat as formulated in Section 4. The experiment delivers 20% false acceptance and 45% false rejection. The false

acceptance here is due to malicious content of some public web sites such as intense advertisements and too many external servers include in the transactions, while the false rejection is due to polite interface of malicious websites before it launch its action. Graph presentation of the experimental results for the normal parties as justified by the reference is shown in Figure 5.8, and for the parties carrying threats in Figure 5.9.

Table 5.3: Experimental result (risk measurement)

Name of External Party	***E***	***F***	***A***	***M***	***Risk***	**Status**
www.17tahun.com	0	0	3	10	0.846	High Risk
www.acm.org	0	0	0	3	0.646	Medium Risk
www.altavista.com	0	0	0	4	0.711	Medium Risk
www.astro.my	0	0	1	2	0.5	Medium Risk
www.babes.com	0	0	2	4	0.711	Medium Risk
www.cisco.com	0	0	15	4	0.990	High Risk
www.cnn.com	0	0	11	9	0.980	High Risk
www.elsevier.com	0	0	2	4	0.711	Medium Risk
www.gatra.com	0	0	0	3	0.646	Medium Risk
www.gawab.com	0	0	1	3	0.646	Medium Risk
www.go.com	0	0	4	8	0.905	High Risk
www.hotmail.com	0	0	6	12	0.952	High Risk
www.hunsa.com	1	0	3	7	1	High Risk
www.ieee.com	0	0	1	7	0.796	High Risk
www.jaring.my	0	0	2	2	0.692	Medium Risk
www.kompas.com	0	0	4	5	0.905	High Risk
www.lycos.com	0	0	1	11	0.842	High Risk
www.melayu.com	0	0	0	2	0.5	Medium Risk
www.mfa.go.th	0	0	0	3	0.646	Medium Risk
www.mtreexxx.net	167	0	30	6	1	High Risk
www.porn.com	4	90	9	6	1	High Risk
www.sanook.com	0	0	8	7	0.935	High Risk
www.sciencedirect.com	0	0	7	2	0.963	High Risk
www.sex.com	0	4	3	2	1	High Risk
www.thaiamateur.net	0	0	0	5	0.75	High Risk
www.thaigirls.net	0	0	0	6	0.776	High Risk
www.thestar.com	0	0	18	11	0.993	High Risk
www.ukm.my	0	0	0	2	0.5	Medium Risk
www.utm.my	0	0	0	3	0.646	Medium Risk
www.xxx.com	0	0	2	4	0.711	Medium Risk

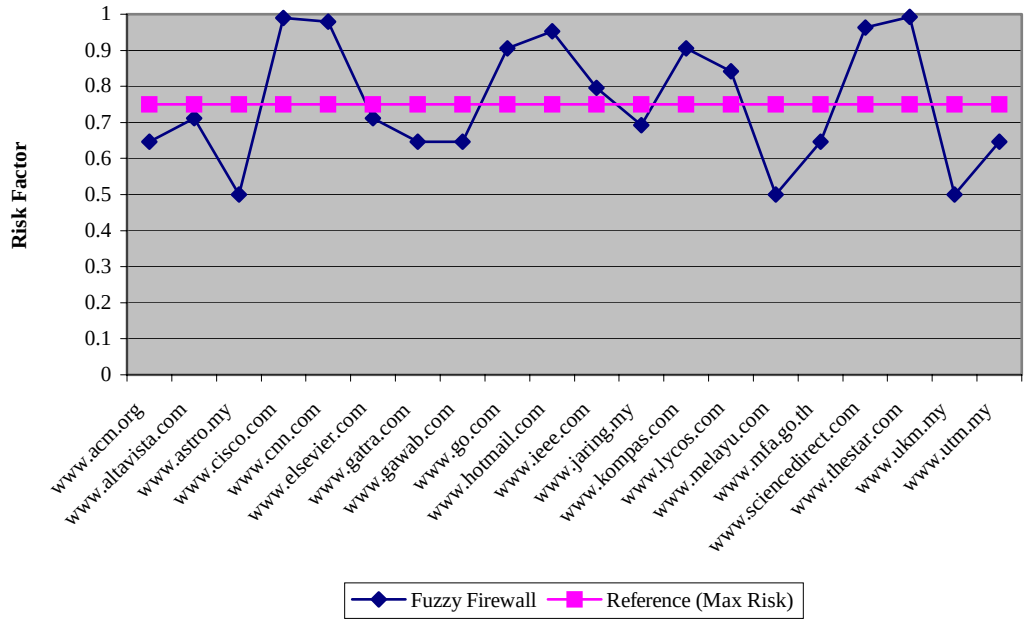


Figure 5.8: Experimental results for normal external parties

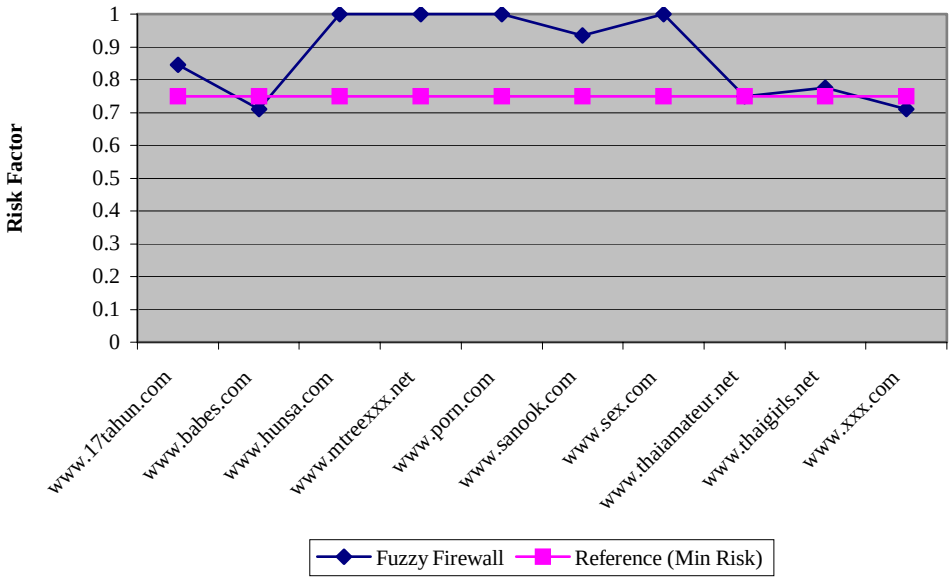


Figure 5.9: Experimental results for threat external parties

Table 5.4: Experimental result (processing time)

Name of External Party	*Size* (Bytes)	$\Delta Time$	$\frac{Size}{Min\ Size}$	$\frac{\Delta Time}{Min\ \Delta Time}$
www.altavista.com	28148	0:00:09	1.00	1.00
www.babes.com	86024	0:01:26	3.06	9.56
www.hotmail.com	100777	0:01:53	3.58	12.56
www.thaiamateur.net	104841	0:02:11	3.72	14.56
www.sex.com	109678	0:02:16	3.90	15.11
www.sciencedirect.com	121757	0:02:44	4.33	18.22
www.utm.my	129005	0:03:27	4.58	23.00
www.xxx.com	129985	0:03:10	4.62	21.11
www.acm.org	137546	0:03:35	4.89	23.89
www.jaring.my	149473	0:04:06	5.31	27.33
www.gawab.com	153902	0:04:28	5.47	29.78
www.gatra.com	165407	0:05:19	5.88	35.44
www.astro.my	165775	0:05:01	5.89	33.44
www.lycos.com	214409	0:08:56	7.62	59.56
www.thaigirls.net	225843	0:10:08	8.02	67.56
www.ieee.com	259714	0:14:22	9.23	95.78
www.cnn.com	274236	0:13:59	9.74	93.22
www.cisco.com	277383	0:08:14	9.85	54.89
www.ukm.my	307656	0:18:24	10.93	122.67
www.melayu.com	324279	0:20:44	11.52	138.22
www.17tahun.com	335085	0:22:42	11.90	151.33
www.elsevier.com	338587	0:24:02	12.03	160.22
www.thestar.com	344841	0:22:22	12.25	149.11
www.mfa.go.th	369902	0:25:08	13.14	167.56
www.go.com	378983	0:26:50	13.46	178.89
www.detik.com	460883	0:43:10	16.37	287.78
www.kompas.com	499875	0:50:31	17.76	336.78
www.porn.com	566070	1:03:57	20.11	426.33
www.sanook.com	628835	1:30:31	22.34	603.44
www.hunsa.com	873838	2:41:50	31.04	1078.89
www.mtreexxx.net	1082247	2:14:17	38.45	895.22

5.6.2 Processing Time

Referring to the experimental result presented in Table 5.4, the processing time of the proposed fuzzy reasoning method increase exponentially following the growth of the buffer size, which the buffer is used to store the collection of network

packets. And if we look back on the proposed algorithm as shown in Figure 5.7, it shows that the iterations to browse and analyse the contents of the buffer become the cause of this condition. Bigger size of buffer would create more iterations, thus longer time is required to finish the process. Here using the reference of the processing time and size buffer that are denoted by Δt_{ref} and s_{ref}, the processing time of the known buffer size can be computed as follows

$$\Delta t = \left(\frac{s}{s_{ref}} \right)^2 \Delta t_{ref} + c \qquad (5.11)$$

with s denote the size of the buffer, and c refers to the time consumption to compute other part of the algorithm in addition to the required time to complete the main iterations. Figure 5.10 shows the empirical measurements of the processing time against the size of buffer for storing network packets. These data shows that predicted processing time using Equation (5.11) is satisfied.

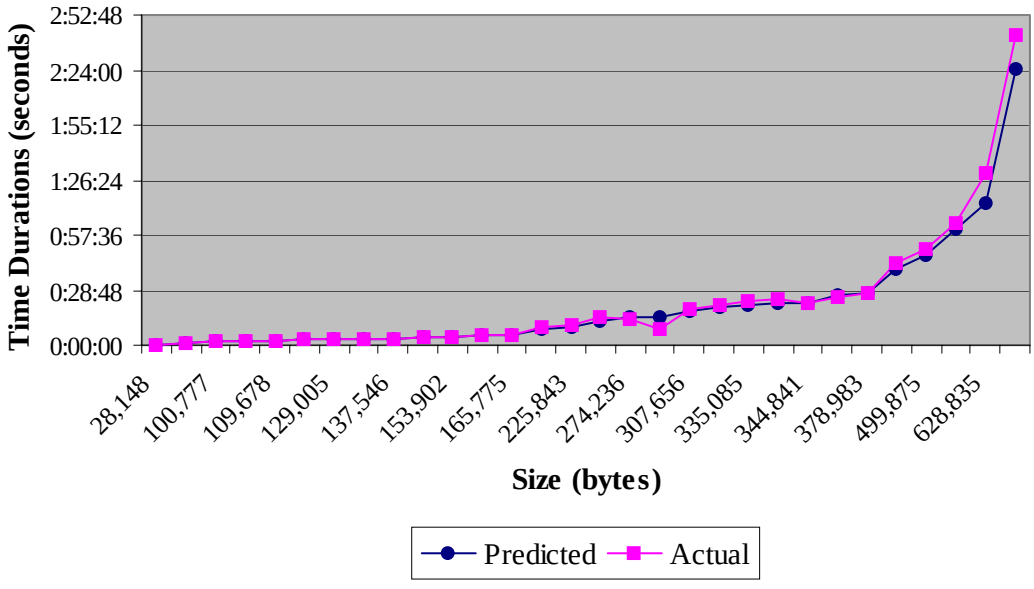


Figure 5.10: The processing time against the buffer size

5.6.3 Sensitivity

Sensitivity is defined as the capability of firewall to react on detecting a threat by executing an action to secure internal users. This parameter is obtained by measuring time distinction between the detection of the threat to the modification of canals corresponding to the malicious external parties. Assuming that the worst-case condition is met i.e. the threat is detected in the beginning of traffic buffer, hence firewall must wait to complete its observation on analysing the packet in the traffic buffer before modifying the canals. In this case, the required time to take an action against the threat is equal to the processing time to browse the buffer. Hence the quadratic increase of processing time as defined in Equation (5.11) is fulfilled.

5.7 Summary

This chapter discusses a method to adaptively updating security rules using fuzzy reasoning. The proposed method uses some parameters to measure the risk of external parties i.e. the existence of executable file, the number of forced information, the number of advertisements, the number of external machines, and the number of cookies. Experiment delivers 20% of false acceptance and 45% of false rejection, with the quadratic increase of processing time following the increase of buffer size.

CHAPTER 5

RUNTIME PROCESS: MINIMIZING THE INTERACTION BETWEEN UNPROTECTED USERS AND UNTRUSTED EXTERNAL PARTIES

5.1 Introduction

This chapter presents the development of runtime process for minimizing the interaction between unprotected users and untrusted external parties as defined in the security strategy in Chapter 3. The methods proposed in this stage are to follow initialisation process presented in Chapter 4. With *t* represent the current time and t_i denote the ending of initialisation process, the operations of firewall at $t > t_i$ are discussed. To carry on this task, a mechanism to identify which users are unprotected and which external parties are untrusted, is developed as well as the mechanism to measure the degree of risk produced from each Internet transaction. Since this method requires quantifying the abstract parameters such as safety and risk of an Internet transaction, thus fuzzy reasoning is employed to handle this task. Referring to the work of (Negnevitsky, 2002) that shows successful application of fuzzy reasoning for dealing with abstract parameter, and Labuschagne and Eloff (1998) and Kim *et al.* (2004) that show the applicability of fuzzy logic in handling real-time analysis on network traffic, fuzzy reasoning is utilized to observe the content of network traffic passing through the firewall. The developed strategy is to define some parameters indicating or causing the appearance of the threats and to formulate their membership function contributing to the risk for accessing Internet. The

advantage of this method is due to continues observation of network traffic, therefore any arising threats originated from Internet transaction in any time can be detected, thus the active firewall can take an action to secure internal network.

5.2 Review of Fuzzy Logic in Network Security

Since its introduction by Zadeh (1965), fuzzy logic has been applied to many systems with the purpose is to inject some degree of intelligence (Klir and Yuan, 1995; Yen *et al.*, 1995). Today, the implementations of fuzzy logic can be found in many disciplines such as finance sector, traffic control and automobile, information system, to medical science. In the domain of network security, fuzzy logic has successfully presented methods to describe the parameters that are difficult to quantify such as the safety and the risk of the network. The work of Ru and Eloff (1996) notably introduced the risk analysis method using fuzzy logic, and was followed by Labuschagne and Eloff (1998) to develop real time risk analysis for securing network. The last effort aims to support firewall to perform identification and authentication on specific users by measuring the global risk of every transaction. Besides observing the header of IP and TCP packet, no traffic content are taken into account by this method, therefore only shallow analysis can be held. Meanwhile, the work of Zou *et al.* (2002) for developing fuzzy adaptive security algorithm for intelligent firewall, consider only the membership function of the security level that are manually determined based on the source and destination of the packet. Although the proposed method is claimed resolving the conflict between security and speed, however the approach applied in the developed algorithm simplifies the analysis of the flowing packet using the predetermined security level. Therefore the algorithm has minimal capability to identify the threat on the flowing traffic. This condition also leads to the dependency to manually set the security levels. Moreover, it seems inapplicable to manually assign security level to every external party.

The effort of Guan *et al.* (2004) to develop intrusion detection systems and Kim *et al.* (2004) to develop network forensic afford to employ fuzzy reasoning to

conduct thorough analysis on network traffic in order to determine the pattern of network intrusion. However the proposed methods consider only the header of network packet. Thus it will be difficult to identify the threat contained in the flowing data. In this research, fuzzy reasoning is used to support the mechanism of active firewall to determine the security level of the flowing packet. Unlike the works of Labuschagne and Eloff (1998) that rely only on the packet header or Zou *et al.* (2002) that manually determine the security level of the packet, here automatic threat analysis is held on the content of the flowing packet. Based on this analysis, security policy of the firewall is adaptively updated during the runtime process. Formulation of this method is described in the next section.

5.3 Network Traffic Analysis

Preliminary works to manually analyse the content of network traffic were held to disclose the factors influencing Internet threat. This effort observed some parameters that might indicate the appearance of the threats as follow, the existence of executable file (*E*), the effort to force user for reading the external information (*F*), the number of advertisements (*A*), the number of external servers involved in a single Internet transaction (*M*), the number of active script (*S*) and the number of cookies (*C*). To verify the influence of these parameters for causing the risk of accessing Internet, a set of network traffics generated from some well-known Internet parties were collected and scrutinized. Here the usage of well-known Internet parties is important in order to facilitate the justification of the safety of accessing Internet i.e. to determine whether the external party is a normal party or causing threat. Thus, the above parameters that lead to the threat could be identified. Results of the observation are presented in Table 5.1.

By proportioning the mean υ of the existence of each predetermined parameter of normal parties against the threat parties as shown in the last row of Table 5.1, and also following the rule of influence presented in Negnevitsky (2002), data presented in Table 5.1 show some phenomenon as follow. The existence of executable file and forced information carried by the external parties absolutely

causes the threat to the internal network users. The number of advertisements slightly indicates the threat, and the numbers of external servers has less influence to the appearance of threat. Meanwhile the number of cookies and active scripts cannot be used to show the appearance of Internet threat since the proportion of the mean of these parameters from normal and threat parties produce the values close to or greater than one. It means these parameters are indifferent both in normal and threat parties, it is shown by large numbers of active scripts and cookies can be found in both parties. These phenomenons are obviously presented in Figure 5.1 below. Therefore, in this research only first four-listed parameters above are used in analysing Internet threats, since the number of cookies and active scripts cannot be employed to detect the Internet threats.

Table 5.1: Measuring the factors influencing the threat

Name of External Party	**Status**	***E***	***F***	***A***	***M***	***C***	***S***
www.ukm.my	safe	0	0	0	0	2	2
www.acm.org	safe	0	0	0	0	3	1
www.utm.my	safe	0	0	0	0	3	0
www.mfa.go.th	safe	0	0	0	0	3	9
www.elsevier.com	safe	0	0	2	1	4	43
www.ieee.com	safe	0	0	1	3	7	15
www.xxx.com	threat	0	0	2	4	4	2
www.babes.com	threat	0	0	2	7	4	6
www.sex.com	threat	0	4	3	9	2	0
www.porn.com	threat	4	90	9	3	6	33
υ of safe parties		0	0	1.5	2	3.67	11.67
υ of threat parties		1	23.5	4	5.75	4	10.25
υ of safe / υ of threat		0	0	0.375	0.348	0.918	1.139

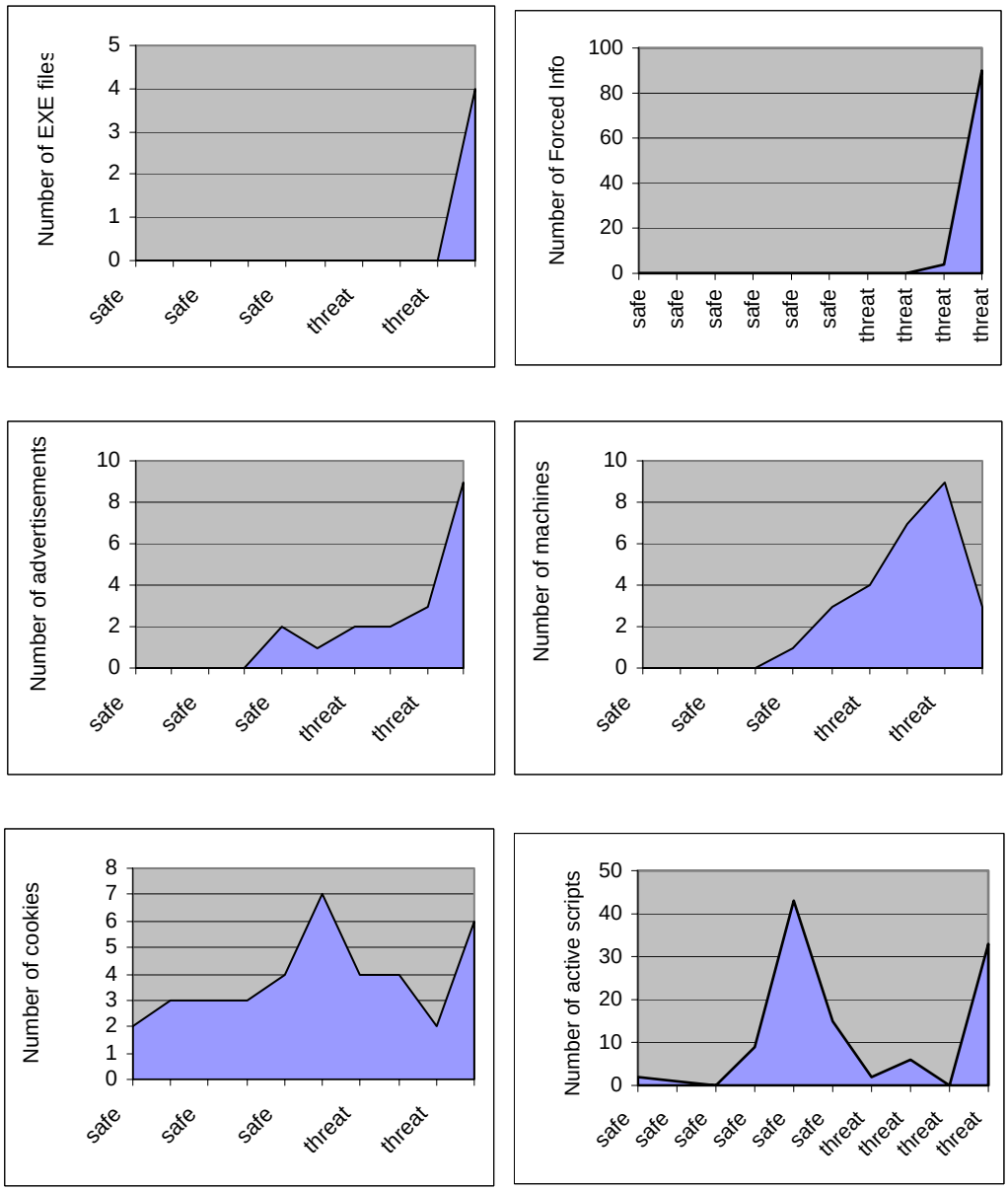


Figure 5.1: Effects of some defined parameters to indicate threats

5.4 Formulating Security Rules Update using Fuzzy Reasoning

Let *Y* be the universe of discourse with elements *y* that denotes the threat of the network traffic passing through the firewall. And let *E*, *F*, *A*, and *M* be the sets of components representing the existence of the executable file, the effort to force internal user to read the information from external parties, the number of advertisements, and the number of the external server involved in a single Internet

transaction respectively, hence fuzzy set *E*, *F*, *A*, and *M* of universe *Y* are defined by the functions $\mu_E(y)$, $\mu_F(y)$, $\mu_A(y)$, and $\mu_M(y)$. Those functions are called the membership function of set *E*, *F*, *A*, and *M* respectively, and are described as follow.

$$\begin{aligned} E &= \{y, \mu_E(y)\} \quad y \in Y, \mu_E(y) : Y \to [0,1] \\ F &= \{y, \mu_F(y)\} \quad y \in Y, \mu_F(y) : Y \to [0,1] \\ A &= \{y, \mu_A(y)\} \quad y \in Y, \mu_A(y) : Y \to [0,1] \\ M &= \{y, \mu_M(y)\} \quad y \in Y, \mu_M(y) : Y \to [0,1] \end{aligned} \tag{5.1}$$

with

$$\mu_E(y) = \begin{cases} 1 \Leftrightarrow E = 0 \\ 0 \Leftrightarrow E > 0 \end{cases} \tag{5.2}$$

$$\mu_F(y) = \begin{cases} 1 \Leftrightarrow F = 0 \\ 0 \Leftrightarrow F > 0 \end{cases} \tag{5.3}$$

$$\mu_A(y) = 1 - \frac{1}{A^{1.7}} \tag{5.4}$$

$$\mu_M(y) = 1 - \frac{1}{2\sqrt{M}} \tag{5.5}$$

The graphs visualizing the membership function of the parameters defined in Equation (5.2) to (5.5), are depicted in Figure 5.2 to 5.5. These graphs become the input of the fuzzy-based security rules update to determine the security level of every external party.

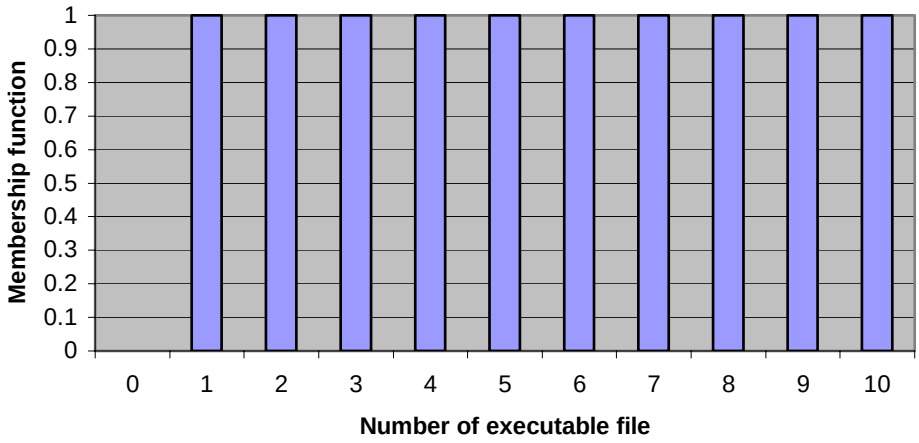


Figure 5.2: The membership function of executable files

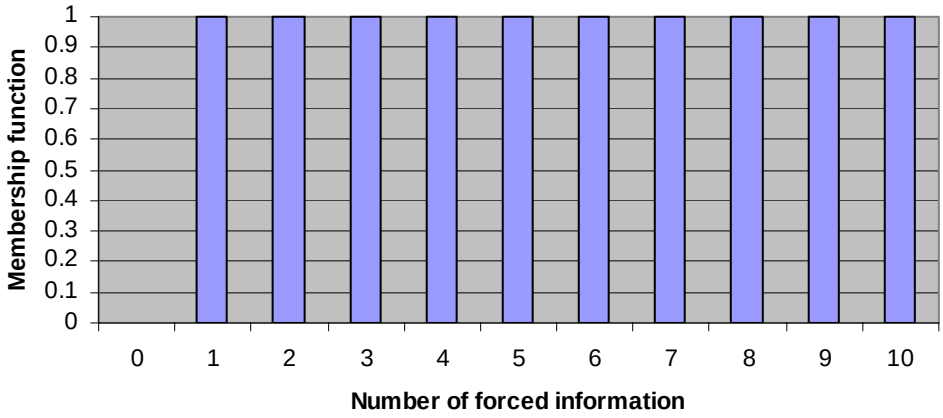


Figure 5.3: The membership function of forced information

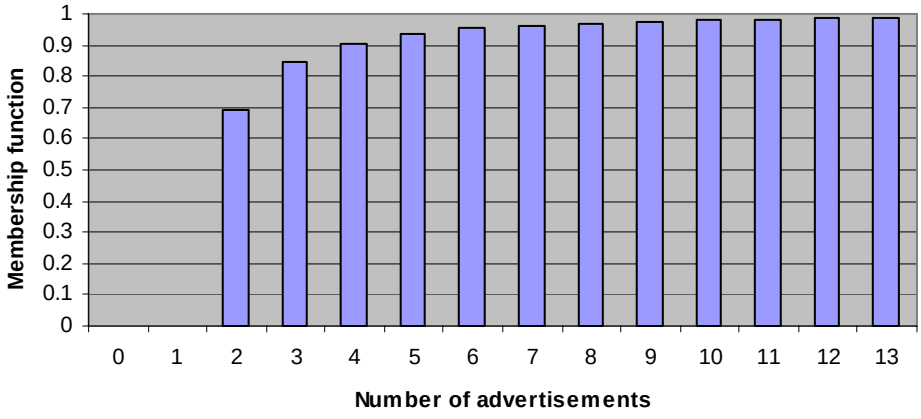


Figure 5.4: The membership function of the number of advertisements

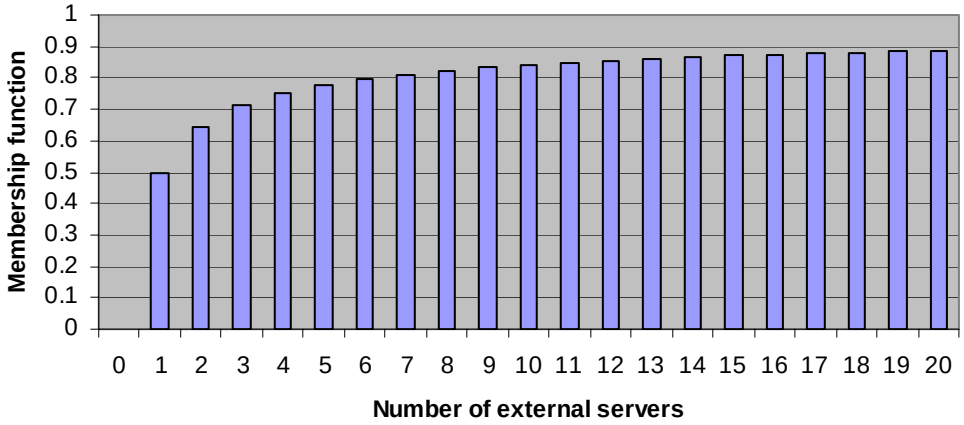


Figure 5.5: The membership function of the number of external machines

To obtain the total threat produced by the membership function of the factors defined above, the equation below is computed.

$$\begin{aligned} x &= \mu_E(y) \,\mathrm{Y}\, \mu_F(y) \,\mathrm{Y}\, \mu_A(y) \,\mathrm{Y}\, \mu_M(y) \\ &= \max\left[\mu_E(y), \mu_F(y), \mu_A(y), \mu_M(y)\right] \end{aligned} \tag{5.6}$$

with x denote the total threat. Let X becomes the universe of discourse of the threat in which $x \in X$. And let *LR*, *MR*, and *HR* become the subsets of X representing the low, medium, and high risk respectively, hence

$$\begin{aligned} LR &= \{x, \mu_{LR}(x)\} \qquad \mu_{LR}(x) : X \to [0,1] \\ MR &= \{x, \mu_{MR}(x)\} \qquad \mu_{MR}(x) : X \to [0,1] \\ HR &= \{x, \mu_{HR}(x)\} \qquad \mu_{HR}(x) : X \to [0,1] \end{aligned} \tag{5.7}$$

with

$$\begin{aligned} \mu_{LR}(x) &= \begin{cases} 1-2x & for \quad 0 \le x \le 0{,}5 \\ 0 & otherwise \end{cases} \\ \mu_{MR}(x) &= \begin{cases} 2x & for \quad 0 \le x \le 0{,}5 \\ 2-2x & for \quad 0{,}5 \le x \le 1 \end{cases} \\ \mu_{HR}(x) &= \begin{cases} 2x-1 & for \quad 0{,}5 \le x \le 1 \\ 0 & otherwise \end{cases} \end{aligned} \tag{5.8}$$

And finally the total risk of the external parties can be computed as follows.

$$\begin{aligned} Risk(x) &= \mu_{LR}(x) \,\mathrm{Y}\, \mu_{MR}(x) \,\mathrm{Y}\, \mu_{HR}(x) \\ &= \max\left[\mu_{LR}(x), \mu_{MR}(x), \mu_{HR}(x)\right] \end{aligned} \tag{5.9}$$

The value of total risk determine which security level shall be assigned to the external party originating the packet of interest. Membership function of the risk is depicted in Figure 5.6. To implement the total risk of a transaction into the security rules of firewall, the following computation is executed:

$$\forall o_m \in O, Risk(x) = \begin{cases} LR: \; o_m(x) \rightarrow O_{j-1} \\ \mathrm{MR}: o_m(x) \rightarrow O_j \\ \mathrm{HR}: o_m(x) \rightarrow O_{j+1} \end{cases} \tag{5.10}$$

with $o_m \in O$ denote a particular external party involved in the transaction, and j is the index of trusted level of external party as defined in Section 4.4.2.

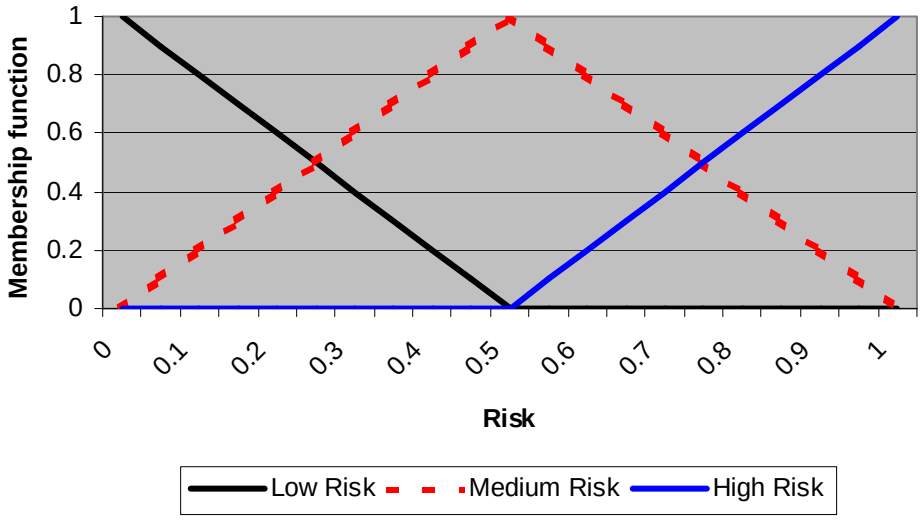


Figure 5.6: Membership function of the risk

5.5 Implementation

Implementation of the proposed security policy update using fuzzy reasoning was conducted by developing algorithm to assign security level to external parties, as depicted in Figure 5.7. The algorithm consists of two main processes i.e. the background process to collect the flowing packets that are stored in the traffic buffer, and the foreground process to analyse the content of the buffer and to compute the security level assign for each external parties. In order to collect the traffic, *ngrep* network monitor is used to capture and store the flowing packet passing through the firewall. The mechanism for collecting packet here is to fill up the traffic buffer with network packet during the running time of the foreground process, and then empty back the buffer when its content have been supplied to the foreground process for running in the next cycle. Result of this program is the security level assigned to each external party, which the security levels identify the risk carried by accessing those parties. Referring to the formulation in Section 5.4, here three security levels are defined namely high risk, medium risk, and low risk. This implementation also follows the risk membership function as depicted in Figure 5.6.

In the runtime process of active firewall, the developed security policy update based on fuzzy reasoning is run in the background although its algorithm consists of the foreground and background process. It is due to the processing intensive of this method, particularly in analysing the content of traffic buffer. Execution to update security policy in the active firewall is held next after completing each cycle of foreground process. Complete implementation of this method is given in Appendix B.

```
Procedure Fuzzy Reasoning
'TBUF  : array of traffic buffer
Begin
   Forced Info = {"exit=true", "exit=false","checkexit"}
   Adv = {"window.open","winopen (","popup ("}
   i = 1
   While i ≤ TBUF.size do
   Begin
        Line_content = ""
         j = 1
        Do Until (TBUF(i) = EOL) Or (i = TBUF.size)
           Line_content = Line_content + TBUF(i)
           If (TBUF(i) = " ") Then
           Begin
                Word(j) = Line_content
                Line_content = ""
                 j = j + 1
           End
           i = i + 1
        Loop
        For  j = 1  to Word.size do
        Begin
           If External_party = External(Word(j)) And External_party is new Then
                External _ party ⇒ Array _ Outsider
           If Internal_user = Internal(Word(j)) And Internal_user is new Then
                Internal _ user ⇒ Array _ Outsider.Internal
           If Word(j) = EXE File Then Array_Outsider.EXE = Array_Outsider.EXE + 1
           If Word(j) = Forced Info Then
                Array_Outsider.Forced Info = Array_Outsider.Forced Info + 1
           If Word(j) = Adv Then Array_Outsider.Adv = Array_Outsider.Adv + 1
        End
   End
   For  j = 1  to Array_Outsider.size do
   Begin
        For  k = 1  to Array_Outsider.size do
        Begin
           If j <> k Then
                If Array_Outsider(j).Internal = Array_Outsider(k).Internal Then
                   If (Array_Outsider(j).Time - Array_Outsider(k).Time) < 15 Then
                        Array_Outsider(j).Machine = Array_Outsider(j).Machine+1
           Risk_EXE = ComputeRiskEXE(Array_Outsider(j).EXE)
           Risk_ForcedInfo = ComputeRiskForcedInfo(Array_Outsider(j). Forced Info)
           Risk_Adv = ComputeRiskAdv(Array_Outsider(j).Adv)
           Risk_Machine = ComputeRiskMachine(Array_Outsider(j).Machine)
           Total_Risk = max[Risk_EXE, Risk_Machine, Risk_Script]
           Select Case Total_Risk
                Total_Risk ≤ 2.5 : Array_Outsider(k) ⇒ Low_Risk
                2.5 < Total_Risk ≤7.5 : Array_Outsider(k) ⇒ Medium_Risk
                Total_Risk > 2.5 : Array_Outsider(k) ⇒ High_Risk
           End Case
        End
   End
End
```

Figure 5.7: Algorithm to assign security level to external parties

5.6 Experiment

Experiments to evaluate the proposed fuzzy reasoning method were held on the same platform as used in Chapter 4. Some collections of network packet were used as the input of the system. These packets were captured from different sources of network transaction as listed in the first column of Table 5.2. Observations of the collected packets using fuzzy reasoning as formulated in Section 5.4 produce the result as presented in Table 5.3 and Table 5.4. These data disclose the aspects of the proposed method i.e. accuracy and processing time. The former parameter corresponds to the performance of fuzzy reasoning in identifying the risk of external parties, while the latter disclose the speed of this method to respond to the malicious external parties. Descriptions of each parameter follow.

5.6.1 Accuracy

Here accuracy is analysed using two well-known parameters i.e. false acceptance rate and false rejection rate. False acceptance refers to the error of accepting malicious party, while false rejection is the error of rejecting the secure or normal outsiders. Both parameters are computed by comparing the data of experimental result in Table 5.3 to the reference of the external parties in term of the list of safety properties as shown in Table 5.2. Here the reference is obtained by holding manual in depth analysis on the global content of each external party, combined with the observation on the reputation of each party. The last two properties i.e. the global content and the reputation, influence the selection for using popular external parties as the object of observation, since it facilitates the creation of the reference.

Table 5.2: List of reference

No	Name of External Party	Risk	Comment
1	www.17tahun.com	Threat	Pornography
2	www.acm.org	Normal	-
3	www.altavista.com	Normal	-
4	www.astro.my	Normal	-
5	www.babes.com	Threat	Pornography
6	www.cisco.com	Normal	-
7	www.cnn.com	Normal	-
8	www.elsevier.com	Normal	-
9	www.gatra.com	Normal	-
10	www.gawab.com	Normal	-
11	www.go.com	Normal	-
12	www.hotmail.com	Normal	-
13	www.hunsa.com	Threat	Web Portal
14	www.ieee.com	Normal	-
15	www.jaring.my	Normal	-
16	www.kompas.com	Normal	-
17	www.lycos.com	Normal	-
18	www.melayu.com	Normal	-
19	www.mfa.go.th	Normal	-
20	www.mtreexxx.net	Threat	Pornography
21	www.porn.com	Threat	Pornography
22	www.sanook.com	Threat	Web Portal
23	www.sciencedirect.com	Normal	-
24	www.sex.com	Threat	Pornography
25	www.thaiamateur.net	Threat	Pornography
26	www.thaigirls.net	Threat	Pornography
27	www.thestar.com	Normal	-
28	www.ukm.my	Normal	-
29	www.utm.my	Normal	-
30	www.xxx.com	Threat	Pornography

Referring to the experimental result and the reference, from ten external parties that are considered dangerous by the reference, two are accepted by fuzzy reasoning method. While from twenty external parties that are accepted as normal by reference, nine of them are considered malicious by the developed fuzzy reasoning. This distinction show the different point of view in evaluating the risk factor of the external parties, which the reference uses high level consideration, while the proposed fuzzy reasoning method conducts low level evaluation based on the predetermined parameters influencing threat as formulated in Section 4. The experiment delivers 20% false acceptance and 45% false rejection. The false

acceptance here is due to malicious content of some public web sites such as intense advertisements and too many external servers include in the transactions, while the false rejection is due to polite interface of malicious websites before it launch its action. Graph presentation of the experimental results for the normal parties as justified by the reference is shown in Figure 5.8, and for the parties carrying threats in Figure 5.9.

Table 5.3: Experimental result (risk measurement)

Name of External Party	***E***	***F***	***A***	***M***	***Risk***	**Status**
www.17tahun.com	0	0	3	10	0.846	High Risk
www.acm.org	0	0	0	3	0.646	Medium Risk
www.altavista.com	0	0	0	4	0.711	Medium Risk
www.astro.my	0	0	1	2	0.5	Medium Risk
www.babes.com	0	0	2	4	0.711	Medium Risk
www.cisco.com	0	0	15	4	0.990	High Risk
www.cnn.com	0	0	11	9	0.980	High Risk
www.elsevier.com	0	0	2	4	0.711	Medium Risk
www.gatra.com	0	0	0	3	0.646	Medium Risk
www.gawab.com	0	0	1	3	0.646	Medium Risk
www.go.com	0	0	4	8	0.905	High Risk
www.hotmail.com	0	0	6	12	0.952	High Risk
www.hunsa.com	1	0	3	7	1	High Risk
www.ieee.com	0	0	1	7	0.796	High Risk
www.jaring.my	0	0	2	2	0.692	Medium Risk
www.kompas.com	0	0	4	5	0.905	High Risk
www.lycos.com	0	0	1	11	0.842	High Risk
www.melayu.com	0	0	0	2	0.5	Medium Risk
www.mfa.go.th	0	0	0	3	0.646	Medium Risk
www.mtreexxx.net	167	0	30	6	1	High Risk
www.porn.com	4	90	9	6	1	High Risk
www.sanook.com	0	0	8	7	0.935	High Risk
www.sciencedirect.com	0	0	7	2	0.963	High Risk
www.sex.com	0	4	3	2	1	High Risk
www.thaiamateur.net	0	0	0	5	0.75	High Risk
www.thaigirls.net	0	0	0	6	0.776	High Risk
www.thestar.com	0	0	18	11	0.993	High Risk
www.ukm.my	0	0	0	2	0.5	Medium Risk
www.utm.my	0	0	0	3	0.646	Medium Risk
www.xxx.com	0	0	2	4	0.711	Medium Risk

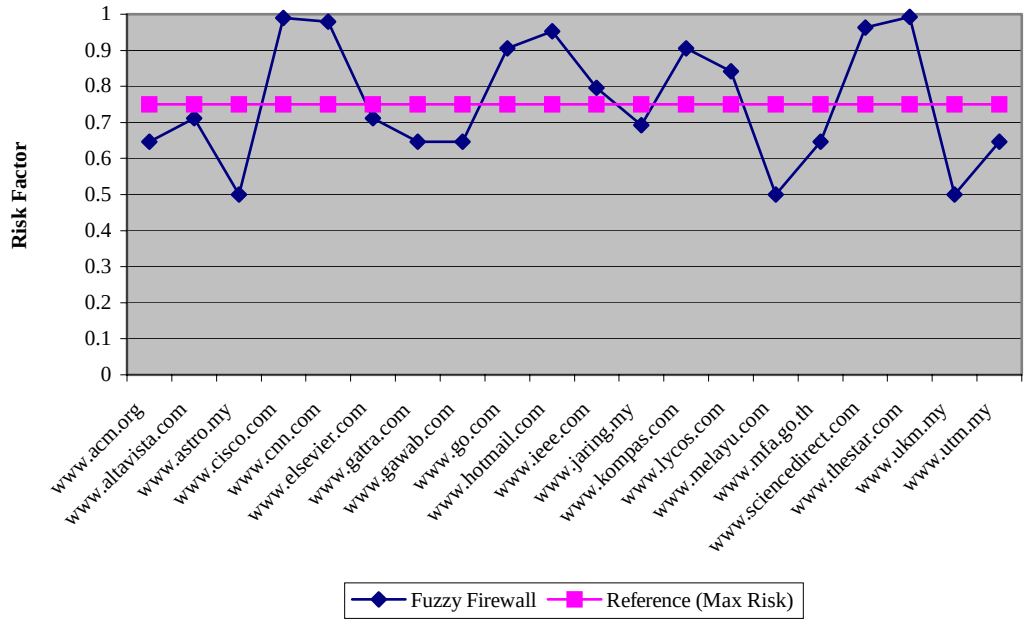


Figure 5.8: Experimental results for normal external parties

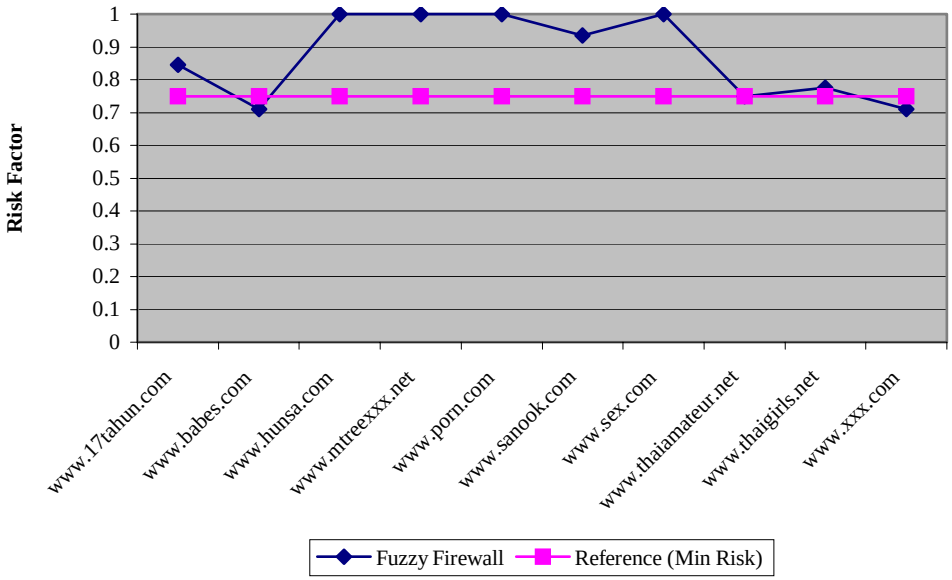


Figure 5.9: Experimental results for threat external parties

Table 5.4: Experimental result (processing time)

Name of External Party	*Size* (Bytes)	$\Delta Time$	$\frac{Size}{Min\ Size}$	$\frac{\Delta Time}{Min\ \Delta Time}$
www.altavista.com	28148	0:00:09	1.00	1.00
www.babes.com	86024	0:01:26	3.06	9.56
www.hotmail.com	100777	0:01:53	3.58	12.56
www.thaiamateur.net	104841	0:02:11	3.72	14.56
www.sex.com	109678	0:02:16	3.90	15.11
www.sciencedirect.com	121757	0:02:44	4.33	18.22
www.utm.my	129005	0:03:27	4.58	23.00
www.xxx.com	129985	0:03:10	4.62	21.11
www.acm.org	137546	0:03:35	4.89	23.89
www.jaring.my	149473	0:04:06	5.31	27.33
www.gawab.com	153902	0:04:28	5.47	29.78
www.gatra.com	165407	0:05:19	5.88	35.44
www.astro.my	165775	0:05:01	5.89	33.44
www.lycos.com	214409	0:08:56	7.62	59.56
www.thaigirls.net	225843	0:10:08	8.02	67.56
www.ieee.com	259714	0:14:22	9.23	95.78
www.cnn.com	274236	0:13:59	9.74	93.22
www.cisco.com	277383	0:08:14	9.85	54.89
www.ukm.my	307656	0:18:24	10.93	122.67
www.melayu.com	324279	0:20:44	11.52	138.22
www.17tahun.com	335085	0:22:42	11.90	151.33
www.elsevier.com	338587	0:24:02	12.03	160.22
www.thestar.com	344841	0:22:22	12.25	149.11
www.mfa.go.th	369902	0:25:08	13.14	167.56
www.go.com	378983	0:26:50	13.46	178.89
www.detik.com	460883	0:43:10	16.37	287.78
www.kompas.com	499875	0:50:31	17.76	336.78
www.porn.com	566070	1:03:57	20.11	426.33
www.sanook.com	628835	1:30:31	22.34	603.44
www.hunsa.com	873838	2:41:50	31.04	1078.89
www.mtreexxx.net	1082247	2:14:17	38.45	895.22

5.6.2 Processing Time

Referring to the experimental result presented in Table 5.4, the processing time of the proposed fuzzy reasoning method increase exponentially following the growth of the buffer size, which the buffer is used to store the collection of network

packets. And if we look back on the proposed algorithm as shown in Figure 5.7, it shows that the iterations to browse and analyse the contents of the buffer become the cause of this condition. Bigger size of buffer would create more iterations, thus longer time is required to finish the process. Here using the reference of the processing time and size buffer that are denoted by Δt_{ref} and s_{ref}, the processing time of the known buffer size can be computed as follows

$$\Delta t = \left(\frac{s}{s_{ref}} \right)^2 \Delta t_{ref} + c \qquad (5.11)$$

with s denote the size of the buffer, and c refers to the time consumption to compute other part of the algorithm in addition to the required time to complete the main iterations. Figure 5.10 shows the empirical measurements of the processing time against the size of buffer for storing network packets. These data shows that predicted processing time using Equation (5.11) is satisfied.

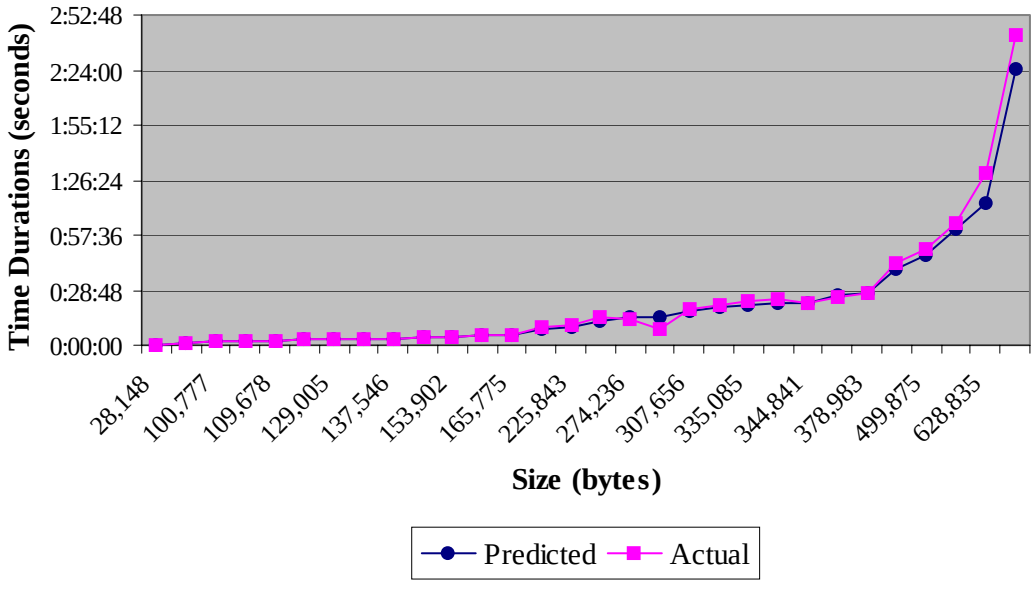


Figure 5.10: The processing time against the buffer size

5.6.3 Sensitivity

Sensitivity is defined as the capability of firewall to react on detecting a threat by executing an action to secure internal users. This parameter is obtained by measuring time distinction between the detection of the threat to the modification of canals corresponding to the malicious external parties. Assuming that the worst-case condition is met i.e. the threat is detected in the beginning of traffic buffer, hence firewall must wait to complete its observation on analysing the packet in the traffic buffer before modifying the canals. In this case, the required time to take an action against the threat is equal to the processing time to browse the buffer. Hence the quadratic increase of processing time as defined in Equation (5.11) is fulfilled.

5.7 Summary

This chapter discusses a method to adaptively updating security rules using fuzzy reasoning. The proposed method uses some parameters to measure the risk of external parties i.e. the existence of executable file, the number of forced information, the number of advertisements, the number of external machines, and the number of cookies. Experiment delivers 20% of false acceptance and 45% of false rejection, with the quadratic increase of processing time following the increase of buffer size.

CHAPTER 6

RUNTIME PROCESS:
MINIMIZING THE UNPROTECTED USERS

6.1 Introduction

With regard to security strategy defined in Chapter 3, this chapter presents the implementation of the strategy for minimizing the unprotected users. The developed method is still intended to handle the runtime process of active firewall. Hence the mechanism for $t > t_i$ is discussed. As mentioned in Chapter 3, it is more applicable to install a distributed detector in the intranet to watch the appearance of the threats, rather than having all security packages in every host of internal machines. Thus the developed method shall be capable to reduce the unprotected users by identifying and isolating the compromised internal machines. The choice spans from having intrusion detection, antivirus software, or vulnerability assessment. Since the purpose is to detect the compromised users, hence intrusion detection is considered more appropriated to be employed in this design. Referring to the work of Li *et al.* (2004) and Bernardes and Moreira (2000), which prove the applicability of agent software to guard the security of internal host, an agent-based system is developed with the purpose is to assist firewall in detecting malicious program running in the internal machines. This way firewall is expected to take a quick action to stop the unauthorized information flow that is potentially caused by the malicious program. Thus the victim machine can be isolated from getting access to or be exploited by the external party. To implement this mechanism, a distributed agent-based method is

developed. The advantage is due to the capability of distributed agent to cover an area that is considered too large to be handled by a single centralized system (Green *et al.*, 1997). Therefore the developed system is expected capable to monitor the running processes of all user machines in the intranet.

6.2 Review of Agent to Support Firewall

The notion of software agent has been around quite sometimes (White, 1996). Green *et al.* (1997) defines agent as an entity having fundamental properties of acting on behalf of others and enjoying a degree of autonomy. Agents also exhibit some level of proactivity and reactivity in its behaviour. Meanwhile Bernades and Moreira (2000) defines agent as a software program capable of executing a complex task on behalf of a user. A set of attributes, may equip the agent such as learning capabilities, cooperation, mobility, and adaptiveness (Green *et al.*, 1997; He and Leung, 2002). A number of advantages are offered by this technology such as capability to assist users in understanding a complex task and holding an ongoing execution for a relatively long period of time. In the domain of security, software agent have been used extensively to develop many types of security method such as security service (Shakshuki *et al.*, 2004), active security system (Zaki and Sobh, 2004), intrusion detection systems (Bernades and Moreira, 2000; Li *et al.*, 2004), network security management (Labiod and Boutapa, 2000) and micro firewalls (Hwang and Gangadharan, 2001). And due to its advantage to support distributed system, the use of software agent to build intrusion detection is widely accepted. However in the field of firewall, only few research efforts are reported formulating software agent to support the mechanism of firewall (Hwang and Gangadharan, 2001; Xian *et al.*, 2002). Mostly the agents in this field hold the function of intrusion detection system. These researches are inline with the suggestion of Davies (2000) to combine firewall and intrusion detection to have active and reactive security. However real-time operation is loosely considered in these efforts such as reported by Xian *et al.* (2002) that produce the empirical response time in the order of minute. Therefore the capability of firewall to stop an on going attack is questionable using this scenario. In

this research real-time operation of software agent is formulated to support active firewall. Detail descriptions follow.

6.3 Formulating Runtime Process using Agent-Based Module

Active firewall formulated here collaborates with distributed agent-based security modules that are deployed in every internal machine. This way, any suspicious process running in the internal machine can quickly be identified and then be informed to the firewall. Upon receiving any signal of the appearance of a threat, active firewall changes its configuration at runtime to isolate the victim machine. Let active firewall controls a set of reconfigureable canals *C* connecting an intranet that consist of *k* internal machines to the Internet, hence a set of active canals of an intranet can be formulated below

$$C = \{c(1), c(2) \Lambda\ c(n) \Lambda\ c(k)\} \tag{6.1}$$

And let Canal Manager handles modification of canals based on the condition of each machine, a set of canals correspond to an *n*-th internal machine is denoted by $c(n) = \{0,1\}$. In the normal condition, access to Internet from *n*-th internal machine is allowed to pass through the canals, thus $c(n) = 1$. Here normal condition is defined as the condition during which the internal machine has no detected threat, $th(n) = 0$. However access to Internet will be dropped if a threat is detected, $th(n) > 0$. This mechanism is formalized as:

$$c(n) = \begin{cases} 1 \Leftrightarrow th(n) = 0 \\ 0 \Leftrightarrow th(n) > 0 \end{cases} \tag{6.2}$$

To carry on implementing this scheme, architecture of active firewall utilizing agent technology is presented in Figure 6.1. Description of this architecture is given by modelling firewall functionalities in term of threat detection process and response time requirement.

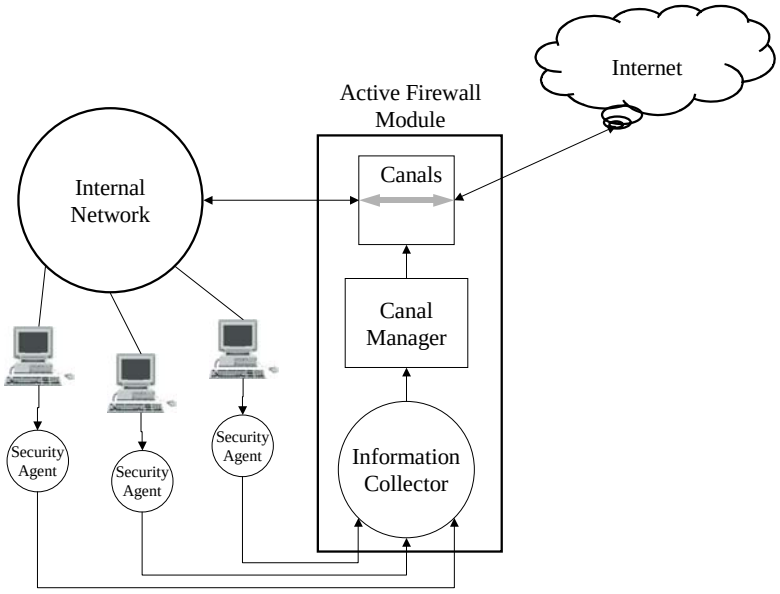


Figure 6.1: Architecture of the proposed agent-based active firewall

6.3.1 Threat Detection Process

The mechanism of threat detection is handled by installing the distributed agent-based security modules in every internal machine. To conduct the detection process, each agent-based module verifies every running application in each machine using the function of integrity check. Based on the information of the name, location, and the size of the running application, comparisons against a reference of authorized applications-list are computed as follow.

$$th(n)\begin{cases} = 0 \Leftrightarrow \left(nm(rp) = nm(ref)\right) \wedge \left(lc(rp) = lc(rp)\right) \wedge \left(sz(rp) = sz(ref)\right) \\ > 0 \Leftrightarrow \left(nm(rp) \neq nm(ref)\right) \vee \left(lc(rp) \neq lc(rp)\right) \vee \left(sz(rp) \neq sz(ref)\right) \end{cases} \quad (6.3)$$

with $th(n)$ denotes the present of the threat in the *n*-th internal machine. While *nm*, *lc* and *sz* represent the name, folder and size of the application respectively, here *rp* and *ref* denote the running process and reference respectively. The term $th(n) > 0$ means there is an appearance of a threat running in the internal machine. Upon detecting a threat, agent-based module will quickly communicate the result to active firewall.

It is notably important to emphasize the difference between a threat and the attack or intrusion. Threat identified here is not necessarily an intrusion, however with the assumption that an unrecognised object may potentially launch an attack, hence any unidentified applications are considered to become a threat to the intranet.

6.3.2 Response Time Requirement

Formulating this model is necessary to determine the required response time of active firewall. When there exist a suspicious process running in any internal machine at time $t = t_s$, agent-based security modules deployed in the intranet will sense the appearance of the threat at time $t = t_d$. Duration of $\Delta t_d = t_d - t_s$ seconds is needed by agent-based security module to detect the threat. Upon completing the detection process, agent-based security module send the information of a threat to active firewall. Duration of Δt_i seconds is consumed for transmitting the information from agent-based module to Information Collector in active firewall machine. And then Canal Manager processes the information in which it consumes Δt_p seconds before executing its final action to close the communication line of the machine where a threat is detected. In this scenario, reconfiguring the canals closes the communication line. Let canal reconfiguration is done at time t_c, hence t_c can be computed as:

$$t_c = t_s + \Delta t_d + \Delta t_i + \Delta t_p \qquad (6.4)$$

Considering that time consumption of the system is the total time required by threat detection process, followed by the communication process between agent-based security module and firewall, and finally the information processing by active firewall to execute canals, thus total time consumption of the system can be computed as:

$$\Delta t_w = \Delta t_d + \Delta t_i + \Delta t_p \qquad (6.5)$$

Substituting Equation (6.5) to Equation (6.4) produces

$$t_c - t_s = \Delta t_w \tag{6.6}$$

Since the purpose is to have real-time response to the appearance of a threat, thus minimum time consumption is desired, $\Delta t_w \to 0$. Applying the last condition to Equation (6.6) delivers:

$$\lim_{\Delta t_w \to 0} (t_c - t_s) = 0 \tag{6.7}$$

Equation (6.7) becomes the formal response time model of the proposed active firewall. This model presents the condition in which time consumption of the system shall be minimized. Therefore active firewall will be able to have real time operation to react to the appearance of a threat. In ideal condition when time consumption is zero second, the action for blocking the communication line of the victim machine can be executed in the same time of the appearance of the threat, i.e. $t_c = t_s$. If this condition is fulfilled, then it will be effective to prevent any unauthorized information flow to outside network.

6.4 Implementation of Active Firewall with Agent-Based Module

The developed runtime process is composed by two different components i.e. active firewall and distributed agent-based security module. The former part has a function to manage connection to the Internet, while the latter is to record any security incidents happening in every internal machine. In this case, the term incident refers to the action of malicious code. Detail explanations of each module follow.

6.4.1 Distributed Agent-Based Security Module

The implemented agent-based module has a function to monitor local processes of the internal machines by inspecting memory utilization. To identify each local process, predetermined authorized-applications list is developed and referred. By referring to this list, unrecognised running process is considered malicious. In this design, fast and accurate response to the present of the attack is desired, and the speed for passing the information of the suspected process to active firewall becomes the main concern as well. Algorithm of agent-based security module is shown in Figure 6.2. In this scheme, agent-based security modules work by sharing the same knowledge of the predefined authorized-applications list that become the reference point for conducting comparison against the running process. Complete implementation of this module is given in Appendix C1.

```
Procedure Agent Module
'REF    : array of predetermined applications reference
'Pr     : array of current process
Begin
        i = 0
        While i < 1 do
        Begin
                Pr = GetProcessName()
                For j = 1 to Pr.max do
                Begin
                        threat_status = True
                        For k = 1 to REF.max do
                        Begin
                                If Pr(j) = REF(k) then
                                Begin
                                        process_size = GetFileSize(REF(k). folder)
                                        If process_size =REF(k). size then
                                        Begin
                                                threat_status = False
                                                Exit For
                                         End
                                End
                         End
                        If threat_status = True then
                                SendMsgtoFirewall(IP address of local machine)
                End
        End
End
```

Figure 6.2: Algorithm of the agent-based security module

6.4.2 Agent-Based Active Firewall

The task of this module is to execute the canals corresponding to the internal victim machine upon receiving the information sent by distributed agent-based security module, since the information contains a log of the suspicious running process. The precaution is to isolate the internal machine where the suspicious process is detected, in which it is done by closing the communication line to the outside world. This way, the malicious information flow originated from this machine can be stopped.

Mechanism of active firewall is given in Figure 6.3. Here active firewall initialise communication line by executing a pre-configuration rule for opening communication by assuming at time *t=0* there is no threat detected. This operation is continued by the iterative process to wait any information transferred by the distributed agent module. When firewall receives the information from agent-module, it derives the identity of the victim machine in term of IP address, and based on this data firewall close the communication line corresponding to the victim machine. Complete implementation is given in Appendix C2.

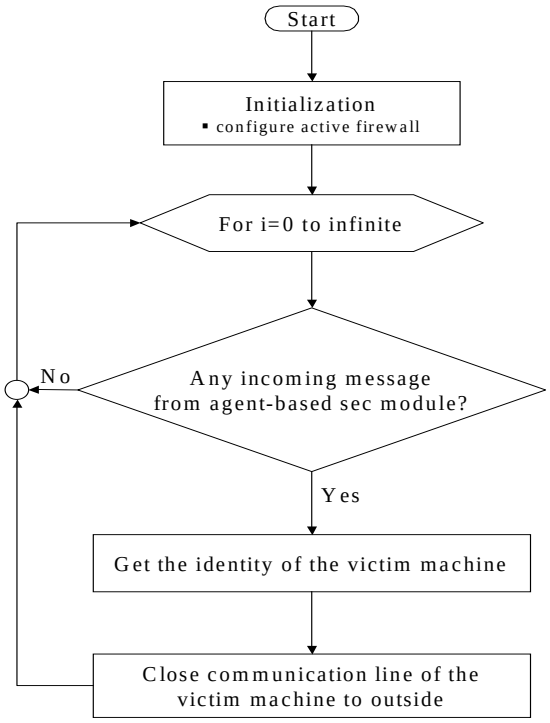


Figure 6.3: Mechanism of agent-based active firewall

6.5 Experiment

The purpose of the experiment is to evaluate the proposed agent-based method using the same set up as described in Chapter 4. The experiment was conducted by running a malicious program that has the mechanism to steal the information from the victim machine reside in the intranet. This program has a simple attack mechanism as shown in Figure 6.4. And then the agent-based active firewall is expected to detect the appearance of this threat and to stop the attack by closing the communication lines through modification of canals. Experimental results are presented in Table 6.1. To analyse these results, the speed of detecting malicious program and the speed to close the canals are discussed. Besides, two other parameters are introduced i.e. the probability to stop malicious information flow and the proportion of exposed time. The former parameter deals with the possibility that firewall capable to actively responding to the appearance of suspicious process, while the latter deals with the possible unauthorized release of information due to the action of suspicious process.

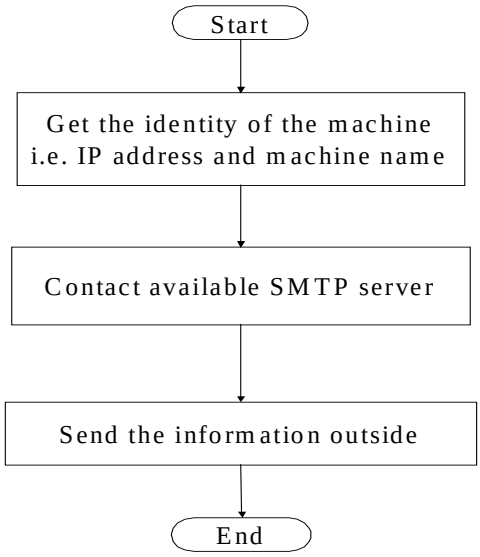


Figure 6.4: The created malicious program

To facilitate the analysis, a timeline graph visualizing the incident caused by a threat and the action of firewall to deter this attack is developed as shown in Figure 6.5. The graph contains a sequence of events occurring at time t_s to t_f, which can

be listed as follows. At t_s a malicious code is active, at t_m malicious code create malicious information flow, at t_d agent-based module detects an active threat and send this information to the firewall, at t_c firewall close the corresponding canals, and finally t_f is the completion of information flow if canals are not disabled at t_c. Let Δt_d, Δt_c, Δt_w, Δt_m, and Δt_f become random variable with Δt_d denote the service time to detect a threat, Δt_c is the service time to close the canals after a threat is detected, Δt_w is the total service time required by the firewall to stop the attack, Δt_m is the consumed time of a threat to start malicious information flow, and Δt_f is the total time consumption of a threat to complete its action. Referring to the graph in Figure 6.5, terms defined above are computed as follow.

$$\begin{aligned} \Delta t_d &= t_d - t_s \\ \Delta t_c &= t_c - t_d \\ \Delta t_w &= d + c \\ \Delta t_m &= t_m - t_s \\ \Delta t_f &= t_f - t_s \end{aligned} \tag{6.8}$$

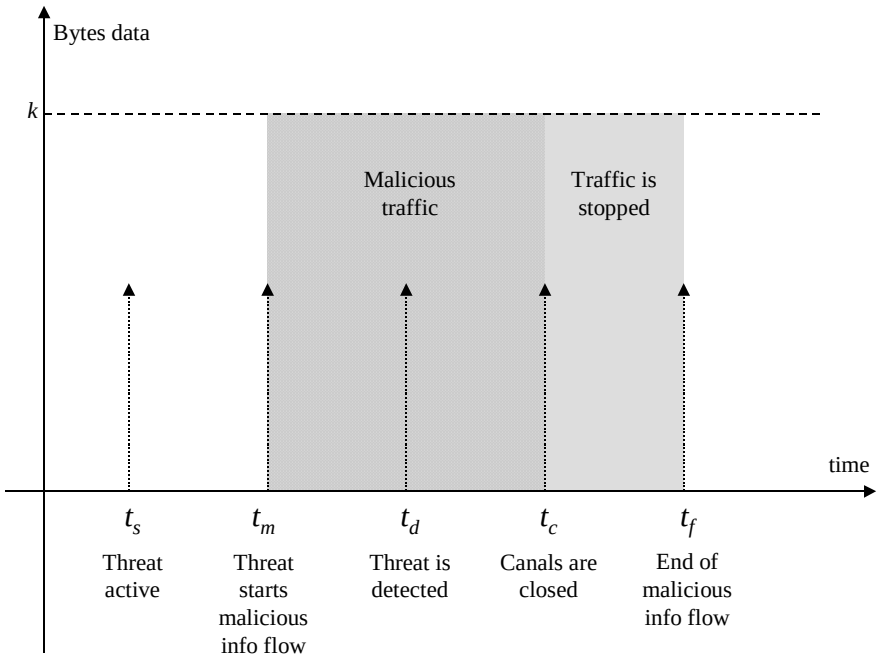


Figure 6.5: A timeline graph of security incident and the action of active firewall

Table 6.1: Experimental results of agent-based active firewall

No.	Threat		Active Firewall			
	Δt_m (second)	Δt_f (second)	Δt_d (second)	Δt_s (second)	Δt_c (second)	Δt_w (second)
1	0.069999993	0.460999966	0.08928	0.00237	0.22	0.31165
2	0.019999981	0.680999994	0.02711	0.001324	0.02	0.048434
3	0.050000012	0.57099998	0.03996	0.001283	0.08	0.121243
4	0.00999999	0.449999988	0.17068	0.002021	0.04	0.212701
5	0.019999981	0.511000037	0.09984	0.001303	0.11	0.211143
6	0.019999981	0.531000018	0.07188	0.001268	0.06	0.133148
7	0.00999999	0.620999992	0.03174	0.002054	0.01	0.043794
8	0.020999968	0.56099999	0.06797	0.001311	0.03	0.099281
9	0.020000041	0.641000032	0.18968	0.001323	0.03	0.221003
10	0.019999981	0.569999993	0.03311	0.001961	0.02	0.055071
11	0.019999981	0.480000019	0.0437	0.001324	0.28	0.325024
12	0.019999981	0.490999997	0.18768	0.001287	0.04	0.228967
13	0.019999981	0.870999992	0.09354	0.002068	0.05	0.145608
14	0.030000031	0.56099999	0.1428	0.00132	0.05	0.19412
15	0.019999981	0.550999999	0.04657	0.001286	0.04	0.087856
16	0.019999981	0.82099998	0.05503	0.002028	0.04	0.097058
17	0.08100003	0.611000001	0.1549	0.001386	0.04	0.196286
18	0.019999981	0.56099999	0.0627	0.001291	0.04	0.103991
19	0.019999981	0.579999983	0.12215	0.002058	0.11	0.234208
20	0.020000041	0.631000042	0.13076	0.001352	0.03	0.162112
21	0.019999981	0.601000011	0.07679	0.001319	0.06	0.138109
22	0.019999981	0.601000011	0.14363	0.001979	0.05	0.195609
23	0.020000041	0.5	0.07788	0.001292	0.09	0.169172
24	0.039999962	0.590999961	0.00019	0.0013	0.04	0.04149
25	0.019999981	3.11500001	0.09658	0.001997	0.04	0.138577
26	0.019999981	0.781000018	0.05974	0.001305	0.11	0.171045
27	0.019999981	0.640999973	0.15255	0.001302	0.04	0.193852
28	0.020000041	0.621000051	0.0777	0.001999	0.04	0.119699
29	0.01000005	0.711000025	0.0737	0.00132	0.04	0.11502
30	0.020000041	0.490999997	0.02323	0.001303	0.05	0.074533
31	0.00999999	0.651000023	0.11241	0.002062	0.15	0.264472
32	0.019999981	0.56099999	0.14446	0.001301	0.11	0.255761
33	0.019999981	0.550999999	0.19464	0.001347	0.1	0.295987
34	0.070000052	0.429999948	0.0374	0.00186	0.1	0.13926
35	0.020000041	0.641000032	0.07058	0.001304	0.04	0.111884
36	0.079999983	0.600999951	0.0677	0.001287	0.04	0.108987
37	0.00999999	0.430999994	0.04344	0.002047	0.04	0.085487
38	0.00999999	0.590999961	0.10015	0.001302	0.14	0.241452
39	0.00999999	0.591000021	0.08041	0.001273	0.09	0.171683
40	0.020000041	0.611000001	0.10758	0.001999	0.05	0.159579
41	0.020000041	0.560000002	0.02689	0.001303	0.11	0.138193
Average	**0.025170732**	**0.649463414**	**0.0885056**	**0.001557**	**0.07**	**0.160062**
Std Dev	**0.018252268**	**0.405695892**	**0.0503302**	**0.000356**	**0.054037**	**0.07293**

6.5.1 Speed of Detecting Suspicious Process

This parameter is critical to the performance of agent-based active firewall for protecting intranet. Referring to the timeline graph in Figure 6.5, active firewall needs to fulfil the following equation:

$$v(agent) < v(attack) \tag{6.9}$$

with v denotes the speed of agent or attack. This requirement is necessary for active firewall to take action following the detection process by further executing a protection mechanism before the attack is completed. Referring to the experimental results in Table 6.1, computation to confront the speed of detection against the speed of attack is presented as follows.

(i) Develop the function of attack f_{th} and function of detection speed f_d from the collected data. Here the experimental results are grouped into some small groups of data, i.e. each group consists of six experimental data, in order to build accurate functions. Therefore, from 41 data as presented in Table 6.1, seven groups of data are created. With x denotes the index of experiments, this step produces the following functions:

Group 1: $f_{th}(x) = -0.0113x^4 + 0.1768x^3 - 0.9557x^2 + 2.0257x - 0.7757$

$f_d(x) = 0.0042x^4 - 0.0692x^3 + 0.3861x^2 - 0.8213x + 0.5931$

Group 2: $f_{th}(x) = 0.0086x^4 - 0.118x^3 + 0.5459x^2 - 0.9897x + 1.1725$

$f_d(x) = y = 0.0069x^4 - 0.0817x^3 + 0.3071x^2 - 0.386x + 0.1803$

Group 3: $f_{th}(x) = 0.0137x^4 - 0.2201x^3 + 1.216x^2 - 2.6745x + 2.5443$

$f_d(x) = -0.0111x^4 + 0.1531x^3 - 0.7138x^2 + 1.2858x - 0.6205$

Group 4: $f_{th}(x) = 0.0038x^4 - 0.0441x^3 + 0.157x^2 - 0.184x + 0.65$

$f_d(x) = -0.0013x^4 + 0.0136x^3 - 0.0465x^2 + 0.053x + 0.1065$

Group 5: $f_{th}(x) = 0.0345x^4 - 0.5986x^3 + 3.7242x^2 - 9.7742x + 9.7193$

$f_d(x) = 0.0037x^4 - 0.054x^3 + 0.2581x^2 - 0.4679x + 0.3532$

Group 6: $f_{th}(x) = -0.0082x^4 + 0.1113x^3 - 0.4929x^2 + 0.7715x + 0.2627$

$f_d(x) = -2E\text{-}05x^4 + 0.0088x^3 - 0.0953x^2 + 0.2863x - 0.0923$

Group 7: $f_{th}(x) = -0.0113x^4 + 0.1429x^3 - 0.6552x^2 + 1.2946x - 0.34$

$f_d(x) = -0.0116x^4 + 0.1364x^3 - 0.5672x^2 + 0.977x - 0.4912$

(ii) Compute the area of each function produced from the previous step by conducting integration for both $F_{th} = \int_{x1}^{x2} f_{th}(x)dx$ and $F_d = \int_{x1}^{x2} f_d(x)dx$. Here $x1 = 1$ and $x2 = 6$ since each group contains six data, except for the last group that only has five data. This step produces:

Group 1: $F_{th} = 2.746917$ and $F_d = 0.39075$

Group 2: $F_{th} = 2.836083$ and $F_d = 0.434458$

Group 3: $F_{th} = 3.110542$ and $F_d = 0.548958$

Group 4: $F_{th} = 2.913292$ and $F_d = 0.509$

Group 5: $F_{th} = 4.29975$ and $F_d = 0.345917$

Group 6: $F_{th} = 2.772625$ and $F_d = 0.536817$

Group 7: $F_{th} = 2.32576$ and $F_d = 0.345653$

(iii) Compare F_{th} and F_d from each group. It produces $F_{th} > F_d$ for all groups of data, thus it can be concluded that Equation (6.9) is fulfilled by the developed agent-based active firewall.

Graph presentation of the function of attacks and detection speeds from each group of data, together with the function of the speed to start malicious information flow and the speed of canals, are presented in Appendix C3.

Repeating the steps above to compare the speed for starting malicious information flow against the speed for detecting suspicious process produces $v(agent) > v(maliciousflow)$. Detail calculations are given in Appendix C4. This fact shows that malicious information flows cannot be prevented if the malicious program has been activated, although if we scrutinize the experimental results there are about 10% (4 from 41 data) of the malicious flows can be prevented. Thus the action to prevent malicious flows shall only be held before the malicious program is activated. Otherwise the unauthorized release of information can only be stopped, not prevented.

6.5.2 Speed of Closing Canals

The speed of closing canals is other important factor influencing the performance of agent-based active firewall. This parameter show how fast canals can be closed relative to the appearance of malicious code as formulated in Equation (6.4). Firewall is effective to protect the intranet if it fulfils the equation below.

$$v(closing\ canals) < v(attack) \tag{6.10}$$

Referring to the experimental results presented in Table 6.1, and repeating the steps described in Section 6.5.1 to compare the speed of closing canals against the speed of the attacks, it produces $F_c < F_{th}$. Detail calculations are given in Appendix C4 with graph presentation in Appendix C3. This condition shows that active firewall capable to provide protection against malicious program launching the unauthorized information flows.

6.5.3 Proportion of Exposed Time

The purpose of formulating the proportion of the exposed time is to measure the possibility of unauthorized release of information before the attack is stopped by the firewall. With the assumption that $t_s < t_m < t_d < t_c < t_f$, proportion of the exposed time fulfils the following equation:

$$ET = \begin{cases} NA & \Leftrightarrow t < t_s \\ \upsilon_W / \upsilon_F & \Leftrightarrow t_s \le t < t_c \\ 0 & \Leftrightarrow t \ge t_c \end{cases} \qquad (6.11)$$

Proof. Malicious information flow here is defined as a flow of data from intranet to the Internet and/or vice versa due to the action of a threat. Referring to timeline graph in Figure 6.5, malicious traffic passing the firewall can be analysed as follow. At $t < t_s$ no threat is active in the intranet, thus no flow is considered malicious and the proportion of exposed time is not applicable (*NA*) in this condition. At $t = t_s$ a malicious code is active in the internal machine and launching a malicious traffic at $t = t_m$. If the communication line is kept opened, malicious traffic will successfully be completed at $t = t_f$. However if the agent-module detects the running threat, firewall will close the canals at $t = t_c$ to prevent any further unauthorized release of information. Thus the proportion of the exposed time can initially be computed as $ET = \frac{\upsilon_W - \upsilon_M}{\upsilon_F - \upsilon_M}$ with υ_M, υ_W and υ_F denote the mean average of Δt_m, Δt_w and Δt_f respectively. Assuming that $\forall t_m (t_m \leftarrow t_s)$, hence $\upsilon_M = \frac{\sum_{i=1}^{n} m_i}{n} = 0$ and $ET = \frac{\upsilon_W}{\upsilon_F}$ for $t_s \le t \le t_c$. However the last condition is violated when $t_c \ge t_f$. In this case, total lost of information is occurred since $\upsilon_W \ge \upsilon_F$ and $ET > 100\%$. Referring to the table of experimental result presented in Table 6.1, experiment shows that the active firewall produces *ET* = 24.4%. It means the internal victim machine is exposed to the external network for the duration of 24.4% of the total time required by the malicious program to launch an unauthorized information flow.

6.5.4 Summary

The implementation of the strategy for reducing unprotected users is presented in this chapter. The proposed method is to develop distributed agent-based security module to equip the operation of active firewall. These modules monitor the running processes of every internal user, and compare each process against a reference that contains a set of permitted applications. If an unrecognised process is detected, the module will inform the firewall, thus firewall can take an action to drop the communication line by modifying the canals corresponding to the internal user machine where the unrecognised process has been detected. Experiment shows that the developed method is capable to stop the malicious information flow driven by a malicious program that aims to steal the information of internal host.

CHAPTER 6

RUNTIME PROCESS:
MINIMIZING THE UNPROTECTED USERS

6.1 Introduction

With regard to security strategy defined in Chapter 3, this chapter presents the implementation of the strategy for minimizing the unprotected users. The developed method is still intended to handle the runtime process of active firewall. Hence the mechanism for $t > t_i$ is discussed. As mentioned in Chapter 3, it is more applicable to install a distributed detector in the intranet to watch the appearance of the threats, rather than having all security packages in every host of internal machines. Thus the developed method shall be capable to reduce the unprotected users by identifying and isolating the compromised internal machines. The choice spans from having intrusion detection, antivirus software, or vulnerability assessment. Since the purpose is to detect the compromised users, hence intrusion detection is considered more appropriated to be employed in this design. Referring to the work of Li *et al.* (2004) and Bernardes and Moreira (2000), which prove the applicability of agent software to guard the security of internal host, an agent-based system is developed with the purpose is to assist firewall in detecting malicious program running in the internal machines. This way firewall is expected to take a quick action to stop the unauthorized information flow that is potentially caused by the malicious program. Thus the victim machine can be isolated from getting access to or be exploited by the external party. To implement this mechanism, a distributed agent-based method is

developed. The advantage is due to the capability of distributed agent to cover an area that is considered too large to be handled by a single centralized system (Green *et al.*, 1997). Therefore the developed system is expected capable to monitor the running processes of all user machines in the intranet.

6.2 Review of Agent to Support Firewall

The notion of software agent has been around quite sometimes (White, 1996). Green *et al.* (1997) defines agent as an entity having fundamental properties of acting on behalf of others and enjoying a degree of autonomy. Agents also exhibit some level of proactivity and reactivity in its behaviour. Meanwhile Bernades and Moreira (2000) defines agent as a software program capable of executing a complex task on behalf of a user. A set of attributes, may equip the agent such as learning capabilities, cooperation, mobility, and adaptiveness (Green *et al.*, 1997; He and Leung, 2002). A number of advantages are offered by this technology such as capability to assist users in understanding a complex task and holding an ongoing execution for a relatively long period of time. In the domain of security, software agent have been used extensively to develop many types of security method such as security service (Shakshuki *et al.*, 2004), active security system (Zaki and Sobh, 2004), intrusion detection systems (Bernades and Moreira, 2000; Li *et al.*, 2004), network security management (Labiod and Boutapa, 2000) and micro firewalls (Hwang and Gangadharan, 2001). And due to its advantage to support distributed system, the use of software agent to build intrusion detection is widely accepted. However in the field of firewall, only few research efforts are reported formulating software agent to support the mechanism of firewall (Hwang and Gangadharan, 2001; Xian *et al.*, 2002). Mostly the agents in this field hold the function of intrusion detection system. These researches are inline with the suggestion of Davies (2000) to combine firewall and intrusion detection to have active and reactive security. However real-time operation is loosely considered in these efforts such as reported by Xian *et al.* (2002) that produce the empirical response time in the order of minute. Therefore the capability of firewall to stop an on going attack is questionable using this scenario. In

this research real-time operation of software agent is formulated to support active firewall. Detail descriptions follow.

6.3 Formulating Runtime Process using Agent-Based Module

Active firewall formulated here collaborates with distributed agent-based security modules that are deployed in every internal machine. This way, any suspicious process running in the internal machine can quickly be identified and then be informed to the firewall. Upon receiving any signal of the appearance of a threat, active firewall changes its configuration at runtime to isolate the victim machine. Let active firewall controls a set of reconfigureable canals *C* connecting an intranet that consist of *k* internal machines to the Internet, hence a set of active canals of an intranet can be formulated below

$$C = \{c(1), c(2) \Lambda\ c(n) \Lambda\ c(k)\} \tag{6.1}$$

And let Canal Manager handles modification of canals based on the condition of each machine, a set of canals correspond to an *n*-th internal machine is denoted by $c(n) = \{0,1\}$. In the normal condition, access to Internet from *n*-th internal machine is allowed to pass through the canals, thus $c(n) = 1$. Here normal condition is defined as the condition during which the internal machine has no detected threat, $th(n) = 0$. However access to Internet will be dropped if a threat is detected, $th(n) > 0$. This mechanism is formalized as:

$$c(n) = \begin{cases} 1 \Leftrightarrow th(n) = 0 \\ 0 \Leftrightarrow th(n) > 0 \end{cases} \tag{6.2}$$

To carry on implementing this scheme, architecture of active firewall utilizing agent technology is presented in Figure 6.1. Description of this architecture is given by modelling firewall functionalities in term of threat detection process and response time requirement.

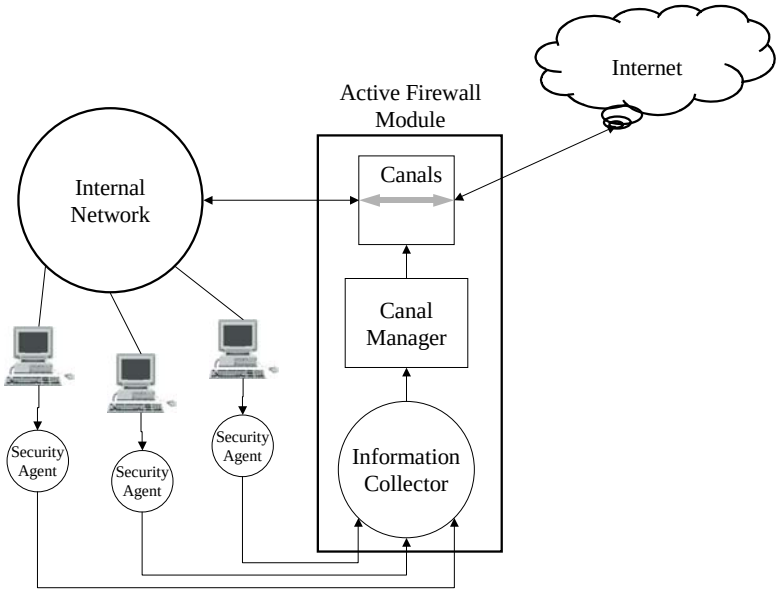


Figure 6.1: Architecture of the proposed agent-based active firewall

6.3.1 Threat Detection Process

The mechanism of threat detection is handled by installing the distributed agent-based security modules in every internal machine. To conduct the detection process, each agent-based module verifies every running application in each machine using the function of integrity check. Based on the information of the name, location, and the size of the running application, comparisons against a reference of authorized applications-list are computed as follow.

$$th(n)\begin{cases} = 0 \Leftrightarrow \left(nm(rp) = nm(ref)\right) \wedge \left(lc(rp) = lc(rp)\right) \wedge \left(sz(rp) = sz(ref)\right) \\ > 0 \Leftrightarrow \left(nm(rp) \neq nm(ref)\right) \vee \left(lc(rp) \neq lc(rp)\right) \vee \left(sz(rp) \neq sz(ref)\right) \end{cases} \quad (6.3)$$

with $th(n)$ denotes the present of the threat in the *n*-th internal machine. While *nm*, *lc* and *sz* represent the name, folder and size of the application respectively, here *rp* and *ref* denote the running process and reference respectively. The term $th(n) > 0$ means there is an appearance of a threat running in the internal machine. Upon detecting a threat, agent-based module will quickly communicate the result to active firewall.

It is notably important to emphasize the difference between a threat and the attack or intrusion. Threat identified here is not necessarily an intrusion, however with the assumption that an unrecognised object may potentially launch an attack, hence any unidentified applications are considered to become a threat to the intranet.

6.3.2 Response Time Requirement

Formulating this model is necessary to determine the required response time of active firewall. When there exist a suspicious process running in any internal machine at time $t = t_s$, agent-based security modules deployed in the intranet will sense the appearance of the threat at time $t = t_d$. Duration of $\Delta t_d = t_d - t_s$ seconds is needed by agent-based security module to detect the threat. Upon completing the detection process, agent-based security module send the information of a threat to active firewall. Duration of Δt_i seconds is consumed for transmitting the information from agent-based module to Information Collector in active firewall machine. And then Canal Manager processes the information in which it consumes Δt_p seconds before executing its final action to close the communication line of the machine where a threat is detected. In this scenario, reconfiguring the canals closes the communication line. Let canal reconfiguration is done at time t_c, hence t_c can be computed as:

$$t_c = t_s + \Delta t_d + \Delta t_i + \Delta t_p \qquad (6.4)$$

Considering that time consumption of the system is the total time required by threat detection process, followed by the communication process between agent-based security module and firewall, and finally the information processing by active firewall to execute canals, thus total time consumption of the system can be computed as:

$$\Delta t_w = \Delta t_d + \Delta t_i + \Delta t_p \qquad (6.5)$$

Substituting Equation (6.5) to Equation (6.4) produces

$$t_c - t_s = \Delta t_w \tag{6.6}$$

Since the purpose is to have real-time response to the appearance of a threat, thus minimum time consumption is desired, $\Delta t_w \to 0$. Applying the last condition to Equation (6.6) delivers:

$$\lim_{\Delta t_w \to 0} (t_c - t_s) = 0 \tag{6.7}$$

Equation (6.7) becomes the formal response time model of the proposed active firewall. This model presents the condition in which time consumption of the system shall be minimized. Therefore active firewall will be able to have real time operation to react to the appearance of a threat. In ideal condition when time consumption is zero second, the action for blocking the communication line of the victim machine can be executed in the same time of the appearance of the threat, i.e. $t_c = t_s$. If this condition is fulfilled, then it will be effective to prevent any unauthorized information flow to outside network.

6.4 Implementation of Active Firewall with Agent-Based Module

The developed runtime process is composed by two different components i.e. active firewall and distributed agent-based security module. The former part has a function to manage connection to the Internet, while the latter is to record any security incidents happening in every internal machine. In this case, the term incident refers to the action of malicious code. Detail explanations of each module follow.

6.4.1 Distributed Agent-Based Security Module

The implemented agent-based module has a function to monitor local processes of the internal machines by inspecting memory utilization. To identify each local process, predetermined authorized-applications list is developed and referred. By referring to this list, unrecognised running process is considered malicious. In this design, fast and accurate response to the present of the attack is desired, and the speed for passing the information of the suspected process to active firewall becomes the main concern as well. Algorithm of agent-based security module is shown in Figure 6.2. In this scheme, agent-based security modules work by sharing the same knowledge of the predefined authorized-applications list that become the reference point for conducting comparison against the running process. Complete implementation of this module is given in Appendix C1.

```
Procedure Agent Module
'REF    : array of predetermined applications reference
'Pr     : array of current process
Begin
        i = 0
        While i < 1 do
        Begin
                Pr = GetProcessName()
                For j = 1 to Pr.max do
                Begin
                        threat_status = True
                        For k = 1 to REF.max do
                        Begin
                                If Pr(j) = REF(k) then
                                Begin
                                        process_size = GetFileSize(REF(k). folder)
                                        If process_size =REF(k). size then
                                        Begin
                                                threat_status = False
                                                Exit For
                                         End
                                End
                         End
                        If threat_status = True then
                                SendMsgtoFirewall(IP address of local machine)
                End
        End
End
```

Figure 6.2: Algorithm of the agent-based security module

6.4.2 Agent-Based Active Firewall

The task of this module is to execute the canals corresponding to the internal victim machine upon receiving the information sent by distributed agent-based security module, since the information contains a log of the suspicious running process. The precaution is to isolate the internal machine where the suspicious process is detected, in which it is done by closing the communication line to the outside world. This way, the malicious information flow originated from this machine can be stopped.

Mechanism of active firewall is given in Figure 6.3. Here active firewall initialise communication line by executing a pre-configuration rule for opening communication by assuming at time *t=0* there is no threat detected. This operation is continued by the iterative process to wait any information transferred by the distributed agent module. When firewall receives the information from agent-module, it derives the identity of the victim machine in term of IP address, and based on this data firewall close the communication line corresponding to the victim machine. Complete implementation is given in Appendix C2.

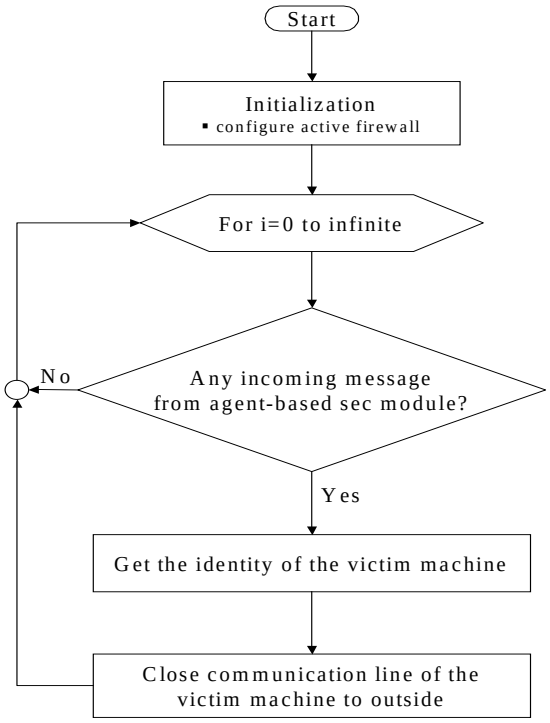


Figure 6.3: Mechanism of agent-based active firewall

6.5 Experiment

The purpose of the experiment is to evaluate the proposed agent-based method using the same set up as described in Chapter 4. The experiment was conducted by running a malicious program that has the mechanism to steal the information from the victim machine reside in the intranet. This program has a simple attack mechanism as shown in Figure 6.4. And then the agent-based active firewall is expected to detect the appearance of this threat and to stop the attack by closing the communication lines through modification of canals. Experimental results are presented in Table 6.1. To analyse these results, the speed of detecting malicious program and the speed to close the canals are discussed. Besides, two other parameters are introduced i.e. the probability to stop malicious information flow and the proportion of exposed time. The former parameter deals with the possibility that firewall capable to actively responding to the appearance of suspicious process, while the latter deals with the possible unauthorized release of information due to the action of suspicious process.

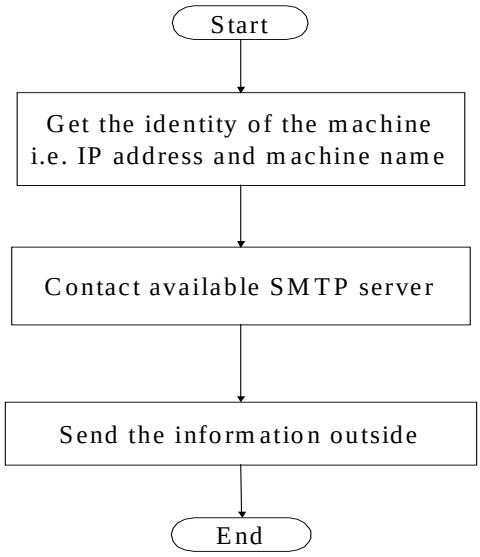


Figure 6.4: The created malicious program

To facilitate the analysis, a timeline graph visualizing the incident caused by a threat and the action of firewall to deter this attack is developed as shown in Figure 6.5. The graph contains a sequence of events occurring at time t_s to t_f, which can

be listed as follows. At t_s a malicious code is active, at t_m malicious code create malicious information flow, at t_d agent-based module detects an active threat and send this information to the firewall, at t_c firewall close the corresponding canals, and finally t_f is the completion of information flow if canals are not disabled at t_c. Let Δt_d, Δt_c, Δt_w, Δt_m, and Δt_f become random variable with Δt_d denote the service time to detect a threat, Δt_c is the service time to close the canals after a threat is detected, Δt_w is the total service time required by the firewall to stop the attack, Δt_m is the consumed time of a threat to start malicious information flow, and Δt_f is the total time consumption of a threat to complete its action. Referring to the graph in Figure 6.5, terms defined above are computed as follow.

$$\begin{aligned} \Delta t_d &= t_d - t_s \\ \Delta t_c &= t_c - t_d \\ \Delta t_w &= d + c \\ \Delta t_m &= t_m - t_s \\ \Delta t_f &= t_f - t_s \end{aligned} \tag{6.8}$$

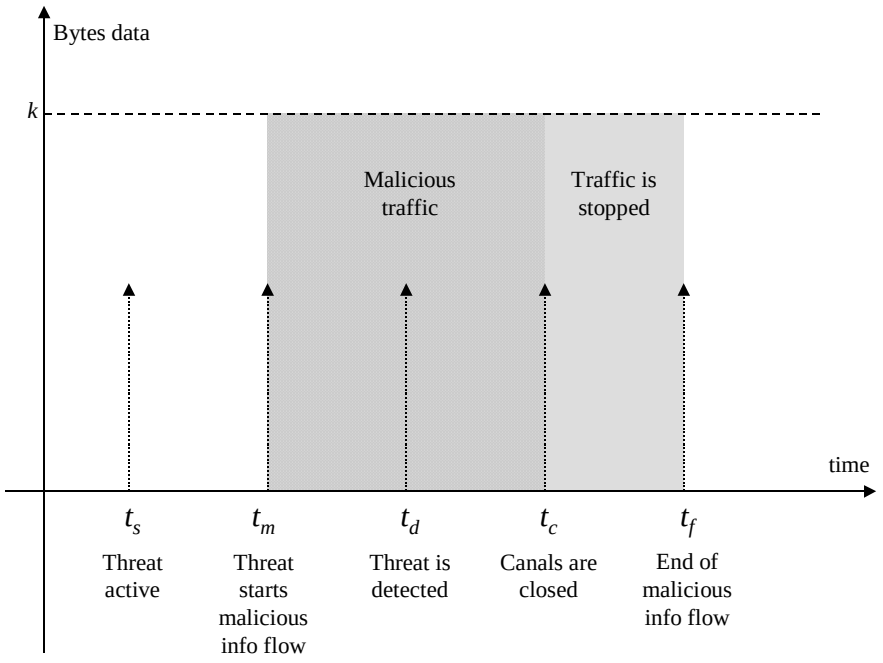


Figure 6.5: A timeline graph of security incident and the action of active firewall

Table 6.1: Experimental results of agent-based active firewall

No.	Threat		Active Firewall			
	Δt_m (second)	Δt_f (second)	Δt_d (second)	Δt_s (second)	Δt_c (second)	Δt_w (second)
1	0.069999993	0.460999966	0.08928	0.00237	0.22	0.31165
2	0.019999981	0.680999994	0.02711	0.001324	0.02	0.048434
3	0.050000012	0.57099998	0.03996	0.001283	0.08	0.121243
4	0.00999999	0.449999988	0.17068	0.002021	0.04	0.212701
5	0.019999981	0.511000037	0.09984	0.001303	0.11	0.211143
6	0.019999981	0.531000018	0.07188	0.001268	0.06	0.133148
7	0.00999999	0.620999992	0.03174	0.002054	0.01	0.043794
8	0.020999968	0.56099999	0.06797	0.001311	0.03	0.099281
9	0.020000041	0.641000032	0.18968	0.001323	0.03	0.221003
10	0.019999981	0.569999993	0.03311	0.001961	0.02	0.055071
11	0.019999981	0.480000019	0.0437	0.001324	0.28	0.325024
12	0.019999981	0.490999997	0.18768	0.001287	0.04	0.228967
13	0.019999981	0.870999992	0.09354	0.002068	0.05	0.145608
14	0.030000031	0.56099999	0.1428	0.00132	0.05	0.19412
15	0.019999981	0.550999999	0.04657	0.001286	0.04	0.087856
16	0.019999981	0.82099998	0.05503	0.002028	0.04	0.097058
17	0.08100003	0.611000001	0.1549	0.001386	0.04	0.196286
18	0.019999981	0.56099999	0.0627	0.001291	0.04	0.103991
19	0.019999981	0.579999983	0.12215	0.002058	0.11	0.234208
20	0.020000041	0.631000042	0.13076	0.001352	0.03	0.162112
21	0.019999981	0.601000011	0.07679	0.001319	0.06	0.138109
22	0.019999981	0.601000011	0.14363	0.001979	0.05	0.195609
23	0.020000041	0.5	0.07788	0.001292	0.09	0.169172
24	0.039999962	0.590999961	0.00019	0.0013	0.04	0.04149
25	0.019999981	3.11500001	0.09658	0.001997	0.04	0.138577
26	0.019999981	0.781000018	0.05974	0.001305	0.11	0.171045
27	0.019999981	0.640999973	0.15255	0.001302	0.04	0.193852
28	0.020000041	0.621000051	0.0777	0.001999	0.04	0.119699
29	0.01000005	0.711000025	0.0737	0.00132	0.04	0.11502
30	0.020000041	0.490999997	0.02323	0.001303	0.05	0.074533
31	0.00999999	0.651000023	0.11241	0.002062	0.15	0.264472
32	0.019999981	0.56099999	0.14446	0.001301	0.11	0.255761
33	0.019999981	0.550999999	0.19464	0.001347	0.1	0.295987
34	0.070000052	0.429999948	0.0374	0.00186	0.1	0.13926
35	0.020000041	0.641000032	0.07058	0.001304	0.04	0.111884
36	0.079999983	0.600999951	0.0677	0.001287	0.04	0.108987
37	0.00999999	0.430999994	0.04344	0.002047	0.04	0.085487
38	0.00999999	0.590999961	0.10015	0.001302	0.14	0.241452
39	0.00999999	0.591000021	0.08041	0.001273	0.09	0.171683
40	0.020000041	0.611000001	0.10758	0.001999	0.05	0.159579
41	0.020000041	0.560000002	0.02689	0.001303	0.11	0.138193
Average	**0.025170732**	**0.649463414**	**0.0885056**	**0.001557**	**0.07**	**0.160062**
Std Dev	**0.018252268**	**0.405695892**	**0.0503302**	**0.000356**	**0.054037**	**0.07293**

6.5.1 Speed of Detecting Suspicious Process

This parameter is critical to the performance of agent-based active firewall for protecting intranet. Referring to the timeline graph in Figure 6.5, active firewall needs to fulfil the following equation:

$$v(agent) < v(attack) \tag{6.9}$$

with v denotes the speed of agent or attack. This requirement is necessary for active firewall to take action following the detection process by further executing a protection mechanism before the attack is completed. Referring to the experimental results in Table 6.1, computation to confront the speed of detection against the speed of attack is presented as follows.

(i) Develop the function of attack f_{th} and function of detection speed f_d from the collected data. Here the experimental results are grouped into some small groups of data, i.e. each group consists of six experimental data, in order to build accurate functions. Therefore, from 41 data as presented in Table 6.1, seven groups of data are created. With x denotes the index of experiments, this step produces the following functions:

Group 1: $f_{th}(x) = -0.0113x^4 + 0.1768x^3 - 0.9557x^2 + 2.0257x - 0.7757$

$f_d(x) = 0.0042x^4 - 0.0692x^3 + 0.3861x^2 - 0.8213x + 0.5931$

Group 2: $f_{th}(x) = 0.0086x^4 - 0.118x^3 + 0.5459x^2 - 0.9897x + 1.1725$

$f_d(x) = y = 0.0069x^4 - 0.0817x^3 + 0.3071x^2 - 0.386x + 0.1803$

Group 3: $f_{th}(x) = 0.0137x^4 - 0.2201x^3 + 1.216x^2 - 2.6745x + 2.5443$

$f_d(x) = -0.0111x^4 + 0.1531x^3 - 0.7138x^2 + 1.2858x - 0.6205$

Group 4: $f_{th}(x) = 0.0038x^4 - 0.0441x^3 + 0.157x^2 - 0.184x + 0.65$

$f_d(x) = -0.0013x^4 + 0.0136x^3 - 0.0465x^2 + 0.053x + 0.1065$

Group 5: $f_{th}(x) = 0.0345x^4 - 0.5986x^3 + 3.7242x^2 - 9.7742x + 9.7193$

$f_d(x) = 0.0037x^4 - 0.054x^3 + 0.2581x^2 - 0.4679x + 0.3532$

Group 6: $f_{th}(x) = -0.0082x^4 + 0.1113x^3 - 0.4929x^2 + 0.7715x + 0.2627$

$f_d(x) = -2E\text{-}05x^4 + 0.0088x^3 - 0.0953x^2 + 0.2863x - 0.0923$

Group 7: $f_{th}(x) = -0.0113x^4 + 0.1429x^3 - 0.6552x^2 + 1.2946x - 0.34$

$f_d(x) = -0.0116x^4 + 0.1364x^3 - 0.5672x^2 + 0.977x - 0.4912$

(ii) Compute the area of each function produced from the previous step by conducting integration for both $F_{th} = \int_{x1}^{x2} f_{th}(x)dx$ and $F_d = \int_{x1}^{x2} f_d(x)dx$. Here $x1 = 1$ and $x2 = 6$ since each group contains six data, except for the last group that only has five data. This step produces:

Group 1: $F_{th} = 2.746917$ and $F_d = 0.39075$

Group 2: $F_{th} = 2.836083$ and $F_d = 0.434458$

Group 3: $F_{th} = 3.110542$ and $F_d = 0.548958$

Group 4: $F_{th} = 2.913292$ and $F_d = 0.509$

Group 5: $F_{th} = 4.29975$ and $F_d = 0.345917$

Group 6: $F_{th} = 2.772625$ and $F_d = 0.536817$

Group 7: $F_{th} = 2.32576$ and $F_d = 0.345653$

(iii) Compare F_{th} and F_d from each group. It produces $F_{th} > F_d$ for all groups of data, thus it can be concluded that Equation (6.9) is fulfilled by the developed agent-based active firewall.

Graph presentation of the function of attacks and detection speeds from each group of data, together with the function of the speed to start malicious information flow and the speed of canals, are presented in Appendix C3.

Repeating the steps above to compare the speed for starting malicious information flow against the speed for detecting suspicious process produces $v(agent) > v(maliciousflow)$. Detail calculations are given in Appendix C4. This fact shows that malicious information flows cannot be prevented if the malicious program has been activated, although if we scrutinize the experimental results there are about 10% (4 from 41 data) of the malicious flows can be prevented. Thus the action to prevent malicious flows shall only be held before the malicious program is activated. Otherwise the unauthorized release of information can only be stopped, not prevented.

6.5.2 Speed of Closing Canals

The speed of closing canals is other important factor influencing the performance of agent-based active firewall. This parameter show how fast canals can be closed relative to the appearance of malicious code as formulated in Equation (6.4). Firewall is effective to protect the intranet if it fulfils the equation below.

$$v(closing\ canals) < v(attack) \tag{6.10}$$

Referring to the experimental results presented in Table 6.1, and repeating the steps described in Section 6.5.1 to compare the speed of closing canals against the speed of the attacks, it produces $F_c < F_{th}$. Detail calculations are given in Appendix C4 with graph presentation in Appendix C3. This condition shows that active firewall capable to provide protection against malicious program launching the unauthorized information flows.

6.5.3 Proportion of Exposed Time

The purpose of formulating the proportion of the exposed time is to measure the possibility of unauthorized release of information before the attack is stopped by the firewall. With the assumption that $t_s < t_m < t_d < t_c < t_f$, proportion of the exposed time fulfils the following equation:

$$ET = \begin{cases} NA & \Leftrightarrow t < t_s \\ \upsilon_W / \upsilon_F & \Leftrightarrow t_s \le t < t_c \\ 0 & \Leftrightarrow t \ge t_c \end{cases} \qquad (6.11)$$

Proof. Malicious information flow here is defined as a flow of data from intranet to the Internet and/or vice versa due to the action of a threat. Referring to timeline graph in Figure 6.5, malicious traffic passing the firewall can be analysed as follow. At $t < t_s$ no threat is active in the intranet, thus no flow is considered malicious and the proportion of exposed time is not applicable (*NA*) in this condition. At $t = t_s$ a malicious code is active in the internal machine and launching a malicious traffic at $t = t_m$. If the communication line is kept opened, malicious traffic will successfully be completed at $t = t_f$. However if the agent-module detects the running threat, firewall will close the canals at $t = t_c$ to prevent any further unauthorized release of information. Thus the proportion of the exposed time can initially be computed as $ET = \frac{\upsilon_W - \upsilon_M}{\upsilon_F - \upsilon_M}$ with υ_M, υ_W and υ_F denote the mean average of Δt_m, Δt_w and Δt_f respectively. Assuming that $\forall t_m (t_m \leftarrow t_s)$, hence $\upsilon_M = \frac{\sum_{i=1}^{n} m_i}{n} = 0$ and $ET = \frac{\upsilon_W}{\upsilon_F}$ for $t_s \le t \le t_c$. However the last condition is violated when $t_c \ge t_f$. In this case, total lost of information is occurred since $\upsilon_W \ge \upsilon_F$ and $ET > 100\%$. Referring to the table of experimental result presented in Table 6.1, experiment shows that the active firewall produces *ET* = 24.4%. It means the internal victim machine is exposed to the external network for the duration of 24.4% of the total time required by the malicious program to launch an unauthorized information flow.

6.5.4 Summary

The implementation of the strategy for reducing unprotected users is presented in this chapter. The proposed method is to develop distributed agent-based security module to equip the operation of active firewall. These modules monitor the running processes of every internal user, and compare each process against a reference that contains a set of permitted applications. If an unrecognised process is detected, the module will inform the firewall, thus firewall can take an action to drop the communication line by modifying the canals corresponding to the internal user machine where the unrecognised process has been detected. Experiment shows that the developed method is capable to stop the malicious information flow driven by a malicious program that aims to steal the information of internal host.

CHAPTER 7

RUNTIME PROCESS: MINIMIZING THE UNTRUSTED EXTERNAL PARTIES WITH ZERO-BASED CONFIGURATION

7.1 Introduction

This chapter presents the development of runtime process using minimal configuration of network services. The methods proposed in this stage is still to follow the initialization process presented in Chapter 4. Referring to the threats reduction strategy developed in Chapter 3, the method introduced in this chapter aims to minimize the untrusted external parties by limiting the available services at runtime, and restricting the access only to the authorized external parties. The experience of Ranum and Avolio (1994) to deal with firewall and network traffic motivates the formulation of this method. In their report (Ranum and Avolio, 1994), it is stated that turning network services at minimum would make firewall harder to break into. Hence it discloses the fact that current firewall technology offers too much services, particularly with static configuration built at start up and remained unchanged until the firewall is shut down. It is also worth to note that any security holes exist in the existing static configuration would be maintained along the operation of firewall. Moreover, Goncalves (2000) stressed the difficulties on auditing and administering the overall available network services at once due to the possibilities of unauthorized information flow that may rise from any of the open services without the knowledge of the internal user. Thus the idea behind zero-based configuration is to offer minimum services to secure Internet access by closing all

open connection in the idle time. This way, there will be no flowing traffic passing through the firewall without being screened first. It ensures that any Internet transactions are done with the trusted external parties.

The term “zero-based” it-self is taken from the financial sector, in which it has been used as the name of a budgeting method. In this field, zero-based stipulates the requirements that each appropriation in a budget year should justify the amount in excess of zero (Anthony, 2003). Therefore the basis of running this method is zero, and any expenditure must be rejustified during each budgeting cycle. In this research, the principle of zero-based approach is used to handle the runtime process of active firewall. Here configuration of network services at run time becomes the object of interest, thus minimizing this configuration become the objective of this method. Formulation is described in the next section.

7.2 Formulation

Let C become a set of reconfigureable canals establishing connection between intranet and the Internet. And $c(n,m) \in C$ denotes a set of canals connecting the internal user n to the requested external parties m, hence access to Internet using zero-based configuration is held by holding some activities as described using a time line graph depicted in Figure 7.1. Referring to this figure, if there exist an internal user requesting access to Internet at time $t = t_r$, the active firewall would establish the connection at time $t = t_e$ by activating $c(n,m)$, maintain the canals for the duration of Δt seconds, and then drop the canals at time $t = t_d$, with $t = t_e + \Delta t$. From this point, canals reconfiguration can be formulated as follows

$$c(n,m) = \begin{cases} 1 \Leftrightarrow (c(n,m) \in C \text{ for } m \in M) \wedge (t_e \le t \le t_d) \\ 0 \quad \text{otherwise} \end{cases} \tag{7.1}$$

with $m \in M$ become a set of external parties allowed to be contacted by the internal users. As mentioned previously that the established canals will be dropped after the duration of Δt seconds, hence Δt is defined as the timeout of network transaction, and for each transaction the timeout is determined using the following equation

$$\Delta t = \upsilon(m) + 3\tau(m) \tag{7.2}$$

with υ and τ denote the average mean and the standard deviation respectively of time consumption to conduct network transaction with the external parties corresponding to $c(n,m)$. In this method, prior measurement to empirically determine time consumption is required. It is worth to note that equation (7.2) follows the empirical rule of statistics with 99,7% approximation of time consumption (Triola, 2001).

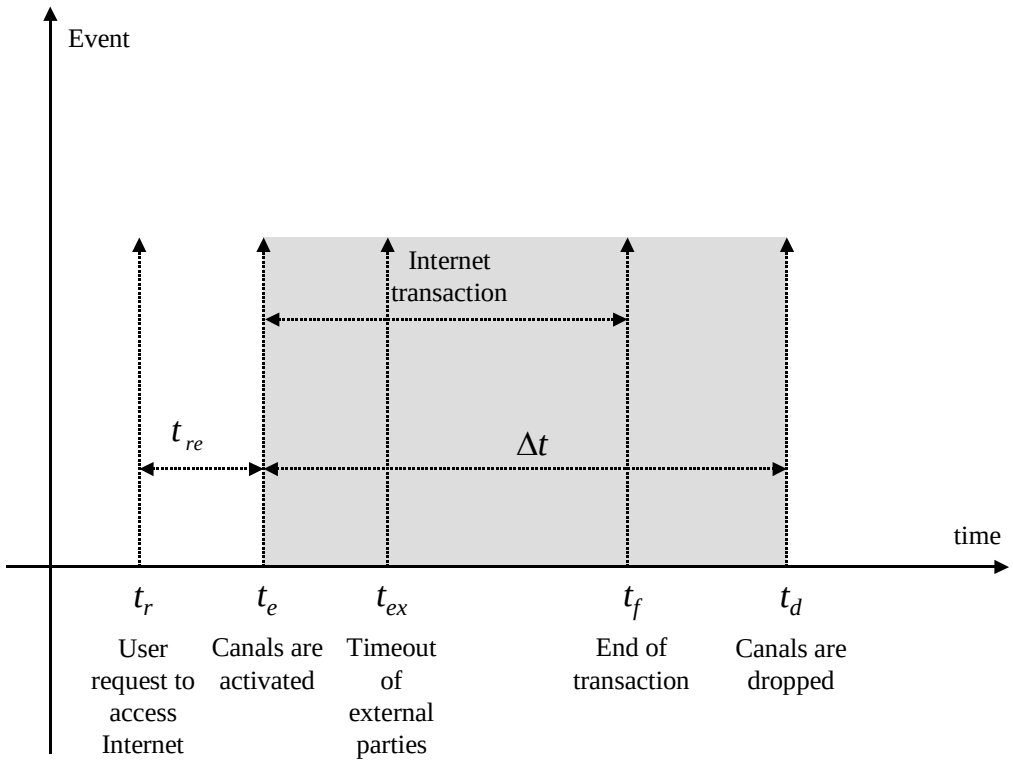


Figure 7.1: A series of events in accessing Internet using zero-based configuration

7.3 Implementation

Active firewall running zero-based configuration is implemented by developing the following mechanism:

(i) Running the close-condition approach for initialization process.

(ii) Waiting for the access request of the internal users at runtime, and for each permitted request, it is followed by enforcing the security rule in term of modifying the configuration of canals.

(iii) For each activated canal, the active firewall will monitor its lifetime. So when the lifetime reaches the timeout, the security rule will be dropped. Thus the zero-configuration of canals can be maintained along the operation of active firewall.

To implement the first mechanism above, the close-condition method developed in Chapter 4 is employed. And to implement the second mechanism, real-time traffic filtering is required to analyze every flowing packet intending to pass through the firewall. This mechanism is processing intensive, hence simplification of the analysis is necessary to achieve the real time process. Therefore only the header of network packet is analyzed. It is implemented by developing the foreground algorithm as depicted in Figure 7.2.

Meanwhile, to implement the third mechanism, prior measurement of time consumption for communicating with the external parties is developed. This data are necessary to compute the timeout of each transaction, thus the live time of each security rule can be determined using the empirical rule as formulated in Equation (7.2). This mechanism is implemented by developing the background algorithm as shown in Figure 7.3. Here the choice to run the dropping mechanism for each life canal in the background is due to the operation of the third mechanism above that less requires real-time execution compared to the enforcing security rule included in the second mechanism. Complete implementations of foreground and background algorithm are given in Appendix D4 and D5 respectively.

```
Procedure Foreground Zero Configuration
'TBUF   : array of traffic buffer
'C      : array of canals
'Q      : table of permitted external parties
Begin
   i = 1
   While i ≤ TBUF.size do
   Begin
        Line_content = ""
        j = 1
        Do Until (TBUF(i) = EOL) Or (i = TBUF.size)
            Line_content = Line_content + TBUF(i)
            If (TBUF(i) = " ") Then
            Begin
                Word(j) = Line_content
                Line_content = ""
                j = j + 1
            End
            i = i + 1
        Loop
        If Word = Packet.Header Then
        Begin
          For j = 1 to Word.size do
          Begin
                If Word(j) = External_party And External_party is new Then
                    External _ party ⇒ Array _ Outsider.External
                If Word(j) = Internal_user And Internal_user is new Then
                    Internal _ user ⇒ Array _ Outsider.Internal
                If Word(j) = Protocol Then Array_Outsider.Protocol = Word(j)
          End
          If Array_Outsider ≠ C Then
                If Array_Outsider.External ∈ Q Then Array_Outsider → C
        End
   End
End
```

Figure 7.2: Foreground algorithm

```
Procedure Background Zero Configuration
'C      : array of canals
'Q      : table of permitted external parties
Begin
   i = 1
   While i ≤ C.size do
   Begin
        For j = 1 to Q.size do
            If C(i).external = Q(j).external Then
                If (PresentTime() - C(i).start) > (Q(j).timeout + 3* Std_dev(Q(j).timeout)) Then
                Drop(C(i))
        i = i + 1
    End
End
```

Figure 7.3: Background algorithm

7.4 Experiment

Evaluation on the implemented zero-based configuration was conducted on the same experimental set up as in Chapter 4. A set of security policies used in the experiment was defined in term of a table of rules that contain the list of permitted Internet parties. The table is presented in Appendix D1. Using this table, firewall can decide whether the request to access Internet is permitted or not. The table also supplies the firewall with the information about the timeout period of each external party. Experimental results are presented in Appendix D2.

7.4.1 Speed to Open Canal

Since the firewall has to manage real time execution of Internet access, the speed of opening canals becomes the important factor influencing the implementation of zero-based configuration. Referring to Figure 7.1, this parameter deals with the establishment of t_e relative to t_r. Thus low speed to open canals most probably causes denial of service. In this implementation, the speed to open canal is affected by the location of Internet access request in the traffic buffer. If the request exists in the beginning of network traffic, the speed to open canal can be computed using the following equation $t_{re} = t_e - t_r$. But if the request appears in the middle of traffic buffer, then the speed to open canal is affected by the processing time of foreground algorithm to analyze the contents of buffer, in which iteration to browse the buffer is held. Here the processing time can be measured by computing the number of packets contained in the buffer. By assuming that the worst-case scenario is met i.e. the access request is appeared at the end of traffic buffer, using the computation to compare the processing time and the number of packets, the processing time to open canal can be determined as follow.

$$t_{re} = \left(\frac{s}{s_{ref}} \right)^n \Delta t_{ref} + k \qquad (7.3)$$

with n refers to the number of iteration, s is the buffer size in term of the number of packets, k is a constant representing the required time to complete the program excluding the main iteration and Δt is the processing time. Since the foreground algorithm depicted in Figure 7.2 produces n=1, thus linear processing time following the growth of packets in the buffer can be delivered. Therefore real time Internet access can be handled by this method.

Referring to the experimental results in Appendix D3, the graph presenting the increase of processing time compared to the number of packets is shown in Figure 7.4. It proves that Equation (7.3) is satisfied. Meanwhile, empirical measurement in Appendix D2 produces the average of 4.9 seconds with standard deviation 4.7 seconds are required by this algorithm to open canals.

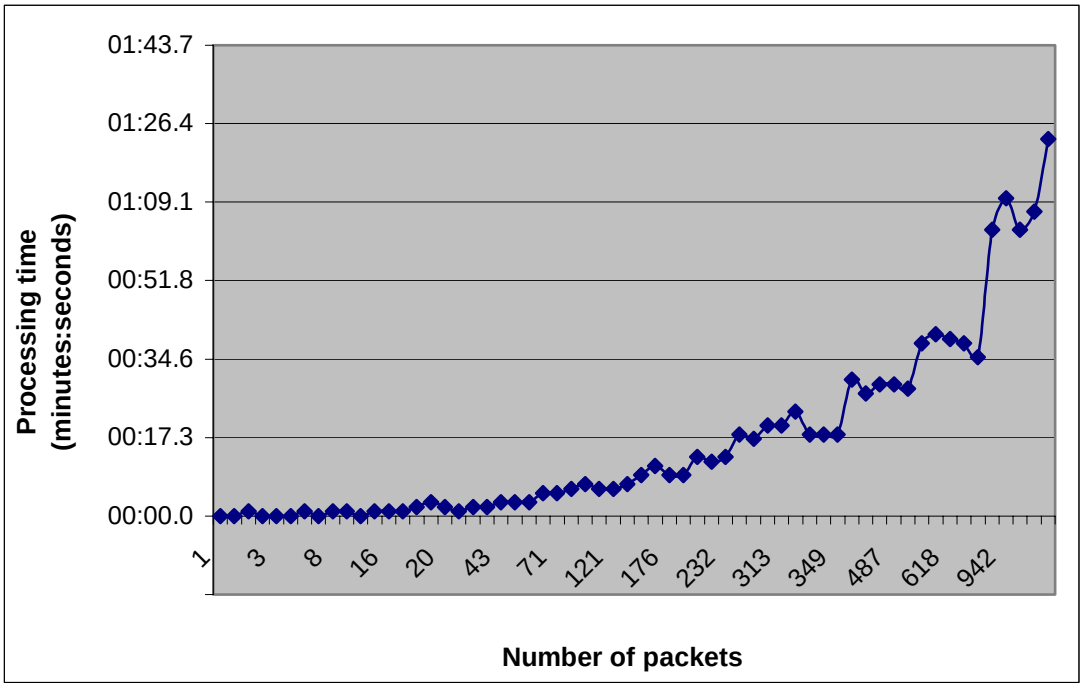


Figure 7.4: Processing time against buffer size

7.4.2 Timeout

Timeout is the other important factor in zero-based configuration. This parameter determines how long firewall need to maintain its open connections to particular external parties. Similarly, the appropriate time when the firewall needs to bring the configuration back to zero can be defined as well. Referring to Figure 7.1, timeout refers to $\Delta t = t_d - t_e$. Inaccurate definition of timeout will lead to two problems. First, shorter timeout causes the firewall to abort Internet transaction since canals are dropped before all packets are received. This problem causes denial of service. And second, longer time out causes the firewall to open the canals while no transactions are running. The second problem causes internal users being exposed to the outside. These problems emphasize the importance of accurate computation to define the timeout. Here the strategy to employ empirical rule to define the required time consumption in accessing external parties as formulated in Equation (7.2), produces dynamic timeouts that follow the speed of Internet transaction. From totally 148 transactions tested in this experiment, empirical measurements discover 3.38% of these transactions created denial of service due to shorter timeout. The measurements also discover the average exposed time as low as 67.1 seconds due to longer timeout.

7.4.3 Denial of Service

Denial of service becomes the critical factor in running zero-based configuration. In this method, denial of service can be introduced from two aspects i.e. shorter canals timeout and late opening canals. Since the former aspect has been discussed in Sub Section 7.4.2, here the discussion focuses in analyzing denial of service due to late opening of canals. And as has been stated in Sub Section 7.4.1, the speed of opening canals is influenced by existence of the transaction request in the traffic buffer. Meanwhile referring to Figure 7.1, the request to have Internet transaction is also limited by the timeout of contacting external parties t_{ex}. Hence denial of service is raised if $t_{re} > t_{ex}$. In this case, the possible cause of this problem

is the existence of access request that is appeared in the middle or at the end of traffic buffer, thus it requires longer time to establish canals, particularly if there is a bigger number of flowing packets are caught by the buffer. However experimental results in Appendix D2 show that this problem can be avoided due to the analysis of the flowing packets that is held only on the header of the network packet as shown in the foreground algorithm given in Figure 7.2. This strategy is proven capable to fasten packet analysis.

7.5 Summary

This chapter presents the implementation of the security strategy to reduce untrusted external parties. It is achieved by minimizing the available services at runtime and permitting the access to only authorized external parties. Zero-based configuration is introduced from this effort. Results of the experiment show that this method requires time consumption as high as 4.9 seconds in the average to open canal, and also create 3.38% denial of service due to shorter timeout of canals lifetime.

CHAPTER 7

RUNTIME PROCESS: MINIMIZING THE UNTRUSTED EXTERNAL PARTIES WITH ZERO-BASED CONFIGURATION

7.1 Introduction

This chapter presents the development of runtime process using minimal configuration of network services. The methods proposed in this stage is still to follow the initialization process presented in Chapter 4. Referring to the threats reduction strategy developed in Chapter 3, the method introduced in this chapter aims to minimize the untrusted external parties by limiting the available services at runtime, and restricting the access only to the authorized external parties. The experience of Ranum and Avolio (1994) to deal with firewall and network traffic motivates the formulation of this method. In their report (Ranum and Avolio, 1994), it is stated that turning network services at minimum would make firewall harder to break into. Hence it discloses the fact that current firewall technology offers too much services, particularly with static configuration built at start up and remained unchanged until the firewall is shut down. It is also worth to note that any security holes exist in the existing static configuration would be maintained along the operation of firewall. Moreover, Goncalves (2000) stressed the difficulties on auditing and administering the overall available network services at once due to the possibilities of unauthorized information flow that may rise from any of the open services without the knowledge of the internal user. Thus the idea behind zero-based configuration is to offer minimum services to secure Internet access by closing all

open connection in the idle time. This way, there will be no flowing traffic passing through the firewall without being screened first. It ensures that any Internet transactions are done with the trusted external parties.

The term “zero-based” it-self is taken from the financial sector, in which it has been used as the name of a budgeting method. In this field, zero-based stipulates the requirements that each appropriation in a budget year should justify the amount in excess of zero (Anthony, 2003). Therefore the basis of running this method is zero, and any expenditure must be rejustified during each budgeting cycle. In this research, the principle of zero-based approach is used to handle the runtime process of active firewall. Here configuration of network services at run time becomes the object of interest, thus minimizing this configuration become the objective of this method. Formulation is described in the next section.

7.2 Formulation

Let C become a set of reconfigureable canals establishing connection between intranet and the Internet. And $c(n,m) \in C$ denotes a set of canals connecting the internal user n to the requested external parties m, hence access to Internet using zero-based configuration is held by holding some activities as described using a time line graph depicted in Figure 7.1. Referring to this figure, if there exist an internal user requesting access to Internet at time $t = t_r$, the active firewall would establish the connection at time $t = t_e$ by activating $c(n,m)$, maintain the canals for the duration of Δt seconds, and then drop the canals at time $t = t_d$, with $t = t_e + \Delta t$. From this point, canals reconfiguration can be formulated as follows

$$c(n,m) = \begin{cases} 1 \Leftrightarrow (c(n,m) \in C \text{ for } m \in M) \wedge (t_e \le t \le t_d) \\ 0 \quad \text{otherwise} \end{cases} \tag{7.1}$$

with $m \in M$ become a set of external parties allowed to be contacted by the internal users. As mentioned previously that the established canals will be dropped after the duration of Δt seconds, hence Δt is defined as the timeout of network transaction, and for each transaction the timeout is determined using the following equation

$$\Delta t = \upsilon(m) + 3\tau(m) \tag{7.2}$$

with υ and τ denote the average mean and the standard deviation respectively of time consumption to conduct network transaction with the external parties corresponding to $c(n,m)$. In this method, prior measurement to empirically determine time consumption is required. It is worth to note that equation (7.2) follows the empirical rule of statistics with 99,7% approximation of time consumption (Triola, 2001).

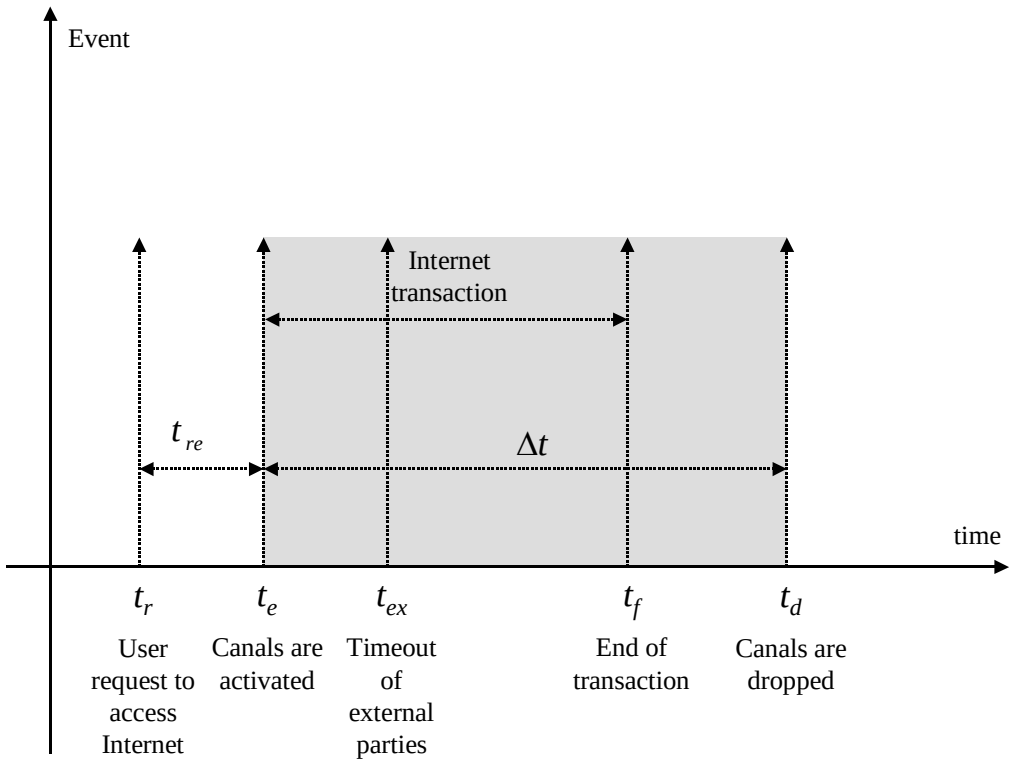


Figure 7.1: A series of events in accessing Internet using zero-based configuration

7.3 Implementation

Active firewall running zero-based configuration is implemented by developing the following mechanism:

(i) Running the close-condition approach for initialization process.

(ii) Waiting for the access request of the internal users at runtime, and for each permitted request, it is followed by enforcing the security rule in term of modifying the configuration of canals.

(iii) For each activated canal, the active firewall will monitor its lifetime. So when the lifetime reaches the timeout, the security rule will be dropped. Thus the zero-configuration of canals can be maintained along the operation of active firewall.

To implement the first mechanism above, the close-condition method developed in Chapter 4 is employed. And to implement the second mechanism, real-time traffic filtering is required to analyze every flowing packet intending to pass through the firewall. This mechanism is processing intensive, hence simplification of the analysis is necessary to achieve the real time process. Therefore only the header of network packet is analyzed. It is implemented by developing the foreground algorithm as depicted in Figure 7.2.

Meanwhile, to implement the third mechanism, prior measurement of time consumption for communicating with the external parties is developed. This data are necessary to compute the timeout of each transaction, thus the live time of each security rule can be determined using the empirical rule as formulated in Equation (7.2). This mechanism is implemented by developing the background algorithm as shown in Figure 7.3. Here the choice to run the dropping mechanism for each life canal in the background is due to the operation of the third mechanism above that less requires real-time execution compared to the enforcing security rule included in the second mechanism. Complete implementations of foreground and background algorithm are given in Appendix D4 and D5 respectively.

```
Procedure Foreground Zero Configuration
'TBUF   : array of traffic buffer
'C      : array of canals
'Q      : table of permitted external parties
Begin
   i = 1
   While i ≤ TBUF.size do
   Begin
        Line_content = ""
        j = 1
        Do Until (TBUF(i) = EOL) Or (i = TBUF.size)
            Line_content = Line_content + TBUF(i)
            If (TBUF(i) = " ") Then
            Begin
                Word(j) = Line_content
                Line_content = ""
                j = j + 1
            End
            i = i + 1
        Loop
        If Word = Packet.Header Then
        Begin
          For j = 1 to Word.size do
          Begin
                If Word(j) = External_party And External_party is new Then
                    External _ party ⇒ Array _ Outsider.External
                If Word(j) = Internal_user And Internal_user is new Then
                    Internal _ user ⇒ Array _ Outsider.Internal
                If Word(j) = Protocol Then Array_Outsider.Protocol = Word(j)
          End
          If Array_Outsider ≠ C Then
                If Array_Outsider.External ∈ Q Then Array_Outsider → C
        End
   End
End
```

Figure 7.2: Foreground algorithm

```
Procedure Background Zero Configuration
'C      : array of canals
'Q      : table of permitted external parties
Begin
   i = 1
   While i ≤ C.size do
   Begin
        For j = 1 to Q.size do
            If C(i).external = Q(j).external Then
                If (PresentTime() - C(i).start) > (Q(j).timeout + 3* Std_dev(Q(j).timeout)) Then
                Drop(C(i))
        i = i + 1
    End
End
```

Figure 7.3: Background algorithm

7.4 Experiment

Evaluation on the implemented zero-based configuration was conducted on the same experimental set up as in Chapter 4. A set of security policies used in the experiment was defined in term of a table of rules that contain the list of permitted Internet parties. The table is presented in Appendix D1. Using this table, firewall can decide whether the request to access Internet is permitted or not. The table also supplies the firewall with the information about the timeout period of each external party. Experimental results are presented in Appendix D2.

7.4.1 Speed to Open Canal

Since the firewall has to manage real time execution of Internet access, the speed of opening canals becomes the important factor influencing the implementation of zero-based configuration. Referring to Figure 7.1, this parameter deals with the establishment of t_e relative to t_r. Thus low speed to open canals most probably causes denial of service. In this implementation, the speed to open canal is affected by the location of Internet access request in the traffic buffer. If the request exists in the beginning of network traffic, the speed to open canal can be computed using the following equation $t_{re} = t_e - t_r$. But if the request appears in the middle of traffic buffer, then the speed to open canal is affected by the processing time of foreground algorithm to analyze the contents of buffer, in which iteration to browse the buffer is held. Here the processing time can be measured by computing the number of packets contained in the buffer. By assuming that the worst-case scenario is met i.e. the access request is appeared at the end of traffic buffer, using the computation to compare the processing time and the number of packets, the processing time to open canal can be determined as follow.

$$t_{re} = \left(\frac{s}{s_{ref}} \right)^n \Delta t_{ref} + k \qquad (7.3)$$

with n refers to the number of iteration, s is the buffer size in term of the number of packets, k is a constant representing the required time to complete the program excluding the main iteration and Δt is the processing time. Since the foreground algorithm depicted in Figure 7.2 produces n=1, thus linear processing time following the growth of packets in the buffer can be delivered. Therefore real time Internet access can be handled by this method.

Referring to the experimental results in Appendix D3, the graph presenting the increase of processing time compared to the number of packets is shown in Figure 7.4. It proves that Equation (7.3) is satisfied. Meanwhile, empirical measurement in Appendix D2 produces the average of 4.9 seconds with standard deviation 4.7 seconds are required by this algorithm to open canals.

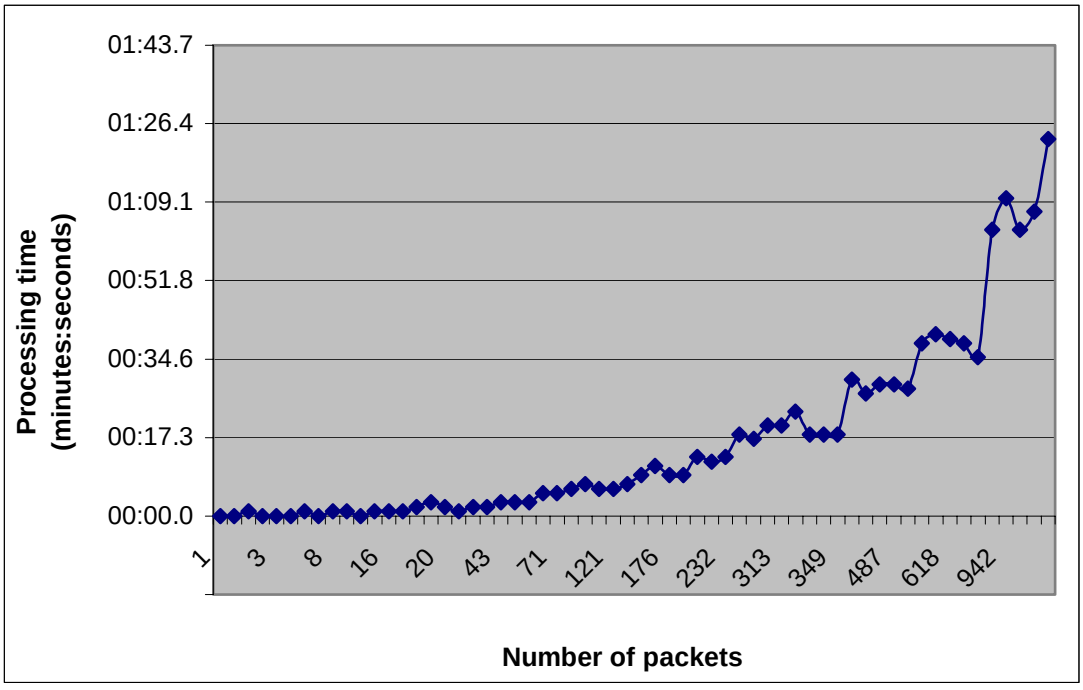


Figure 7.4: Processing time against buffer size

7.4.2 Timeout

Timeout is the other important factor in zero-based configuration. This parameter determines how long firewall need to maintain its open connections to particular external parties. Similarly, the appropriate time when the firewall needs to bring the configuration back to zero can be defined as well. Referring to Figure 7.1, timeout refers to $\Delta t = t_d - t_e$. Inaccurate definition of timeout will lead to two problems. First, shorter timeout causes the firewall to abort Internet transaction since canals are dropped before all packets are received. This problem causes denial of service. And second, longer time out causes the firewall to open the canals while no transactions are running. The second problem causes internal users being exposed to the outside. These problems emphasize the importance of accurate computation to define the timeout. Here the strategy to employ empirical rule to define the required time consumption in accessing external parties as formulated in Equation (7.2), produces dynamic timeouts that follow the speed of Internet transaction. From totally 148 transactions tested in this experiment, empirical measurements discover 3.38% of these transactions created denial of service due to shorter timeout. The measurements also discover the average exposed time as low as 67.1 seconds due to longer timeout.

7.4.3 Denial of Service

Denial of service becomes the critical factor in running zero-based configuration. In this method, denial of service can be introduced from two aspects i.e. shorter canals timeout and late opening canals. Since the former aspect has been discussed in Sub Section 7.4.2, here the discussion focuses in analyzing denial of service due to late opening of canals. And as has been stated in Sub Section 7.4.1, the speed of opening canals is influenced by existence of the transaction request in the traffic buffer. Meanwhile referring to Figure 7.1, the request to have Internet transaction is also limited by the timeout of contacting external parties t_{ex}. Hence denial of service is raised if $t_{re} > t_{ex}$. In this case, the possible cause of this problem

is the existence of access request that is appeared in the middle or at the end of traffic buffer, thus it requires longer time to establish canals, particularly if there is a bigger number of flowing packets are caught by the buffer. However experimental results in Appendix D2 show that this problem can be avoided due to the analysis of the flowing packets that is held only on the header of the network packet as shown in the foreground algorithm given in Figure 7.2. This strategy is proven capable to fasten packet analysis.

7.5 Summary

This chapter presents the implementation of the security strategy to reduce untrusted external parties. It is achieved by minimizing the available services at runtime and permitting the access to only authorized external parties. Zero-based configuration is introduced from this effort. Results of the experiment show that this method requires time consumption as high as 4.9 seconds in the average to open canal, and also create 3.38% denial of service due to shorter timeout of canals lifetime.

CHAPTER 8

RESEARCH EVALUATION

8.1 Introduction

After developing the strategies to activate the mechanism of firewall, now it comes to the stage of evaluating the proposed methods. This chapter presents the research evaluation on the developed firewall methods, in which security analysis on each firewall methods together with comparative study to no firewall, static rule, and dynamic configuration are held. To carry on running this task, the integrations between the initialization methods and runtime process are produced. The following combinations are delivered:

- Open-condition + Fuzzy-based security rules update (*OF*)
- Lattice-based initialization + Fuzzy-based security rules update (*LF*)
- Open-condition + Agent-based suspicious program detection (*OA*)
- Lattice-based initialization + Agent-based suspicious program detection (*LA*)
- Close-condition + Zero-configuration (*CZ*)

It is important to note that the combinations between the closed-conditions and the runtime methods other than zero-configuration cannot be produced, since the initialization process based on closed-condition requires a mechanism to open the canals for enabling Internet access. Meanwhile the combinations between the initialization processes other than closed-condition with zero-configuration cannot be

delivered as well due to the mechanism of zero-configuration to drop all available connections. Therefore only five listed-combinations above are evaluated.

8.2 Security Analysis

This study is conducted based on the requirements of firewall and its security considerations defined in RFC 2979 (Freed, 2000), the standard behavior of and the requirements for Internet firewall. As described in Section 2.4, RFC 2979 state that firewall must satisfy the transparency rule i.e. the introduction of firewall in a network environment must not cause any unintended failures nor preventing people from doing a useful works. The following parameters are defined to represent these requirements:

- The probability of available network services (*AS*)
- The probability of exposed line (*EL*)
- The probability of denial of service (*DS*)

Hence the combinations between the initialization and runtime process as listed above are investigated using these parameters. Assuming that each combination run in the same firewall machine with the same network environment, the security analysis is described as follow:

8.2.1 The Probability of Available Network Services

Here the availability of network services is defined as the available connection in responding to the request of accessing Internet. This parameter is computed using the following formula.

$$P(AS) = P(c(n,m) = 1 \mid f_{initialization}) \wedge P(c(n,m) = 1 \mid f_{runtime}) \quad (8.1)$$

To measure this parameter, basic mechanism of each method is observed as shown in Equation (8.1). Table 8.1 is delivered as the product of this observation. Detail calculation is given in Appendix E1. This table discloses the following fact.

$$P(AS-OA) = P(AS-OF) \geq P(AS-LF) = P(AS-LA) \geq P(AS-CZ) \quad (8.2)$$

Proof. Assuming that evaluation is conducted in the normal condition therefore no threat exist, and $P(th(n)=0)=1$. And referring to the use of canals, $\forall c(n,m)=1$ means all connection to the Internet are opened, thus $P(c(n,m)=1)=1$ is produced. Meanwhile $\forall c(n,m) \in C=1$ produces $P(c(n,m)=1)=[0,1]$ since $\exists c(n,m) \notin C$ exist. So it can be concluded that

$$P(th(n)=0) = P(c(n,m)=1) \geq P(c(n,m)=1 \mid \forall c(n,m) \in C=1) \quad (8.3)$$

Thus Equation (8.3) proves Equation (8.2). Here $P(AS-CZ)$ has the lowest rank since $P(c(n,m)=1 \mid \forall c(n,m) \in C=1)$ involves time variables $t_e \leq t \leq t_d$ that limits the availability of services.

Tables 8.1: Result of measuring the availability of network services

Methods	***P(AS)***
Open-condition + Fuzzy-based security rules update (*OF*)	1
Open-condition + Agent-based suspicious program detection (*OA*)	1
Lattice-based initialization + Fuzzy-based security rules update (*LF*)	$P(c(n,m)=1 \mid \forall c(n,m) \in C=1 \Leftrightarrow protection(n) \geq risk(m))$
Lattice-based initialization + Agent-based suspicious program detection (*LA*)	$P(c(n,m)=1 \mid \forall c(n,m) \in C=1 \Leftrightarrow protection(n) \geq risk(m))$
Close-condition + Zero-configuration (*CZ*)	$P(c(n,m)=1 \mid \forall c(n,m) \in C=1 \Leftrightarrow t_e \leq t < t_d)$

8.2.2 Probability of Exposed Line

Exposed line is defined as the status of firewall when the opened connection exist, while there is no user request to access the Internet, or there is the appearance of Internet threats. This parameter is computed using the following formula:

$$P(EL) = P(c(n,m) = 1 \mid th(n) > 0 \vee protection(n) < risk(m) \vee t < t_e \vee t \geq t_d) \quad (8.4)$$

Computing the exposed line to the combination of initialization method and runtime process as listed in Section 8.1 produces the data as shown in Table 8.2. Detail calculations to obtain these data are given in Appendix E2.

Table 8.2: Result of measuring the probability of exposed line

Methods	***P(EL)***
Open-condition + Fuzzy-based security rules update (*OF*)	• For $t < t_r$ or $t > t_{ft}$, $P(EL-OF) = 1$ • For $th(n) > 0$, $P(EL-OF) = 1$ • For $protection(n) < risk(m)$, $P(EL-OF) = 0$
Open-condition + Agent-based suspicious program detection (*OA*)	• For $t < t_r$ or $t > t_{ft}$, $P(EL-OA) = 1$ • For $th(n) > 0$, $P(EL-OA) = 0$ • For $protection(n) < risk(m)$, $P(EL-OA) = 1$
Lattice-based initialization + Fuzzy-based security rules update (*LF*)	• For $t < t_r$ or $t > t_{ft}$, $P(EL-LF) = P(c(n,m) = 1 \mid \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m))$ • For $th(n) > 0$, $P(EL-LF) = P(c(n,m) = 1 \mid \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m))$ • For $protection(n) < risk(m)$, $P(EL-LF) = 0$
Lattice-based initialization + Agent-based suspicious program detection (*LA*)	• For $t < t_r$ or $t > t_{ft}$, $P(EL-LA) = P(c(n,m) = 1 \mid \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m))$ • For $th(n) > 0$, $P(EL-LA) = 0$ • For $protection(n) < risk(m)$, $P(EL-LA) = 0$
Close-condition + Zero-configuration (*CZ*)	• For $t < t_r$ or $t > t_{ft}$, $P(EL-CZ) = 0$ • For $th(n) > 0$, $P(EL-CZ) = P(c(n,m) = 1 \mid \forall c(n,m) \in C = 1 \Leftrightarrow t_e \leq t < t_d)$ • For $protection(n) < risk(m)$, $P(EL-CZ) = P(c(n,m) = 1 \mid \forall c(n,m) \in C = 1 \Leftrightarrow t_e \leq t < t_d)$

Table 8.2 show that the combination of close-condition and zero-configuration provides the best protection in the idle time, in which zero exposed line can be delivered by this method. However this method is insensitive to the appearance of the threat and the raising risk of external parties, which the combination of lattice-initialization and agent-based malicious program detection can be best countering these attacks although exposed-line is produced in the idle time. And among the combination above, the methods employing open-condition and any runtime process notably creates more exposed-line compared to other methods. These methods produce more total open connection in the idle time as well as in responding of the raising risk or the appearance of the threats. Thus the combinations of the open-condition and fuzzy-based or agent-based runtime process are considered delivering the worst performance in preventing the exposed-line. Based on these conditions, the performance for protecting intranet from exposed line can be ranked as follow:

$$P(EL-LA) \le P(EL-CZ) \le P(EL-LF) \le P(EL-OF) = P(EL-OA) \qquad (8.5)$$

8.2.3 Probability of Denial of Service

This parameter discloses the possibility of denial of services introduced by each method. It is important to note that the denial of services only takes account of the authorized access to external parties. Therefore the denial to unauthorized access due to the enforcement of security rule is excluded. This assumption eliminates three conditions preventing the Internet access i.e. the appearance of threats in the internal users ($th(n) > 0$), the increasing risk of external parties that exceeds the protection level of internal users ($protection(n) < risk(m)$), and the condition when there is no request to access Internet ($t < t_r \vee t > t_{ft}$). To measure the probability of denial of service, the following formula is computed.

$$P(DS) = P(c(n,m) = 0 \mid f_{initialization}) \vee P(c(n,m) = 0 \mid f_{runtime}) \qquad (8.6)$$

Computing Equation (8.6) to the listed combination in Section 8.1 produces the data as shown in Table 8.3. Detail calculations to get these data are given in Appendix E.3.

Table 8.3: Result produced by the probability of denial of service

Methods	***P(DS)***
Open-condition + Fuzzy-based security rules update (*OF*)	0
Open-condition + Agent-based suspicious program detection (*OA*)	0
Lattice-based initialization + Fuzzy-based security rules update (*LF*)	$P(c(n,m)=0 \mid \forall c(n,m) \in C=1 \Leftrightarrow \ protection(n) \geq risk(m))$
Lattice-based initialization + Agent-based suspicious program detection (*LA*)	$P(c(n,m)=0 \mid \forall c(n,m) \in C=1 \Leftrightarrow \ protection(n) \geq risk(m))$
Close-condition + Zero-configuration (*CZ*)	$P(c(n,m)=0 \mid \forall c(n,m) \in C=1 \Leftrightarrow t_e \leq t < t_d)$

Table 8.3 discloses the fact that the combination of close-condition and zero-configuration most suffer from denial of service. This condition is due to the mechanism of this method to actively open and close canals in responding to the user request, thus $t_e \leq t < t_d$ become the dominant factor increasing denial of service. Therefore the speed to establish canals and accurate prediction to the lifetime of Internet transaction are critical. The approach to only inspect the header of network packet for establishing canals, and to employ empirical rules with 99.7% approximation to compute the timeout, seem become the correct method toward implementing zero configuration. This approach will not let any factor preventing Internet transactions. Second rank for suffering denial of service is the method consisting of lattice-based initialization. It is due to the factor of $\forall c(n,m) \in C=1$ that may introduce $\exists c(n,m) \notin C$. However if C is large enough to hold a huge collection of the identities of external parties, denial of service can be minimized. Referring to Table 8.3, probability of denial of service can be ranked as follow.

$$P(DS-CZ) \geq P(DS-LF) = P(DS-LA) \geq P(DS-OA) = P(DS-OF) \quad (8.7)$$

8.3 Comparative Study

To further evaluate the developed active firewall systems, comparison to no firewall, static, and dynamic configuration are held. The choice to use these configurations is due to proven capabilities and reliabilities of these methods to provide connection to the Internet (Reumann *et al.*, 2001; Toth and Kruegel 2002). For static configuration, a comprehensive firewall script developed by Bob Sully is used (Sully, 2005). This script was originally designed and implemented by Craig Zeller using ipfwadm (Zeller, 2004), but then it is translated into ipchains and iptables with some additions and modifications. Here the iptables version of this script is chosen for this study since it delivers more complete protection to the intranet. The script is presented in Appendix F1. Meanwhile, a concise firewall script developed by Robbins (2001) is used for dynamic configuration. This script as presented in Appendix F2, is capable to deter the flowing malicious packets. The results of measuring these configurations using the parameters defined in Section 8.2 are presented in Table 8.4. Detail calculations to obtain these data are given in Appendix F3, F4 and F5 for no firewall, static, and dynamic firewall configuration respectively.

Using the data in Table 8.1 to 8.4, analysis is described as follow. Probability of available services obtained from no firewall configuration equals to the available services obtained from open-condition combined with agent-based detection. While for static firewall configuration, the available services totally depend on the manual definition of the permitted Internet access. If it is assumed that static configuration is capable to provide appropriate connection to the Internet, hence $P(AS-SF)=1$. The same assumption can be applied to dynamic configuration as well, therefore $P(AS-DF)=1$. Thus based on the probability of available services, the performance of each method can be ranked as follow:

$$NF = SF = DF = OA = OF \geq LF = LA \geq CZ \qquad (8.8)$$

For exposed line, the open connection produces probability of exposed line as big as all available services are exposed. It means the whole internal users can be

seen and easily compromised from outside. Meanwhile static configuration performs better since it is influenced by manual configuration established at start up. However if the same assumptions as measuring available services above are applied, then static configuration produces the same exposed line as no firewall configuration. It is due to the character of this configuration that insensitive to any changes of network condition, including the raising threats. And for dynamic configuration, the raising threats can be detected although it is very limited, thus the connections can be modified. Ranking each method based on the probability of exposed line produces:

$$NF = SF = OA = OF \geq DF \cong LF \geq CZ \geq LA \tag{8.9}$$

While for denial of services, no firewall configuration produces zero probability since it is merely an open connection. The similar probability is delivered by static and dynamic configurations if the assumption for defining available services is applied again to compute this parameter. However both methods are influenced by manual definition to allow or reject particular external parties built at start up. Thus ranking of each method based on the probability of denial of services produces:

$$NF = OF = OA \leq SF \cong DF \cong LF = LA \leq CZ \tag{8.10}$$

Equation (8.8) discloses the cost for establishing active firewall mechanism i.e. lower service availability can be provided by the proposed active firewalls compared to the services provided by the existing methods. Only open-condition combined with agent-based that is capable to compete with the existing methods. However for denial of services, only close-condition combined with zero configurations suffer from this problem worse than the existing methods as stated in Equation (8.10). Other active firewall methods have similar performance as no firewall, static or dynamic configuration. And finally equation (8.9) proves that all active firewall methods outperform no firewall or static firewall configuration in providing network security in which smaller exposed line can be delivered. While for the existing methods based on dynamic configuration, only active firewalls based on the combination involving open-condition that cannot outperform the dynamic configuration.

Table 8.4: *P(AS), P(EL)* and *P(DS)* of no firewall, static and dynamic configuration

Methods	Parameters
No Firewall Configuration (*NF*)	$P(AS-NF)=1$
	• For $t<t_r$ or $t>t_{ft}$, $P(EL-NF)=1$ • For $th(n)>0$, $P(EL-NF)=1$ • For $protection(n)<risk(m)$, $P(EL-NF)=1$
	$P(DS-NF)=0$
Static Firewall Configuration (*SF*)	$P(AS-SF)=P(c(n,m)=1 \mid \exists c(n,m)=1+\exists c(n,m)=0)$
	• For $t<t_r$ or $t>t_{ft}$, $P(EL-SF)=P(c(n,m)=1 \mid \exists c(n,m)=1+\exists c(n,m)=0)$ • For $th(n)>0$, $P(EL-SF)=P(c(n,m)=1 \mid \exists c(n,m)=1+\exists c(n,m)=0)$ • For $protection(n)<risk(m)$, $P(EL-SF)=P(c(n,m)=1 \mid \exists c(n,m)=1+\exists c(n,m)=0)$
	$P(DS-SF)=P(c(n,m)=0 \mid \exists c(n,m)=1+\exists c(n,m)=0)$
Dynamic Firewall Configuration (*DF*)	$P(AS-DF)=P(c(n,m)=1 \mid \exists c(n,m)=1+\exists c(n,m)=0)$
	• For $t<t_r$ or $t>t_{ft}$, $P(EL-DF)=P(c(n,m)=1 \mid \exists c(n,m)=1+\exists c(n,m)=0)$ • For $th(n)>0$, $P(EL-DF)=0$ • For $protection(n)<risk(m)$, $P(EL-DF)=P(c(n,m)=1 \mid \exists c(n,m)=1+\exists c(n,m)=0)$
	$P(DS-DF)=P(c(n,m)=0 \mid \exists c(n,m)=1+\exists c(n,m)=0)$

Further evaluation on the results of benchmarking is held by ranking each method based on Equation (8.8), (8.9) and (8.10), and then give the value from zero to ten with zero represents the best performance while ten is the worst. The methods having performance in between the best and the worst are assigned values that are equally distributed between zero and ten. Table 8.5 shows this evaluation. The overall performance is obtained by summing up the values of all parameters obtained

by each method. Here the lower value represents better active firewall method in providing intranet protection. The results sorted from the best method to the worst are given below:

(i) *NF, OA, OF, LA* = 10
(ii) *DF* = 11,66
(iii) *SF* = 15
(iv) *LF* = 16.66
(v) *CZ* = 23.33

The list above shows that *OA*, *OF*, and *LA* outperform other developed active firewall methods. However if we refer to the probability of exposed line among these three methods, it shows that *LA* outperforms other methods. Meanwhile *CZ* most probably suffer from usability since this method perform worst in providing network services as well as suffering from denial of service. This condition is due to the mechanism of *CZ* to always drop any available services at runtime.

Table 8.5: Ranking the performance of all methods

Parameter	**The Best Performance (0)**	**The Performance in Between (0 – 10)**		**The Worst Performance (10)**
Available Service	*NF, SF, DF, OA, OF*	{*LF*, ***LA***} = 5		*CZ*
Exposed Line	***LA***	*CZ* = 3.33	{*DF, LF*} = 6.66	*NF, SF, OA, OF*
Denial of Service	*NF, OA, OF*	{*SF, DF, LF*, ***LA***} = 5		*CZ*

8.4 Summary

Evaluations on the proposed methods for developing active firewall are presented in this chapter, in which it consists of two stages i.e. security analysis and comparative study against the known firewall methods. To evaluate the complete mechanisms of active firewall, the developed initialization and runtime process are combined. Five different active firewall methods are produced as shown in Section 8.1. Security analysis is held based on RFC 2979 (Freed, 2000), in which three parameters i.e. probability of available service, probability of exposed line and probability of denial of service are defined to represent the transparency rule of this standard. And to do the comparative study, each developed active firewall method is compared with each other, and also to the existing methods i.e. no firewall, static, and dynamic configurations. This study is also held based on the parameters defined for security analysis. Results of this effort show that the developed active firewalls are capable to combat the present of Internet threats. And lattice-initialization combined with agent-based module (*LA*) is proven having better security and usability among other developed methods, and also outperforms the existing methods.

CHAPTER 8

RESEARCH EVALUATION

8.1 Introduction

After developing the strategies to activate the mechanism of firewall, now it comes to the stage of evaluating the proposed methods. This chapter presents the research evaluation on the developed firewall methods, in which security analysis on each firewall methods together with comparative study to no firewall, static rule, and dynamic configuration are held. To carry on running this task, the integrations between the initialization methods and runtime process are produced. The following combinations are delivered:

- Open-condition + Fuzzy-based security rules update (*OF*)
- Lattice-based initialization + Fuzzy-based security rules update (*LF*)
- Open-condition + Agent-based suspicious program detection (*OA*)
- Lattice-based initialization + Agent-based suspicious program detection (*LA*)
- Close-condition + Zero-configuration (*CZ*)

It is important to note that the combinations between the closed-conditions and the runtime methods other than zero-configuration cannot be produced, since the initialization process based on closed-condition requires a mechanism to open the canals for enabling Internet access. Meanwhile the combinations between the initialization processes other than closed-condition with zero-configuration cannot be

delivered as well due to the mechanism of zero-configuration to drop all available connections. Therefore only five listed-combinations above are evaluated.

8.2 Security Analysis

This study is conducted based on the requirements of firewall and its security considerations defined in RFC 2979 (Freed, 2000), the standard behavior of and the requirements for Internet firewall. As described in Section 2.4, RFC 2979 state that firewall must satisfy the transparency rule i.e. the introduction of firewall in a network environment must not cause any unintended failures nor preventing people from doing a useful works. The following parameters are defined to represent these requirements:

- The probability of available network services (*AS*)
- The probability of exposed line (*EL*)
- The probability of denial of service (*DS*)

Hence the combinations between the initialization and runtime process as listed above are investigated using these parameters. Assuming that each combination run in the same firewall machine with the same network environment, the security analysis is described as follow:

8.2.1 The Probability of Available Network Services

Here the availability of network services is defined as the available connection in responding to the request of accessing Internet. This parameter is computed using the following formula.

$$P(AS) = P(c(n,m) = 1 \mid f_{initialization}) \wedge P(c(n,m) = 1 \mid f_{runtime}) \quad (8.1)$$

To measure this parameter, basic mechanism of each method is observed as shown in Equation (8.1). Table 8.1 is delivered as the product of this observation. Detail calculation is given in Appendix E1. This table discloses the following fact.

$$P(AS-OA) = P(AS-OF) \geq P(AS-LF) = P(AS-LA) \geq P(AS-CZ) \quad (8.2)$$

Proof. Assuming that evaluation is conducted in the normal condition therefore no threat exist, and $P(th(n)=0)=1$. And referring to the use of canals, $\forall c(n,m)=1$ means all connection to the Internet are opened, thus $P(c(n,m)=1)=1$ is produced. Meanwhile $\forall c(n,m) \in C=1$ produces $P(c(n,m)=1)=[0,1]$ since $\exists c(n,m) \notin C$ exist. So it can be concluded that

$$P(th(n)=0) = P(c(n,m)=1) \geq P(c(n,m)=1 \mid \forall c(n,m) \in C=1) \quad (8.3)$$

Thus Equation (8.3) proves Equation (8.2). Here $P(AS-CZ)$ has the lowest rank since $P(c(n,m)=1 \mid \forall c(n,m) \in C=1)$ involves time variables $t_e \leq t \leq t_d$ that limits the availability of services.

Tables 8.1: Result of measuring the availability of network services

Methods	***P(AS)***
Open-condition + Fuzzy-based security rules update (*OF*)	1
Open-condition + Agent-based suspicious program detection (*OA*)	1
Lattice-based initialization + Fuzzy-based security rules update (*LF*)	$P(c(n,m)=1 \mid \forall c(n,m) \in C=1 \Leftrightarrow protection(n) \geq risk(m))$
Lattice-based initialization + Agent-based suspicious program detection (*LA*)	$P(c(n,m)=1 \mid \forall c(n,m) \in C=1 \Leftrightarrow protection(n) \geq risk(m))$
Close-condition + Zero-configuration (*CZ*)	$P(c(n,m)=1 \mid \forall c(n,m) \in C=1 \Leftrightarrow t_e \leq t < t_d)$

8.2.2 Probability of Exposed Line

Exposed line is defined as the status of firewall when the opened connection exist, while there is no user request to access the Internet, or there is the appearance of Internet threats. This parameter is computed using the following formula:

$$P(EL) = P(c(n,m) = 1 \mid th(n) > 0 \vee protection(n) < risk(m) \vee t < t_e \vee t \geq t_d) \quad (8.4)$$

Computing the exposed line to the combination of initialization method and runtime process as listed in Section 8.1 produces the data as shown in Table 8.2. Detail calculations to obtain these data are given in Appendix E2.

Table 8.2: Result of measuring the probability of exposed line

Methods	***P(EL)***
Open-condition + Fuzzy-based security rules update (*OF*)	• For $t < t_r$ or $t > t_{ft}$, $P(EL-OF) = 1$ • For $th(n) > 0$, $P(EL-OF) = 1$ • For $protection(n) < risk(m)$, $P(EL-OF) = 0$
Open-condition + Agent-based suspicious program detection (*OA*)	• For $t < t_r$ or $t > t_{ft}$, $P(EL-OA) = 1$ • For $th(n) > 0$, $P(EL-OA) = 0$ • For $protection(n) < risk(m)$, $P(EL-OA) = 1$
Lattice-based initialization + Fuzzy-based security rules update (*LF*)	• For $t < t_r$ or $t > t_{ft}$, $P(EL-LF) = P(c(n,m) = 1 \mid \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m))$ • For $th(n) > 0$, $P(EL-LF) = P(c(n,m) = 1 \mid \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m))$ • For $protection(n) < risk(m)$, $P(EL-LF) = 0$
Lattice-based initialization + Agent-based suspicious program detection (*LA*)	• For $t < t_r$ or $t > t_{ft}$, $P(EL-LA) = P(c(n,m) = 1 \mid \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m))$ • For $th(n) > 0$, $P(EL-LA) = 0$ • For $protection(n) < risk(m)$, $P(EL-LA) = 0$
Close-condition + Zero-configuration (*CZ*)	• For $t < t_r$ or $t > t_{ft}$, $P(EL-CZ) = 0$ • For $th(n) > 0$, $P(EL-CZ) = P(c(n,m) = 1 \mid \forall c(n,m) \in C = 1 \Leftrightarrow t_e \leq t < t_d)$ • For $protection(n) < risk(m)$, $P(EL-CZ) = P(c(n,m) = 1 \mid \forall c(n,m) \in C = 1 \Leftrightarrow t_e \leq t < t_d)$

Table 8.2 show that the combination of close-condition and zero-configuration provides the best protection in the idle time, in which zero exposed line can be delivered by this method. However this method is insensitive to the appearance of the threat and the raising risk of external parties, which the combination of lattice-initialization and agent-based malicious program detection can be best countering these attacks although exposed-line is produced in the idle time. And among the combination above, the methods employing open-condition and any runtime process notably creates more exposed-line compared to other methods. These methods produce more total open connection in the idle time as well as in responding of the raising risk or the appearance of the threats. Thus the combinations of the open-condition and fuzzy-based or agent-based runtime process are considered delivering the worst performance in preventing the exposed-line. Based on these conditions, the performance for protecting intranet from exposed line can be ranked as follow:

$$P(EL-LA) \le P(EL-CZ) \le P(EL-LF) \le P(EL-OF) = P(EL-OA) \qquad (8.5)$$

8.2.3 Probability of Denial of Service

This parameter discloses the possibility of denial of services introduced by each method. It is important to note that the denial of services only takes account of the authorized access to external parties. Therefore the denial to unauthorized access due to the enforcement of security rule is excluded. This assumption eliminates three conditions preventing the Internet access i.e. the appearance of threats in the internal users ($th(n) > 0$), the increasing risk of external parties that exceeds the protection level of internal users ($protection(n) < risk(m)$), and the condition when there is no request to access Internet ($t < t_r \vee t > t_{ft}$). To measure the probability of denial of service, the following formula is computed.

$$P(DS) = P(c(n,m) = 0 \mid f_{initialization}) \vee P(c(n,m) = 0 \mid f_{runtime}) \qquad (8.6)$$

Computing Equation (8.6) to the listed combination in Section 8.1 produces the data as shown in Table 8.3. Detail calculations to get these data are given in Appendix E.3.

Table 8.3: Result produced by the probability of denial of service

Methods	***P(DS)***
Open-condition + Fuzzy-based security rules update (*OF*)	0
Open-condition + Agent-based suspicious program detection (*OA*)	0
Lattice-based initialization + Fuzzy-based security rules update (*LF*)	$P(c(n,m)=0 \mid \forall c(n,m) \in C=1 \Leftrightarrow \ protection(n) \geq risk(m))$
Lattice-based initialization + Agent-based suspicious program detection (*LA*)	$P(c(n,m)=0 \mid \forall c(n,m) \in C=1 \Leftrightarrow \ protection(n) \geq risk(m))$
Close-condition + Zero-configuration (*CZ*)	$P(c(n,m)=0 \mid \forall c(n,m) \in C=1 \Leftrightarrow t_e \leq t < t_d)$

Table 8.3 discloses the fact that the combination of close-condition and zero-configuration most suffer from denial of service. This condition is due to the mechanism of this method to actively open and close canals in responding to the user request, thus $t_e \leq t < t_d$ become the dominant factor increasing denial of service. Therefore the speed to establish canals and accurate prediction to the lifetime of Internet transaction are critical. The approach to only inspect the header of network packet for establishing canals, and to employ empirical rules with 99.7% approximation to compute the timeout, seem become the correct method toward implementing zero configuration. This approach will not let any factor preventing Internet transactions. Second rank for suffering denial of service is the method consisting of lattice-based initialization. It is due to the factor of $\forall c(n,m) \in C=1$ that may introduce $\exists c(n,m) \notin C$. However if C is large enough to hold a huge collection of the identities of external parties, denial of service can be minimized. Referring to Table 8.3, probability of denial of service can be ranked as follow.

$$P(DS-CZ) \geq P(DS-LF) = P(DS-LA) \geq P(DS-OA) = P(DS-OF) \quad (8.7)$$

8.3 Comparative Study

To further evaluate the developed active firewall systems, comparison to no firewall, static, and dynamic configuration are held. The choice to use these configurations is due to proven capabilities and reliabilities of these methods to provide connection to the Internet (Reumann *et al.*, 2001; Toth and Kruegel 2002). For static configuration, a comprehensive firewall script developed by Bob Sully is used (Sully, 2005). This script was originally designed and implemented by Craig Zeller using ipfwadm (Zeller, 2004), but then it is translated into ipchains and iptables with some additions and modifications. Here the iptables version of this script is chosen for this study since it delivers more complete protection to the intranet. The script is presented in Appendix F1. Meanwhile, a concise firewall script developed by Robbins (2001) is used for dynamic configuration. This script as presented in Appendix F2, is capable to deter the flowing malicious packets. The results of measuring these configurations using the parameters defined in Section 8.2 are presented in Table 8.4. Detail calculations to obtain these data are given in Appendix F3, F4 and F5 for no firewall, static, and dynamic firewall configuration respectively.

Using the data in Table 8.1 to 8.4, analysis is described as follow. Probability of available services obtained from no firewall configuration equals to the available services obtained from open-condition combined with agent-based detection. While for static firewall configuration, the available services totally depend on the manual definition of the permitted Internet access. If it is assumed that static configuration is capable to provide appropriate connection to the Internet, hence $P(AS-SF)=1$. The same assumption can be applied to dynamic configuration as well, therefore $P(AS-DF)=1$. Thus based on the probability of available services, the performance of each method can be ranked as follow:

$$NF = SF = DF = OA = OF \geq LF = LA \geq CZ \qquad (8.8)$$

For exposed line, the open connection produces probability of exposed line as big as all available services are exposed. It means the whole internal users can be

seen and easily compromised from outside. Meanwhile static configuration performs better since it is influenced by manual configuration established at start up. However if the same assumptions as measuring available services above are applied, then static configuration produces the same exposed line as no firewall configuration. It is due to the character of this configuration that insensitive to any changes of network condition, including the raising threats. And for dynamic configuration, the raising threats can be detected although it is very limited, thus the connections can be modified. Ranking each method based on the probability of exposed line produces:

$$NF = SF = OA = OF \geq DF \cong LF \geq CZ \geq LA \tag{8.9}$$

While for denial of services, no firewall configuration produces zero probability since it is merely an open connection. The similar probability is delivered by static and dynamic configurations if the assumption for defining available services is applied again to compute this parameter. However both methods are influenced by manual definition to allow or reject particular external parties built at start up. Thus ranking of each method based on the probability of denial of services produces:

$$NF = OF = OA \leq SF \cong DF \cong LF = LA \leq CZ \tag{8.10}$$

Equation (8.8) discloses the cost for establishing active firewall mechanism i.e. lower service availability can be provided by the proposed active firewalls compared to the services provided by the existing methods. Only open-condition combined with agent-based that is capable to compete with the existing methods. However for denial of services, only close-condition combined with zero configurations suffer from this problem worse than the existing methods as stated in Equation (8.10). Other active firewall methods have similar performance as no firewall, static or dynamic configuration. And finally equation (8.9) proves that all active firewall methods outperform no firewall or static firewall configuration in providing network security in which smaller exposed line can be delivered. While for the existing methods based on dynamic configuration, only active firewalls based on the combination involving open-condition that cannot outperform the dynamic configuration.

Table 8.4: *P(AS), P(EL)* and *P(DS)* of no firewall, static and dynamic configuration

Methods	Parameters
No Firewall Configuration (*NF*)	$P(AS-NF)=1$
	• For $t<t_r$ or $t>t_{ft}$, $P(EL-NF)=1$ • For $th(n)>0$, $P(EL-NF)=1$ • For $protection(n)<risk(m)$, $P(EL-NF)=1$
	$P(DS-NF)=0$
Static Firewall Configuration (*SF*)	$P(AS-SF)=P(c(n,m)=1 \mid \exists c(n,m)=1+\exists c(n,m)=0)$
	• For $t<t_r$ or $t>t_{ft}$, $P(EL-SF)=P(c(n,m)=1 \mid \exists c(n,m)=1+\exists c(n,m)=0)$ • For $th(n)>0$, $P(EL-SF)=P(c(n,m)=1 \mid \exists c(n,m)=1+\exists c(n,m)=0)$ • For $protection(n)<risk(m)$, $P(EL-SF)=P(c(n,m)=1 \mid \exists c(n,m)=1+\exists c(n,m)=0)$
	$P(DS-SF)=P(c(n,m)=0 \mid \exists c(n,m)=1+\exists c(n,m)=0)$
Dynamic Firewall Configuration (*DF*)	$P(AS-DF)=P(c(n,m)=1 \mid \exists c(n,m)=1+\exists c(n,m)=0)$
	• For $t<t_r$ or $t>t_{ft}$, $P(EL-DF)=P(c(n,m)=1 \mid \exists c(n,m)=1+\exists c(n,m)=0)$ • For $th(n)>0$, $P(EL-DF)=0$ • For $protection(n)<risk(m)$, $P(EL-DF)=P(c(n,m)=1 \mid \exists c(n,m)=1+\exists c(n,m)=0)$
	$P(DS-DF)=P(c(n,m)=0 \mid \exists c(n,m)=1+\exists c(n,m)=0)$

Further evaluation on the results of benchmarking is held by ranking each method based on Equation (8.8), (8.9) and (8.10), and then give the value from zero to ten with zero represents the best performance while ten is the worst. The methods having performance in between the best and the worst are assigned values that are equally distributed between zero and ten. Table 8.5 shows this evaluation. The overall performance is obtained by summing up the values of all parameters obtained

by each method. Here the lower value represents better active firewall method in providing intranet protection. The results sorted from the best method to the worst are given below:

(i) *NF, OA, OF, LA* = 10
(ii) *DF* = 11,66
(iii) *SF* = 15
(iv) *LF* = 16.66
(v) *CZ* = 23.33

The list above shows that *OA*, *OF*, and *LA* outperform other developed active firewall methods. However if we refer to the probability of exposed line among these three methods, it shows that *LA* outperforms other methods. Meanwhile *CZ* most probably suffer from usability since this method perform worst in providing network services as well as suffering from denial of service. This condition is due to the mechanism of *CZ* to always drop any available services at runtime.

Table 8.5: Ranking the performance of all methods

Parameter	**The Best Performance (0)**	**The Performance in Between (0 – 10)**		**The Worst Performance (10)**
Available Service	*NF, SF, DF, OA, OF*	{*LF*, ***LA***} = 5		*CZ*
Exposed Line	***LA***	*CZ* = 3.33	{*DF, LF*} = 6.66	*NF, SF, OA, OF*
Denial of Service	*NF, OA, OF*	{*SF, DF, LF*, ***LA***} = 5		*CZ*

8.4 Summary

Evaluations on the proposed methods for developing active firewall are presented in this chapter, in which it consists of two stages i.e. security analysis and comparative study against the known firewall methods. To evaluate the complete mechanisms of active firewall, the developed initialization and runtime process are combined. Five different active firewall methods are produced as shown in Section 8.1. Security analysis is held based on RFC 2979 (Freed, 2000), in which three parameters i.e. probability of available service, probability of exposed line and probability of denial of service are defined to represent the transparency rule of this standard. And to do the comparative study, each developed active firewall method is compared with each other, and also to the existing methods i.e. no firewall, static, and dynamic configurations. This study is also held based on the parameters defined for security analysis. Results of this effort show that the developed active firewalls are capable to combat the present of Internet threats. And lattice-initialization combined with agent-based module (*LA*) is proven having better security and usability among other developed methods, and also outperforms the existing methods.

CHAPTER 9

CONCLUSION

An effort to combat Internet threats has been presented, with the purpose is to provide intranet protection in accessing Internet. It is achieved by activating the mechanism of firewall, the most deployed security tool for securing information systems (CERT, 2004). In this study, the security strategies to eliminate Internet threats are developed, and then are used as the foundation for activating the mechanism of network firewall. Meanwhile, the model of active firewall is formulated with the concept of canalisation to represent the connections of internal users to external parties. And the operation of active firewall is formulated into two main stages i.e. initialisation and runtime process. To handle the initialisation, three methods are developed i.e. close-condition, open-condition, and lattice-based. While for the runtime process, three methods are also developed i.e. adaptive security rules update using fuzzy reasoning, suspicious process detection using distributed agent-based module, and zero-configuration for minimizing available security services. Combinations of both initialisation methods and runtime processes produce five firewall methods, as listed in Section 8.1. Security analysis and comparative study are conducted to evaluate the proposed methods. Referring to the works above, some points can be drawn to conclude this study as follows:

(i) Each combinations of initialisation and runtime process produced from this research represent the employed strategy for reducing Internet threat. It is shown in Table 9.1. Thus the performance of each combination of

initialisation and runtime process for activating firewall determines the performance of each security strategy for reducing Internet threats.

Table 9.1: The correlation between the developed methods and the applied security strategy

Methods	**Strategy to Reduce Internet Threats**
Open-condition + Fuzzy-based security rules update (*OF*)	Minimizing $U_{up} \cap O_{ut}$
Open-condition + Agent-based suspicious program detection (*OA*)	Minimizing U_{up}
Lattice-based initialization + Fuzzy-based security rules update (*LF*)	Minimizing $U_{up} \cap O_{ut}$
Lattice-based initialization + Agent-based suspicious program detection (*LA*)	Minimizing $U_{up} \cap O_{ut}$ + Minimizing U_{up}
Close-condition + Zero-configuration (*CZ*)	Minimizing O_{ut}

(ii) Research evaluation proves that lattice-based initialisation combined with agent-based module (*LA*) delivers the best performance both for providing Internet protection and facilitating the Internet access. Referring to Table 9.1, it shows that successful intranet protection is driven by the strategy for minimizing unprotected users combined with the strategy for minimizing the interaction between unprotected users and untrusted external parties. This strategy is definitely more complete compared to other combinations since the method for handling initialisation and runtime process works on different approach for intranet protection, thus minimal exposed line can be delivered by this method. Moreover, this approach is also sensitive to the appearance of threat or the raising risk of external parties. Meanwhile, the Internet access is facilitated by this method due to moderate approach of lattice-based initialisation to keep the open access to the trusted external parties at anytime during the runtime process. Referring to the evaluation results, moderate probability of available services and probability of denial of service are delivered by this method.

(iii) Second best performance for activating firewall is delivered by open condition combined with agent-based module (*OA*) and open condition combined with fuzzy reasoning (*OF*). The performance of both methods is driven by the power to facilitate Internet access. It is due to the opened-condition that is employed to handle the initialisation. Thus maximum available of service and minimum denial of service can be produced. However both methods produce a weak protection since maximum exposed line is produced. This evidence shows that the strategy to minimize unprotected users and the strategy to minimize the interaction between unprotected users and untrusted external parties are compromised by the open-condition, thus it can be concluded that these methods loosely influence to strengthen the intranet protection.

(iv) The third performance for activating firewall is delivered by the combination between lattice-based initialisation and fuzzy reasoning (*LF*). Although this effort only applies a strategy to minimize the interaction between unprotected users and untrusted external parties, however this strategy is employed by both initialisation and runtime process, therefore the intranet protection is enforced since the first time the firewall is started up to the end of its operation. However only moderate performance can be delivered by this method both for protecting intranet and facilitating Internet connection. Thus it can be concluded that the strategy to minimize the interaction between unprotected users and untrusted external parties only produce moderate performance in activating firewall.

(v) The worst performance produced from this effort is delivered by the combination between close-condition and zero-configuration (*CZ*). This method applies a strategy to minimize the untrusted external parties, Although this approach is proved capable to maintain minimal services at runtime in order to have a hard secure system (Ranum and Avolio, 1994), however it still produces more exposed line compared to the combination of lattice-based initialisation and agent-based (*LA*). This condition is due to the insensitive mechanism of zero-configuration to the presents of Internet threats. Meanwhile Internet connection is greatly compromised by this

method since it consistently maintains the close-condition along the operation of firewall by computing the timeout of and dropping each opened canal. This mechanism reduces the availability of services and increases denial of services. The speeds for opening canals become the critical parameters in running this method, particularly to deal with real-time Internet access.

The points above disclose the advantages and weaknesses of the strategies to activate firewall mechanism. Although it has been proved that these approaches are capable to solve the problems of the existing firewall technology, further work on improving the performance of active firewall is still necessary. Therefore some points can be highlighted to become the object of further study as follows:

(i) The experiment of agent-based module shows that active firewall is capable to only stop the action of malicious information flow. This condition is due to the detection process that comes later after the malicious flow has been started. Thus current achievement of active firewall still cannot prevent the appearance of malicious information flow before it is started, although 10% of the experimental results fulfilled this condition. An effort to improve the speed of the detection process and closing canals would significantly deliver a more powerful approach in protecting intranet.

(ii) The close condition and zero-based configuration has been proved capable to provide hard protection to the intranet. It is achieved by minimizing the available services at runtime. This method however introduces denial of services. Thus the effort to improve the speed for opening canals as well as to accurately compute the timeout for each external party would significantly increase the performance of this system. The work would also increase the applicability of zero-configuration to deal with real time operation of active firewall.

CHAPTER 9

CONCLUSION

An effort to combat Internet threats has been presented, with the purpose is to provide intranet protection in accessing Internet. It is achieved by activating the mechanism of firewall, the most deployed security tool for securing information systems (CERT, 2004). In this study, the security strategies to eliminate Internet threats are developed, and then are used as the foundation for activating the mechanism of network firewall. Meanwhile, the model of active firewall is formulated with the concept of canalisation to represent the connections of internal users to external parties. And the operation of active firewall is formulated into two main stages i.e. initialisation and runtime process. To handle the initialisation, three methods are developed i.e. close-condition, open-condition, and lattice-based. While for the runtime process, three methods are also developed i.e. adaptive security rules update using fuzzy reasoning, suspicious process detection using distributed agent-based module, and zero-configuration for minimizing available security services. Combinations of both initialisation methods and runtime processes produce five firewall methods, as listed in Section 8.1. Security analysis and comparative study are conducted to evaluate the proposed methods. Referring to the works above, some points can be drawn to conclude this study as follows:

(i) Each combinations of initialisation and runtime process produced from this research represent the employed strategy for reducing Internet threat. It is shown in Table 9.1. Thus the performance of each combination of

initialisation and runtime process for activating firewall determines the performance of each security strategy for reducing Internet threats.

Table 9.1: The correlation between the developed methods and the applied security strategy

Methods	**Strategy to Reduce Internet Threats**
Open-condition + Fuzzy-based security rules update (*OF*)	Minimizing $U_{up} \cap O_{ut}$
Open-condition + Agent-based suspicious program detection (*OA*)	Minimizing U_{up}
Lattice-based initialization + Fuzzy-based security rules update (*LF*)	Minimizing $U_{up} \cap O_{ut}$
Lattice-based initialization + Agent-based suspicious program detection (*LA*)	Minimizing $U_{up} \cap O_{ut}$ + Minimizing U_{up}
Close-condition + Zero-configuration (*CZ*)	Minimizing O_{ut}

(ii) Research evaluation proves that lattice-based initialisation combined with agent-based module (*LA*) delivers the best performance both for providing Internet protection and facilitating the Internet access. Referring to Table 9.1, it shows that successful intranet protection is driven by the strategy for minimizing unprotected users combined with the strategy for minimizing the interaction between unprotected users and untrusted external parties. This strategy is definitely more complete compared to other combinations since the method for handling initialisation and runtime process works on different approach for intranet protection, thus minimal exposed line can be delivered by this method. Moreover, this approach is also sensitive to the appearance of threat or the raising risk of external parties. Meanwhile, the Internet access is facilitated by this method due to moderate approach of lattice-based initialisation to keep the open access to the trusted external parties at anytime during the runtime process. Referring to the evaluation results, moderate probability of available services and probability of denial of service are delivered by this method.

(iii) Second best performance for activating firewall is delivered by open condition combined with agent-based module (*OA*) and open condition combined with fuzzy reasoning (*OF*). The performance of both methods is driven by the power to facilitate Internet access. It is due to the opened-condition that is employed to handle the initialisation. Thus maximum available of service and minimum denial of service can be produced. However both methods produce a weak protection since maximum exposed line is produced. This evidence shows that the strategy to minimize unprotected users and the strategy to minimize the interaction between unprotected users and untrusted external parties are compromised by the open-condition, thus it can be concluded that these methods loosely influence to strengthen the intranet protection.

(iv) The third performance for activating firewall is delivered by the combination between lattice-based initialisation and fuzzy reasoning (*LF*). Although this effort only applies a strategy to minimize the interaction between unprotected users and untrusted external parties, however this strategy is employed by both initialisation and runtime process, therefore the intranet protection is enforced since the first time the firewall is started up to the end of its operation. However only moderate performance can be delivered by this method both for protecting intranet and facilitating Internet connection. Thus it can be concluded that the strategy to minimize the interaction between unprotected users and untrusted external parties only produce moderate performance in activating firewall.

(v) The worst performance produced from this effort is delivered by the combination between close-condition and zero-configuration (*CZ*). This method applies a strategy to minimize the untrusted external parties, Although this approach is proved capable to maintain minimal services at runtime in order to have a hard secure system (Ranum and Avolio, 1994), however it still produces more exposed line compared to the combination of lattice-based initialisation and agent-based (*LA*). This condition is due to the insensitive mechanism of zero-configuration to the presents of Internet threats. Meanwhile Internet connection is greatly compromised by this

method since it consistently maintains the close-condition along the operation of firewall by computing the timeout of and dropping each opened canal. This mechanism reduces the availability of services and increases denial of services. The speeds for opening canals become the critical parameters in running this method, particularly to deal with real-time Internet access.

The points above disclose the advantages and weaknesses of the strategies to activate firewall mechanism. Although it has been proved that these approaches are capable to solve the problems of the existing firewall technology, further work on improving the performance of active firewall is still necessary. Therefore some points can be highlighted to become the object of further study as follows:

(i) The experiment of agent-based module shows that active firewall is capable to only stop the action of malicious information flow. This condition is due to the detection process that comes later after the malicious flow has been started. Thus current achievement of active firewall still cannot prevent the appearance of malicious information flow before it is started, although 10% of the experimental results fulfilled this condition. An effort to improve the speed of the detection process and closing canals would significantly deliver a more powerful approach in protecting intranet.

(ii) The close condition and zero-based configuration has been proved capable to provide hard protection to the intranet. It is achieved by minimizing the available services at runtime. This method however introduces denial of services. Thus the effort to improve the speed for opening canals as well as to accurately compute the timeout for each external party would significantly increase the performance of this system. The work would also increase the applicability of zero-configuration to deal with real time operation of active firewall.

REFERENCES

Alexander, D.S., Anagnostaskis, K.G., Arbaugh, W.A., Keromytis, A.D. and Smith, M.S. (1999). *The Price of Safety in an Active Network.* Journal of Communications and Networks (JCN), special issue on Programmable Switches and Routers. March 2001. 3(1): 4-18.

Anagnostaskis, K.G., Greenwald, M.B., Ionnidis, S., Keromytis, A.D., and Li, D. (2003). *A Cooperative Immunization System for an Untrusting Internet.* 2003. 11th IEEE International Conference on Networks (ICON). September/October 2003. Sydney, Australia: IEEE, 403-408.

Anthony, R.N. (2003). *Management Accounting: A Personal History*. Journal of Management Accounting Research. ABI/INFORM Global.15: 249-253.

Arbaugh, W.A. (2002). *Active Systems Management: The Evolution of Firewalls*. Invited paper to the 3rd International Workshop on Information Security Applications. August 2002. Cheju Island, Korea, 19-30.

Arbaugh, W.A. (2003). *Firewalls: An Outdated Defense*. In IEEE Computer. June 2003. 36(6): 112-113.

Associated Press. (2003). *Spammers Steal E-mail Address from Orbitz.* Weekly Magazine InformationWeek. 29 October 2003. http://www.informationweek.com

Associated Press. (2004a). *Hackers Crack Purdue's Computer System*. CNN Technology. 22 October 2004. http://www.cnn.com

Associated Press. (2004b). *CoolWebSearch A Spyware Mystery: Who's Behind it?*. CNN Technology. 2 November 2004. http://www.cnn.com

Bellovin, S.M. (1999). *Distributed Firewalls*. In ;login, The USENIX Magazine. November 1999. 37-39.

Bernades, M.C. and Moreira, E.D.S. (2000). *Implementation of an Intrusion Detection System Based on Mobile Agents*. International Symposium on Software engineering for Parallel and Distributed Systems. June 2000. Limerick, Ireland: IEEE, 158-164.

Castano, S., Fugini, M., Martella, G., and Samarati, P. (1995). *Database Security*. Great Britain: ACM (Association for Computing Machinery) Inc Press.

CERT. (2002). *CERT Coordination Center 2002 Annual Report*. CERT Coordination Center Annual Report. http://www.cert.org

CERT. (2003). CERT *Coordination Center 2003 Annual Report*. CERT Coordination Center Annual Report. http://www.cert.org

CERT. (2004). *2004 E-Crime Watch Survey™: Summary of Findings*. CERT Coordination Center. http://www.cert.org

Cherry, S.M. (2002). *All the Ills that Flesh is Heir To: Internet Viruses Can Get Worse-Much Worse*. IEEE Spectrum. November 2002. IEEE, 52.

Christodorescu, M., and Jha, S. (2003). *Static Analysis of Executable to Detect Malicious Patterns*. In 12th USENIX Security Symposium. August 2003. Washington DC, USA: USENIX, 169-186.

Davies, R.M. (2002). *Firewalls, Intrusion Detection Systems and Vulnerability Assessment: A Superior Conjunction?*. Network Security. 2002(9): 8-11.

Denning, D.E. (1976). *A Lattice Model of Secure Information Flow*. Communications of ACM, 1976. 19(5): 236-243.

Eschelbeck, G. (2000). *Active Security: A Proactive Approach for Computer Security Systems*. Journal of Network and Computer Applications. April 2000. 23(2): 109-130.

Freed, N. (2000). *RFC 2979 – Behavior and Requirements for Internet Firewalls*. Internet RFC/STD/FYI/BCP Archives. October 2000. Network Working Group, Request for Comments: 2979.

Garkinfel, S., and Spafford, G. (1997). *Web Security and Commerce.* USA: O'Reilly and Associates.

Goncalves, M. (2000). *Firewalls: A Complete Guide*. USA: McGraw Hill Osborne Media.

Green, S., Hurst, L., Nangle, B., Cunningham, P., Somers, F., and Evans, R. (1997). *Software Agents: A Review*. A Trinity College Dublin: Technical Report.

Guan, J., Liu, D.X. and Wang, T. (2004). *Applications of Fuzzy Data Mining Methods for Intrusion Detection Systems.* ICCSA 2004, Lecturer Notes on Computer Science 3035. Springer 2004: 706-714.

Haixin, D., Jianping, W., and Xing, L. (2000). *Policy-Based Access Control Framework for Large Networks*. Proceedings of IEEE International Conference on Network (ICON 2000). 5-8 September 2000. Singapore: IEEE, 267-272.

He, M., and Leung, H. (2002). *Agents in E-Commerce: State of the Art.* Knowledge and Information Systems. July 2002. 4(3): 257-282.

Hernandez, J.C., Sierra, J.M. and Ramos, B. (2001). *Search Engines as a Security Threat*. IEEE Computer. October 2001. 34(10): 25-30.

Hickman, B., Newman, D., Tadjudin, S., Martin, T. (2003). RFC 3511 - Benchmarking Methodology for Firewall Performance. Internet RFC/STD/FYI/BCP Archives. April 2003. Network Working Group, Request for Comments: 3511.

Hicks, M., Keromytis, A.D. and Smith, J.M. (2003). *A Secure PLAN*. IEEE Transactions on System, Man, and Cybernetics-Part C: Applications and Review, August 2003. 33(3): 413-426.

Huang, Y.W., Yu, F., Hang, C., Tsai, C.H., Lee, D.T., and Kuo, S.Y. (2004). *Securing Web Application Code by Static Analysis and Runtime Protection.* World Wide Web 2004. May 17-22, 2004. New York, USA: ACM, 40-51.

Hunt, R., and Verwoerd, T. (2003). *Reactive Firewalls: A New Technique*. Elsevier Journal on Computer Communications. July 2003. 26(12): 1302-1317.

Hwang, K. and Gangadharan, M. (2001). *Micro-Firewalls for Dynamic Network Security with Distributed Intrusion Detection*. IEEE International Symposium on Network Computing and Applications. October 2001.Cambridge, MA, USA: IEEE, 68-79.

Ioannidis, S., Keromytis, D., Bellovin, S.M., and Smith, J.M. (2000). *Implementing a Distributed Firewall.* 7th ACM Conference on Computer and Communication Security. 1-4 November 2000.Athens, Greece.

Kamara, S., Fahmy, S., Schultz, E., Kreschbaum, F., and Frantzen, M. (2003). *Analysis of Vulnerabilities in Internet Firewalls*. Elsevier Computer and Security 2003. 22(3): 214-232.

Kayssi, A., Harik, L., Ferzli, R. and Fawaz, M. (2000). *FPGA-Based Internet Protocol Firewall Chip*. The 7th IEEE International Conference on Electronics, Circuits and Systems (ICECS 2000). December 2000. Kaslik, Lebanon: IEEE, 316 -319.

Kienzle, D.M., and Elder, M.C. (2003). *Recent Worms: A Survey and Trends*. Proceedings of the 2003 ACM Workshop on Rapid Malcode. 27 October 2003. Washington DC, USA: ACM, 1-10.

Kim. J.S., Kim, M.S. and Noh, B.N. (2004). *A Fuzzy Expert System for Network Forensics*. ICCSA 2004, Lecturer Notes in Computer Science 3034. Springer 2004: 175-182.

Klir, G.J. and Yuan, B. (1995). *Fuzzy Sets and Fuzzy Logic: Theory and Application*. USA: Prentice Hall Inc.

Labiod, K.B.H., Boutaba, R. and Guessoum, Z. (2000). *Network Security Management with Intelligent Agents*. IEEE/IFIP Network Operations and Managements Symposium. September 2000. Hawaii, USA: IEEE, 579-592.

Labuschagne, L. and Eloff, J.H.P. (1998). *The Use of Real-Time Risk Analysis to Enable Dynamic Activation of Countermeasures*. Elsevier Journal of Computer and Security. 17(4): 347-357.

Lai, S.C., Kuo, W.C., and Hsieh, M.C. (2004). *Defending Against Internet Worm-Like Infestations*. Proceedings of the 18th International Conference on Advanced Information Networking and Application (AINA '04). March 2004. Fukuoka, Japan: IEEE, 152-157.

Li, J., Zhang, G.Y. and Gu, G.C. (2004). *A Multi-Agent-Based Architecture for Network Attack Resistant System*. GCC 2003, Part I, Lecturer Notes in Computer Science 3032. Springer 2004: 980-983.

Lee, T.K., Yusuf, S., Luk, W., Sloman, M., Lupu, E. and Dulay, N. (2002). *Development Framework for Firewall Processors*. Proceedings of the 2002 IEEE International Conference on Field Programmable Technology. December 2002. Hongkong, China: IEEE, 352-355.

Lehtonen, S., Ahola, K., Koskinen, T., Lyijynen, M. and Pesole, J. (2003). *Roaming Active Filtering Firewall*. Proceedings of Smart Objects Conference (SOC'2003). 15-17 May 2003. Grenoble, France.

Lockwood, J.W., Neely, C., Zuver, C., Moscola, J., Dharmapurikar, S. and Lim, D. (2003). *An Extensible, System-On-Programmable-Chip, Content-Aware Internet Firewall*. FPL 2003, Lecturer Notes on Computer Science 2778. Springer 2003: 859-868.

Negnevitsky, M. (2002). *Artificial Intelligence: A Guide to Intelligent Systems*. England: Pearson Education Limited.

Newman, D. (1999). *RFC 2647 – Benchmarking Terminology for Firewall Performance*. Internet RFC/STD/FYI/BCP Archives. August 1999. Network Working Group, Request for Comments: 2647.

Ogletree, T.W. (2000). *Practical Firewalls.* USA: Que Corporation. June 2000.

Payne, C. and Markham, T. (2001). *Architecture and Applications for a Distributed Embedded Firewall*. In 17th Annual Computer Security Applications Conference. December 2001. New Orleans, LA, USA: IEEE, 329-338.

Power, R. (2002). *CSI/FBI Computer Crime and Security Survey*. Computer Security Issues and Trends. 2002. 8(1): 1-24.

Pruitt, S. (2004). *Web Attack Aims to Steal Surfers' Financial Details*. ComputerWorld. 25 June 2004. http://computerworld.com/securitytopics/security

Ranum, M.J. and Avolio, F.M. (1994). *A Toolkit and Methods for Internet Firewalls*. In the Proceedings of the Summer USENIX Conference. June 1994. Boston, Massachusetts, USA: USENIX, 37-44.

Ren, Y, Buskens, R. and Gonzales, O. (2004). *Dependable Initialization on Large-Scale Distributed Software*. Proceedings of the 2004 International Conference on Dependable Systems and Networks (DSN '04), IEEE Computer Society.

Reumann, J., Jamjoom, H. and Shin, K. (2001). *Adaptive Packet Filters*. In the Proceedings of IEEE Global Telecommunications Conference. November 2001. San Antonio, Texas, USA: IEEE, 25-29.

Robbins, D. (2001). *Common threads - Dynamic iptables firewalls.* From IBM website, updated 1 April 2001. http://www-136.ibm.com/developerworks/linux.

Ru, W.G. and Eloff, H.P. (1996). *Risk Analysis Modelling with the Use of Fuzzy Logic*. Elsevier Journal of Computer & Security. 15(3): 239-248.

Sandhu, R.S. (1993). *Lattice-Based Access Control Models*. IEEE Transactions on Computer, November 1993. 26(11): 9-19.

Seo, J., Kim, H.S., Cho, S. and Cha, S. (2004). *Web Server Attack Categorization Based on Root Causes and Their Locations*. Proceedings of the International Conference on Information Technology Coding and Computing (ITCC '04). April 2004. Las Vegas, Nevada, USA: IEEE, 90-96.

Shakshuki, E., Luo, Z. and Gong, J. (2004). *An Agent-Based Approach to Security Service*. Elsevier Journal of Network and Computer Applications, Article In Press.

Silva, S.D., Yemini, Y. and Florissi, D. (2001). *The NetScript Active Network System*. IEEE Journal on Selected Areas in Communications, 2001. 19(3): 538-551.

Sully, B. (2005). *IPTables Firewall Script and Configuration Files for Linux 2.4.x-2.6.x*. In Malibyte website, updated 8 April 2005. http://www.malibyte.net/iptables/scripts/fwscripts.html.

Toth, T., and Kruegel, C. (2002). *Evaluating the Impact of Automated Intrusion Response Mechanism*. Proceedings of the 18th Annual Computer Security Applications Conference (ACSAC '02). December 2002. Las Vegas, Nevada, USA: IEEE, 301-310.

Triola, M.F. (2001). *Elementary Statistics*. Eighth Edition. USA: Addison Wesley Longman.

Venkatesan, R.M. and Bhattacharya, S. (1997). *Threat-Adaptive Security Policy*. 1997. In the International Conference for High Performance, Computing and Communications. February 1997. San Jose, CA, USA: IEEE, 525-531.

Verwoerd, T., and Hunt, R. (2002). *Policy and Implementation of an Adaptive Firewall*. Proceedings of the 10th IEEE International Conference on Networks. August 2002. Singapore: IEEE, 434-439.

White, J.E. (1996). *Telescript Technology: Mobile Agents.* In BradShaw, J. (ed): Software Agents. AAI Press/MIT Press.

Whitman, M.E. (2003). *Enemy at the Gate: Threats to Information Security*. Communications of the ACM. August 2003. 46(8): 91-95.

Wired News. (2004). *New Virus May Steal Data*. Wired News. 25 June 2004. http://www.wired.com/news/infostructure

Xian, Z., Jin, H., Liu, K., and Han, Z. (2002). *A Mobile-Agent Based Distributed Dynamic μFirewall Architecture*. Proceedings of the 9th International Conference on Parallel and Distributed Systems (ICPADS '02). December 2002. Taiwan: IEEE, 431-436.

Yen , J., Langari, R., and Zadeh, L.A. (1995). *Industrial Applications of Fuzzy Logic and Intelligent Systems.* New York, USA: IEEE Press.

Zadeh, L. (1965). *Fuzzy Sets.* Information and Control. 8(3): 338-353.

Zaki, M. and Sobh, T.S. (2004). *A Cooperative Agent-Based Model for Active Security Systems*. Elsevier Journal of Network and Computer Applications. 27 (2004): 201-220.

Zeller, C. (2004). *Craig Zeller's Firewall Scripts.* From ZDI website, updated 17 June 2004. http://www.zdi.net/Linux/firewalls.html.

Zetter, K. (2004). *Information Security News: Kazza Delivers More Than Tunes*. InfoSec News. 12 January 2004. http://www.weird.com/news/business

Zou, J., Lu, K. and Jin, Z. (2002). *Architecture and Fuzzy Adaptive Security Algorithm in Intelligent Firewall*. In the Proceedings of Military Communications Conference. October 2002. California, USA: IEEE, 1145-1149.

REFERENCES

Alexander, D.S., Anagnostaskis, K.G., Arbaugh, W.A., Keromytis, A.D. and Smith, M.S. (1999). *The Price of Safety in an Active Network.* Journal of Communications and Networks (JCN), special issue on Programmable Switches and Routers. March 2001. 3(1): 4-18.

Anagnostaskis, K.G., Greenwald, M.B., Ionnidis, S., Keromytis, A.D., and Li, D. (2003). *A Cooperative Immunization System for an Untrusting Internet.* 2003. 11th IEEE International Conference on Networks (ICON). September/October 2003. Sydney, Australia: IEEE, 403-408.

Anthony, R.N. (2003). *Management Accounting: A Personal History*. Journal of Management Accounting Research. ABI/INFORM Global.15: 249-253.

Arbaugh, W.A. (2002). *Active Systems Management: The Evolution of Firewalls*. Invited paper to the 3rd International Workshop on Information Security Applications. August 2002. Cheju Island, Korea, 19-30.

Arbaugh, W.A. (2003). *Firewalls: An Outdated Defense*. In IEEE Computer. June 2003. 36(6): 112-113.

Associated Press. (2003). *Spammers Steal E-mail Address from Orbitz.* Weekly Magazine InformationWeek. 29 October 2003. http://www.informationweek.com

Associated Press. (2004a). *Hackers Crack Purdue's Computer System*. CNN Technology. 22 October 2004. http://www.cnn.com

Associated Press. (2004b). *CoolWebSearch A Spyware Mystery: Who's Behind it?*. CNN Technology. 2 November 2004. http://www.cnn.com

Bellovin, S.M. (1999). *Distributed Firewalls*. In ;login, The USENIX Magazine. November 1999. 37-39.

Bernades, M.C. and Moreira, E.D.S. (2000). *Implementation of an Intrusion Detection System Based on Mobile Agents*. International Symposium on Software engineering for Parallel and Distributed Systems. June 2000. Limerick, Ireland: IEEE, 158-164.

Castano, S., Fugini, M., Martella, G., and Samarati, P. (1995). *Database Security*. Great Britain: ACM (Association for Computing Machinery) Inc Press.

CERT. (2002). *CERT Coordination Center 2002 Annual Report*. CERT Coordination Center Annual Report. http://www.cert.org

CERT. (2003). CERT *Coordination Center 2003 Annual Report*. CERT Coordination Center Annual Report. http://www.cert.org

CERT. (2004). *2004 E-Crime Watch Survey™: Summary of Findings*. CERT Coordination Center. http://www.cert.org

Cherry, S.M. (2002). *All the Ills that Flesh is Heir To: Internet Viruses Can Get Worse-Much Worse*. IEEE Spectrum. November 2002. IEEE, 52.

Christodorescu, M., and Jha, S. (2003). *Static Analysis of Executable to Detect Malicious Patterns*. In 12th USENIX Security Symposium. August 2003. Washington DC, USA: USENIX, 169-186.

Davies, R.M. (2002). *Firewalls, Intrusion Detection Systems and Vulnerability Assessment: A Superior Conjunction?*. Network Security. 2002(9): 8-11.

Denning, D.E. (1976). *A Lattice Model of Secure Information Flow*. Communications of ACM, 1976. 19(5): 236-243.

Eschelbeck, G. (2000). *Active Security: A Proactive Approach for Computer Security Systems*. Journal of Network and Computer Applications. April 2000. 23(2): 109-130.

Freed, N. (2000). *RFC 2979 – Behavior and Requirements for Internet Firewalls*. Internet RFC/STD/FYI/BCP Archives. October 2000. Network Working Group, Request for Comments: 2979.

Garkinfel, S., and Spafford, G. (1997). *Web Security and Commerce.* USA: O'Reilly and Associates.

Goncalves, M. (2000). *Firewalls: A Complete Guide*. USA: McGraw Hill Osborne Media.

Green, S., Hurst, L., Nangle, B., Cunningham, P., Somers, F., and Evans, R. (1997). *Software Agents: A Review*. A Trinity College Dublin: Technical Report.

Guan, J., Liu, D.X. and Wang, T. (2004). *Applications of Fuzzy Data Mining Methods for Intrusion Detection Systems.* ICCSA 2004, Lecturer Notes on Computer Science 3035. Springer 2004: 706-714.

Haixin, D., Jianping, W., and Xing, L. (2000). *Policy-Based Access Control Framework for Large Networks*. Proceedings of IEEE International Conference on Network (ICON 2000). 5-8 September 2000. Singapore: IEEE, 267-272.

He, M., and Leung, H. (2002). *Agents in E-Commerce: State of the Art.* Knowledge and Information Systems. July 2002. 4(3): 257-282.

Hernandez, J.C., Sierra, J.M. and Ramos, B. (2001). *Search Engines as a Security Threat*. IEEE Computer. October 2001. 34(10): 25-30.

Hickman, B., Newman, D., Tadjudin, S., Martin, T. (2003). RFC 3511 - Benchmarking Methodology for Firewall Performance. Internet RFC/STD/FYI/BCP Archives. April 2003. Network Working Group, Request for Comments: 3511.

Hicks, M., Keromytis, A.D. and Smith, J.M. (2003). *A Secure PLAN*. IEEE Transactions on System, Man, and Cybernetics-Part C: Applications and Review, August 2003. 33(3): 413-426.

Huang, Y.W., Yu, F., Hang, C., Tsai, C.H., Lee, D.T., and Kuo, S.Y. (2004). *Securing Web Application Code by Static Analysis and Runtime Protection.* World Wide Web 2004. May 17-22, 2004. New York, USA: ACM, 40-51.

Hunt, R., and Verwoerd, T. (2003). *Reactive Firewalls: A New Technique*. Elsevier Journal on Computer Communications. July 2003. 26(12): 1302-1317.

Hwang, K. and Gangadharan, M. (2001). *Micro-Firewalls for Dynamic Network Security with Distributed Intrusion Detection*. IEEE International Symposium on Network Computing and Applications. October 2001.Cambridge, MA, USA: IEEE, 68-79.

Ioannidis, S., Keromytis, D., Bellovin, S.M., and Smith, J.M. (2000). *Implementing a Distributed Firewall.* 7th ACM Conference on Computer and Communication Security. 1-4 November 2000.Athens, Greece.

Kamara, S., Fahmy, S., Schultz, E., Kreschbaum, F., and Frantzen, M. (2003). *Analysis of Vulnerabilities in Internet Firewalls*. Elsevier Computer and Security 2003. 22(3): 214-232.

Kayssi, A., Harik, L., Ferzli, R. and Fawaz, M. (2000). *FPGA-Based Internet Protocol Firewall Chip*. The 7th IEEE International Conference on Electronics, Circuits and Systems (ICECS 2000). December 2000. Kaslik, Lebanon: IEEE, 316 -319.

Kienzle, D.M., and Elder, M.C. (2003). *Recent Worms: A Survey and Trends*. Proceedings of the 2003 ACM Workshop on Rapid Malcode. 27 October 2003. Washington DC, USA: ACM, 1-10.

Kim. J.S., Kim, M.S. and Noh, B.N. (2004). *A Fuzzy Expert System for Network Forensics*. ICCSA 2004, Lecturer Notes in Computer Science 3034. Springer 2004: 175-182.

Klir, G.J. and Yuan, B. (1995). *Fuzzy Sets and Fuzzy Logic: Theory and Application*. USA: Prentice Hall Inc.

Labiod, K.B.H., Boutaba, R. and Guessoum, Z. (2000). *Network Security Management with Intelligent Agents*. IEEE/IFIP Network Operations and Managements Symposium. September 2000. Hawaii, USA: IEEE, 579-592.

Labuschagne, L. and Eloff, J.H.P. (1998). *The Use of Real-Time Risk Analysis to Enable Dynamic Activation of Countermeasures*. Elsevier Journal of Computer and Security. 17(4): 347-357.

Lai, S.C., Kuo, W.C., and Hsieh, M.C. (2004). *Defending Against Internet Worm-Like Infestations*. Proceedings of the 18th International Conference on Advanced Information Networking and Application (AINA '04). March 2004. Fukuoka, Japan: IEEE, 152-157.

Li, J., Zhang, G.Y. and Gu, G.C. (2004). *A Multi-Agent-Based Architecture for Network Attack Resistant System*. GCC 2003, Part I, Lecturer Notes in Computer Science 3032. Springer 2004: 980-983.

Lee, T.K., Yusuf, S., Luk, W., Sloman, M., Lupu, E. and Dulay, N. (2002). *Development Framework for Firewall Processors*. Proceedings of the 2002 IEEE International Conference on Field Programmable Technology. December 2002. Hongkong, China: IEEE, 352-355.

Lehtonen, S., Ahola, K., Koskinen, T., Lyijynen, M. and Pesole, J. (2003). *Roaming Active Filtering Firewall*. Proceedings of Smart Objects Conference (SOC'2003). 15-17 May 2003. Grenoble, France.

Lockwood, J.W., Neely, C., Zuver, C., Moscola, J., Dharmapurikar, S. and Lim, D. (2003). *An Extensible, System-On-Programmable-Chip, Content-Aware Internet Firewall*. FPL 2003, Lecturer Notes on Computer Science 2778. Springer 2003: 859-868.

Negnevitsky, M. (2002). *Artificial Intelligence: A Guide to Intelligent Systems*. England: Pearson Education Limited.

Newman, D. (1999). *RFC 2647 – Benchmarking Terminology for Firewall Performance*. Internet RFC/STD/FYI/BCP Archives. August 1999. Network Working Group, Request for Comments: 2647.

Ogletree, T.W. (2000). *Practical Firewalls.* USA: Que Corporation. June 2000.

Payne, C. and Markham, T. (2001). *Architecture and Applications for a Distributed Embedded Firewall*. In 17th Annual Computer Security Applications Conference. December 2001. New Orleans, LA, USA: IEEE, 329-338.

Power, R. (2002). *CSI/FBI Computer Crime and Security Survey*. Computer Security Issues and Trends. 2002. 8(1): 1-24.

Pruitt, S. (2004). *Web Attack Aims to Steal Surfers' Financial Details*. ComputerWorld. 25 June 2004. http://computerworld.com/securitytopics/security

Ranum, M.J. and Avolio, F.M. (1994). *A Toolkit and Methods for Internet Firewalls*. In the Proceedings of the Summer USENIX Conference. June 1994. Boston, Massachusetts, USA: USENIX, 37-44.

Ren, Y, Buskens, R. and Gonzales, O. (2004). *Dependable Initialization on Large-Scale Distributed Software*. Proceedings of the 2004 International Conference on Dependable Systems and Networks (DSN '04), IEEE Computer Society.

Reumann, J., Jamjoom, H. and Shin, K. (2001). *Adaptive Packet Filters*. In the Proceedings of IEEE Global Telecommunications Conference. November 2001. San Antonio, Texas, USA: IEEE, 25-29.

Robbins, D. (2001). *Common threads - Dynamic iptables firewalls.* From IBM website, updated 1 April 2001. http://www-136.ibm.com/developerworks/linux.

Ru, W.G. and Eloff, H.P. (1996). *Risk Analysis Modelling with the Use of Fuzzy Logic*. Elsevier Journal of Computer & Security. 15(3): 239-248.

Sandhu, R.S. (1993). *Lattice-Based Access Control Models*. IEEE Transactions on Computer, November 1993. 26(11): 9-19.

Seo, J., Kim, H.S., Cho, S. and Cha, S. (2004). *Web Server Attack Categorization Based on Root Causes and Their Locations*. Proceedings of the International Conference on Information Technology Coding and Computing (ITCC '04). April 2004. Las Vegas, Nevada, USA: IEEE, 90-96.

Shakshuki, E., Luo, Z. and Gong, J. (2004). *An Agent-Based Approach to Security Service*. Elsevier Journal of Network and Computer Applications, Article In Press.

Silva, S.D., Yemini, Y. and Florissi, D. (2001). *The NetScript Active Network System*. IEEE Journal on Selected Areas in Communications, 2001. 19(3): 538-551.

Sully, B. (2005). *IPTables Firewall Script and Configuration Files for Linux 2.4.x-2.6.x*. In Malibyte website, updated 8 April 2005. http://www.malibyte.net/iptables/scripts/fwscripts.html.

Toth, T., and Kruegel, C. (2002). *Evaluating the Impact of Automated Intrusion Response Mechanism*. Proceedings of the 18th Annual Computer Security Applications Conference (ACSAC '02). December 2002. Las Vegas, Nevada, USA: IEEE, 301-310.

Triola, M.F. (2001). *Elementary Statistics*. Eighth Edition. USA: Addison Wesley Longman.

Venkatesan, R.M. and Bhattacharya, S. (1997). *Threat-Adaptive Security Policy*. 1997. In the International Conference for High Performance, Computing and Communications. February 1997. San Jose, CA, USA: IEEE, 525-531.

Verwoerd, T., and Hunt, R. (2002). *Policy and Implementation of an Adaptive Firewall*. Proceedings of the 10th IEEE International Conference on Networks. August 2002. Singapore: IEEE, 434-439.

White, J.E. (1996). *Telescript Technology: Mobile Agents.* In BradShaw, J. (ed): Software Agents. AAI Press/MIT Press.

Whitman, M.E. (2003). *Enemy at the Gate: Threats to Information Security*. Communications of the ACM. August 2003. 46(8): 91-95.

Wired News. (2004). *New Virus May Steal Data*. Wired News. 25 June 2004. http://www.wired.com/news/infostructure

Xian, Z., Jin, H., Liu, K., and Han, Z. (2002). *A Mobile-Agent Based Distributed Dynamic μFirewall Architecture*. Proceedings of the 9th International Conference on Parallel and Distributed Systems (ICPADS '02). December 2002. Taiwan: IEEE, 431-436.

Yen , J., Langari, R., and Zadeh, L.A. (1995). *Industrial Applications of Fuzzy Logic and Intelligent Systems.* New York, USA: IEEE Press.

Zadeh, L. (1965). *Fuzzy Sets.* Information and Control. 8(3): 338-353.

Zaki, M. and Sobh, T.S. (2004). *A Cooperative Agent-Based Model for Active Security Systems*. Elsevier Journal of Network and Computer Applications. 27 (2004): 201-220.

Zeller, C. (2004). *Craig Zeller's Firewall Scripts.* From ZDI website, updated 17 June 2004. http://www.zdi.net/Linux/firewalls.html.

Zetter, K. (2004). *Information Security News: Kazza Delivers More Than Tunes*. InfoSec News. 12 January 2004. http://www.weird.com/news/business

Zou, J., Lu, K. and Jin, Z. (2002). *Architecture and Fuzzy Adaptive Security Algorithm in Intelligent Firewall*. In the Proceedings of Military Communications Conference. October 2002. California, USA: IEEE, 1145-1149.

APPENDIX A1

The Script of Lattice-Based Initialization Process

```
#!/bin/sh

echo ============================================================
echo LATTICE-BASED FIREWALL : which is not stated is prohibited
echo ============================================================

start=`cat /PROJECT/TEMP/TIMESTAMP`

# Implementation of Lattice is conducted on initialization
# eth0 = EXTERNAL NETWORK
# eth1 = INTERNAL NETWORK

# ------------------
# 1. Initialization
# ------------------

# INTERNAL_SUBNET="24.4.74"

> /PROJECT/TEMP/PROXY_BUFF_INT

# > /PROJECT/TEMP/PROXY_BUFF_EXT

> /PROJECT/TEMP/IP_SENDER
> /PROJECT/TEMP/IP_ADDR_SENDER
> /PROJECT/TEMP/IP_ADDR_RECEIVER
> /PROJECT/TEMP/TABLEIP
> /PROJECT/TEMP/HISTORY

> /PROJECT/TEMP/firewall_rule

# > /PROJECT/TEMP/DELTATIME
# > /PROJECT/TEMP/TIMEOUTPUT

> /PROJECT/TEMP/LOG

cd /PROJECT/FIREWALL
# To preconfigure firewall machine

echo "0" > /proc/sys/net/ipv4/ip_forward

#### Setting up firewall_rule configuration file
## -------------------------------------------

echo "## Preconfiguration"    >> /PROJECT/TEMP/firewall_rule
echo "## ================"    >> /PROJECT/TEMP/firewall_rule

# Default policy (Recording process)
# ----------------------------------
echo "iptables -P INPUT DROP"    >> /PROJECT/TEMP/firewall_rule
echo "iptables -P OUTPUT ACCEPT" >> /PROJECT/TEMP/firewall_rule
echo "iptables -P FORWARD DROP"  >> /PROJECT/TEMP/firewall_rule

# Flushing the existing rule (Recording process)
# ----------------------------------------------
echo "iptables --flush" >> /PROJECT/TEMP/firewall_rule
echo "iptables --table nat --flush" >> /PROJECT/TEMP/firewall_rule
echo "iptables --delete-chain" >> /PROJECT/TEMP/firewall_rule
echo "iptables --table nat --delete-chain" >>
/PROJECT/TEMP/firewall_rule
```

```
# Setting up Protection Level
# ---------------------------
protection_level1="24.4.74.0/24"    # full-protected
protection_level2="24.4.75.0/24"
protection_level3="24.4.76.0/24"    # unprotected

# Setting up Safety Level
# -----------------------
safety_level1="table1"              # un-trusted
safety_level2="table2"
safety_level3="table3"              # full- trusted

##====================================================================##
##====================================================================##

###### Process Protection Level 1
###### -------------------------
echo .
echo PROTECTION LEVEL 1

echo "## ==================" >> /PROJECT/TEMP/firewall_rule
echo "## Protection Level 1" >> /PROJECT/TEMP/firewall_rule
echo "## ==================" >> /PROJECT/TEMP/firewall_rule

insider_ip=$protection_level1       # Protection level 1

### Process safety level 1
### ----------------------

echo .
echo Safety Level 1

echo "# safety level 1" >> /PROJECT/TEMP/firewall_rule
echo "# --------------" >> /PROJECT/TEMP/firewall_rule

# to get jml baris of table
# -------------------------
baris_table=`wc -l /PROJECT/TABLE/$safety_level1 | awk '{print $1}'`
echo size table linewise "$[baris_table]"

## Process every data in table line-wise
## -------------------------------------
j=1
while [[ $j -le $baris_table ]]
do
      echo .
      echo .
      echo $j
      protocol=`head -n $j /PROJECT/TABLE/$safety_level1 | tail -n 1
      | awk -F, '{print $1}'`
      outsider_name=`head -n $j /PROJECT/TABLE/$safety_level1 | tail
      -n 1 | awk -F, '{print $2}'`

      # Finding IP address of outsider and insider machine
      # --------------------------------------------------
      # insider_ip=$protection_level1           # Protection level 1
      outsider_ip_addr=`grep $outsider_name /PROJECT/TABLE/host_list
      | awk '{print $2}'`

      outsider_number_of_ip=`grep -c $outsider_name
      /PROJECT/TABLE/host_list`
```

```
      # If host list contain more than one IP addresses
      # ------------------------------------------------
      if [ $outsider_number_of_ip  -gt 1 ]
      then
            echo $outsider_ip_addr > /PROJECT/TEMP/OUTSIDER_IP_ADDR
            i=1
            while [[ $i -le $outsider_number_of_ip ]]
            do
                  outsider_ip_addr=`cut -d' ' -f$i
                  /PROJECT/TEMP/OUTSIDER_IP_ADDR`
                  # echo outsider_ip_addr = $outsider_ip_addr
                  i=`expr ${i} + 1`
                  echo $protocol -- $outsider_name --
                  $outsider_ip_addr -- $insider_ip

                  # Execute iptables (Recording process)
                  # ------------------------------------
                  echo "iptables -A FORWARD -i eth1 -p $protocol -s
                  $insider_ip -d $outsider_ip_addr -j ACCEPT" >>
                  /PROJECT/TEMP/firewall_rule
                  echo "iptables -A FORWARD -i eth0 -p $protocol -s
                  $outsider_ip_addr -d $insider_ip -j ACCEPT" >>
                  /PROJECT/TEMP/firewall_rule
            done
      fi

      # If host list contain only single IP address of Internet side
      # -------------------------------------------------------------
      if [ $outsider_number_of_ip  -eq 1 ]
      then
            echo $protocol -- $outsider_name -- $outsider_ip_addr --
            $insider_ip

            # Execute iptables (Recording process)
            # ------------------------------------
            echo "iptables -A FORWARD -i eth1 -p $protocol -s
            $insider_ip -d $outsider_ip_addr -j ACCEPT" >>
            /PROJECT/TEMP/firewall_rule
            echo "iptables -A FORWARD -i eth0 -p $protocol -s
            $outsider_ip_addr -d $insider_ip -j ACCEPT" >>
            /PROJECT/TEMP/firewall_rule
      fi
      j=`expr ${j} + 1`
done

### Process safety level 2
### ----------------------

echo .
echo Safety Level 2

echo "# safety level 2"  >> /PROJECT/TEMP/firewall_rule
echo "# --------------"  >> /PROJECT/TEMP/firewall_rule

# to get jml baris of table
# -------------------------
baris_table=`wc -l /PROJECT/TABLE/$safety_level2 | awk '{print $1}'`
echo size table linewise "$[baris_table]"

## Process every data in table line-wise
## -------------------------------------
```

```
j=1
while [[ $j -le $baris_table ]]
do
      echo .
      echo .
      echo $j
      protocol=`head -n $j /PROJECT/TABLE/$safety_level2 | tail -n 1
      | awk -F, '{print $1}'`
      outsider_name=`head -n $j /PROJECT/TABLE/$safety_level2 | tail
      -n 1 | awk -F, '{print $2}'`

      # Finding IP address of outsider and insider machine
      # --------------------------------------------------
      # insider_ip=$protection_level1           # Protection level 1
      outsider_ip_addr=`grep $outsider_name /PROJECT/TABLE/host_list
      | awk '{print $2}'`

      outsider_number_of_ip=`grep -c $outsider_name
      /PROJECT/TABLE/host_list`

      # If host list contain more than one IP addresses
      # -----------------------------------------------
      if [ $outsider_number_of_ip  -gt 1 ]
      then
            echo $outsider_ip_addr > /PROJECT/TEMP/OUTSIDER_IP_ADDR
            i=1
            while [[ $i -le $outsider_number_of_ip ]]
            do
                  outsider_ip_addr=`cut -d' ' -f$i
                  /PROJECT/TEMP/OUTSIDER_IP_ADDR`
                  # echo outsider_ip_addr = $outsider_ip_addr

                  i=`expr ${i} + 1`

                  echo $protocol -- $outsider_name --
                  $outsider_ip_addr -- $insider_ip

                  # Execute iptables (Recording process)
                  # ------------------------------------
                  echo "iptables -A FORWARD -i eth1 -p $protocol -s
                  $insider_ip -d $outsider_ip_addr -j ACCEPT" >>
                  /PROJECT/TEMP/firewall_rule
                  echo "iptables -A FORWARD -i eth0 -p $protocol -s
                  $outsider_ip_addr -d $insider_ip -j ACCEPT" >>
                  /PROJECT/TEMP/firewall_rule
            done
      fi

      # If host list contain only single IP address of Internet side
      # -------------------------------------------------------------
      if [ $outsider_number_of_ip  -eq 1 ]
      then
            echo $protocol -- $outsider_name -- $outsider_ip_addr --
            $insider_ip

            # Execute iptables (Recording process)
            # ------------------------------------
            echo "iptables -A FORWARD -i eth1 -p $protocol -s
            $insider_ip -d $outsider_ip_addr -j ACCEPT" >>
            /PROJECT/TEMP/firewall_rule
```

```
            echo "iptables -A FORWARD -i eth0 -p $protocol -s
            $outsider_ip_addr -d $insider_ip -j ACCEPT" >>
            /PROJECT/TEMP/firewall_rule
      fi
      j=`expr ${j} + 1`
done

### Process safety level 3
### ----------------------

echo .
echo Safety Level 3

echo "# safety level 3"       >> /PROJECT/TEMP/firewall_rule
echo "# --------------"       >> /PROJECT/TEMP/firewall_rule

# to get jml baris of table
# -------------------------
baris_table=`wc -l /PROJECT/TABLE/$safety_level3 | awk '{print $1}'`
echo size table linewise "$[baris_table]"

## Process every data in table line-wise
## -------------------------------------
j=1
while [[ $j -le $baris_table ]]
do
      echo .
      echo .
      echo $j
      protocol=`head -n $j /PROJECT/TABLE/$safety_level3 | tail -n 1
      | awk -F, '{print $1}'`
      outsider_name=`head -n $j /PROJECT/TABLE/$safety_level3 | tail
      -n 1 | awk -F, '{print $2}'`

      # Finding IP address of outsider and insider machine
      # --------------------------------------------------
      # insider_ip=$protection_level1           # Protection level 1
      outsider_ip_addr=`grep $outsider_name /PROJECT/TABLE/host_list
      | awk '{print $2}'`

      outsider_number_of_ip=`grep -c $outsider_name
      /PROJECT/TABLE/host_list`

      # If host list contain more than one IP addresses
      # -----------------------------------------------
      if [ $outsider_number_of_ip  -gt 1 ]
      then
            echo $outsider_ip_addr > /PROJECT/TEMP/OUTSIDER_IP_ADDR
            i=1
            while [[ $i -le $outsider_number_of_ip ]]
            do
                  outsider_ip_addr=`cut -d' ' -f$i
                  /PROJECT/TEMP/OUTSIDER_IP_ADDR`
                  # echo outsider_ip_addr = $outsider_ip_addr
                  i=`expr ${i} + 1`
                  echo $protocol -- $outsider_name --
                  $outsider_ip_addr -- $insider_ip

                  # Execute iptables (Recording process)
                  # ------------------------------------
```

```
                  echo "iptables -A FORWARD -i eth1 -p $protocol -s
                  $insider_ip -d $outsider_ip_addr -j ACCEPT" >>
                  /PROJECT/TEMP/firewall_rule
                  echo "iptables -A FORWARD -i eth0 -p $protocol -s
                  $outsider_ip_addr -d $insider_ip -j ACCEPT" >>
                  /PROJECT/TEMP/firewall_rule
            done
      fi

      # If host list contain only single IP address of Internet side
      # -------------------------------------------------------------
      if [ $outsider_number_of_ip  -eq 1 ]
      then
            echo $protocol -- $outsider_name -- $outsider_ip_addr --
            $insider_ip

            # Execute iptables (Recording process)
            # ------------------------------------
            echo "iptables -A FORWARD -i eth1 -p $protocol -s
            $insider_ip -d $outsider_ip_addr -j ACCEPT" >>
            /PROJECT/TEMP/firewall_rule
            echo "iptables -A FORWARD -i eth0 -p $protocol -s
            $outsider_ip_addr -d $insider_ip -j ACCEPT" >>
            /PROJECT/TEMP/firewall_rule
      fi
      j=`expr ${j} + 1`
done

##============================================================##
##============================================================##

###### Process Protection Level 2
###### --------------------------
echo .
echo PROTECTION LEVEL 2

echo "## ==================="        >> /PROJECT/TEMP/firewall_rule
echo "## Protection Level 2"         >> /PROJECT/TEMP/firewall_rule
echo "## ==================="        >> /PROJECT/TEMP/firewall_rule

insider_ip=$protection_level2       # Protection level 2

### Process safety level 2
### ----------------------

echo .
echo Safety Level 2

echo "# safety level 2"       >> /PROJECT/TEMP/firewall_rule
echo "# --------------"        >> /PROJECT/TEMP/firewall_rule

# to get jml baris of table
# -------------------------
baris_table=`wc -l /PROJECT/TABLE/$safety_level2 | awk '{print $1}'`
echo size table linewise "$[baris_table]"

## Process every data in table line-wise
## -------------------------------------
j=1
while [[ $j -le $baris_table ]]
do
```

```
echo .
echo .
echo $j
protocol=`head -n $j /PROJECT/TABLE/$safety_level2 | tail -n 1
| awk -F, '{print $1}'`
outsider_name=`head -n $j /PROJECT/TABLE/$safety_level2 | tail
-n 1 | awk -F, '{print $2}'`

# Finding IP address of outsider and insider machine
# ------------------------------------------------
# insider_ip=$protection_level2           # Protection level 2
outsider_ip_addr=`grep $outsider_name /PROJECT/TABLE/host_list
| awk '{print $2}'`

outsider_number_of_ip=`grep -c $outsider_name
/PROJECT/TABLE/host_list`

# If host list contain more than one IP addresses
# ------------------------------------------------
if [ $outsider_number_of_ip  -gt 1 ]
then
      echo $outsider_ip_addr > /PROJECT/TEMP/OUTSIDER_IP_ADDR
      i=1
      while [[ $i -le $outsider_number_of_ip ]]
      do
            outsider_ip_addr=`cut -d' ' -f$i
            /PROJECT/TEMP/OUTSIDER_IP_ADDR`
            # echo outsider_ip_addr = $outsider_ip_addr

            i=`expr ${i} + 1`

            echo $protocol -- $outsider_name --
            $outsider_ip_addr -- $insider_ip

            # Execute iptables (Recording process)
            # ------------------------------------
            echo "iptables -A FORWARD -i eth1 -p $protocol -s
            $insider_ip -d $outsider_ip_addr -j ACCEPT" >>
            /PROJECT/TEMP/firewall_rule
            echo "iptables -A FORWARD -i eth0 -p $protocol -s
            $outsider_ip_addr -d $insider_ip -j ACCEPT" >>
            /PROJECT/TEMP/firewall_rule
      done
fi

# If host list contain only single IP address of Internet side
# -------------------------------------------------------------
if [ $outsider_number_of_ip  -eq 1 ]
then
      echo $protocol -- $outsider_name -- $outsider_ip_addr --
      $insider_ip

      # Execute iptables (Recording process)
      # ------------------------------------
      echo "iptables -A FORWARD -i eth1 -p $protocol -s
      $insider_ip -d $outsider_ip_addr -j ACCEPT" >>
      /PROJECT/TEMP/firewall_rule
      echo "iptables -A FORWARD -i eth0 -p $protocol -s
      $outsider_ip_addr -d $insider_ip -j ACCEPT" >>
      /PROJECT/TEMP/firewall_rule
fi
```

```
      j=`expr ${j} + 1`
done


### Process safety level 3
### ----------------------

echo .
echo Safety Level 3

echo "# safety level 3"       >> /PROJECT/TEMP/firewall_rule
echo "# --------------"       >> /PROJECT/TEMP/firewall_rule

# to get jml baris of table
# -------------------------
baris_table=`wc -l /PROJECT/TABLE/$safety_level3 | awk '{print $1}'`
echo size table linewise "$[baris_table]"

## Process every data in table line-wise
## -------------------------------------
j=1
while [[ $j -le $baris_table ]]
do
      echo .
      echo .
      echo $j
      protocol=`head -n $j /PROJECT/TABLE/$safety_level3 | tail -n 1
      | awk -F, '{print $1}'`
      outsider_name=`head -n $j /PROJECT/TABLE/$safety_level3 | tail
      -n 1 | awk -F, '{print $2}'`

      # Finding IP address of outsider and insider machine
      # --------------------------------------------------
      # insider_ip=$protection_level2           # Protection level 2
      outsider_ip_addr=`grep $outsider_name /PROJECT/TABLE/host_list
      | awk '{print $2}'`

      outsider_number_of_ip=`grep -c $outsider_name
      /PROJECT/TABLE/host_list`

      # If host list contain more than one IP addresses
      # -----------------------------------------------
      if [ $outsider_number_of_ip  -gt 1 ]
      then
            echo $outsider_ip_addr > /PROJECT/TEMP/OUTSIDER_IP_ADDR
            i=1
            while [[ $i -le $outsider_number_of_ip ]]
            do
                  outsider_ip_addr=`cut -d' ' -f$i
                  /PROJECT/TEMP/OUTSIDER_IP_ADDR`
                  # echo outsider_ip_addr = $outsider_ip_addr

                  i=`expr ${i} + 1`

                  echo $protocol -- $outsider_name --
                  $outsider_ip_addr -- $insider_ip

                  # Execute iptables (Recording process)
                  # ------------------------------------
```

```
                  echo "iptables -A FORWARD -i eth1 -p $protocol -s
                  $insider_ip -d $outsider_ip_addr -j ACCEPT" >>
                  /PROJECT/TEMP/firewall_rule
                  echo "iptables -A FORWARD -i eth0 -p $protocol -s
                  $outsider_ip_addr -d $insider_ip -j ACCEPT" >>
                  /PROJECT/TEMP/firewall_rule
            done
      fi

      # If host list contain only single IP address of Internet side
      # -------------------------------------------------------------
      if [ $outsider_number_of_ip  -eq 1 ]
      then
            echo $protocol -- $outsider_name -- $outsider_ip_addr --
            $insider_ip

            # Execute iptables (Recording process)
            # ------------------------------------
            echo "iptables -A FORWARD -i eth1 -p $protocol -s
            $insider_ip -d $outsider_ip_addr -j ACCEPT" >>
            /PROJECT/TEMP/firewall_rule
            echo "iptables -A FORWARD -i eth0 -p $protocol -s
            $outsider_ip_addr -d $insider_ip -j ACCEPT" >>
            /PROJECT/TEMP/firewall_rule
      fi
      j=`expr ${j} + 1`
done


##=============================================================##
##=============================================================##

###### Process Protection Level 3
###### --------------------------
echo .
echo PROTECTION LEVEL 3

echo "## ==================="        >> /PROJECT/TEMP/firewall_rule
echo "## Protection Level 3"         >> /PROJECT/TEMP/firewall_rule
echo "## ==================="        >> /PROJECT/TEMP/firewall_rule

insider_ip=$protection_level3       # Protection level 3

### Process safety level 3
### ----------------------

echo .
echo Safety Level 3

echo "# safety level 3"       >> /PROJECT/TEMP/firewall_rule
echo "# --------------"        >> /PROJECT/TEMP/firewall_rule

# to get jml baris of table
# -------------------------
baris_table=`wc -l /PROJECT/TABLE/$safety_level3 | awk '{print $1}'`
echo size table linewise "$[baris_table]"

## Process every data in table line-wise
## -------------------------------------
j=1
while [[ $j -le $baris_table ]]
```

```
do
      echo .
      echo .
      echo $j
      protocol=`head -n $j /PROJECT/TABLE/$safety_level3 | tail -n 1
      | awk -F, '{print $1}'`
      outsider_name=`head -n $j /PROJECT/TABLE/$safety_level3 | tail
      -n 1 | awk -F, '{print $2}'`

      # Finding IP address of outsider and insider machine
      # ------------------------------------------------
      # insider_ip=$protection_level3           # Protection level 3
      outsider_ip_addr=`grep $outsider_name /PROJECT/TABLE/host_list
      | awk '{print $2}'`

      outsider_number_of_ip=`grep -c $outsider_name
      /PROJECT/TABLE/host_list`

      # If host list contain more than one IP addresses
      # ------------------------------------------------
      if [ $outsider_number_of_ip  -gt 1 ]
      then
            echo $outsider_ip_addr > /PROJECT/TEMP/OUTSIDER_IP_ADDR
            i=1
            while [[ $i -le $outsider_number_of_ip ]]
            do
                  outsider_ip_addr=`cut -d' ' -f$i
                  /PROJECT/TEMP/OUTSIDER_IP_ADDR`
                  # echo outsider_ip_addr = $outsider_ip_addr

                  i=`expr ${i} + 1`

                  echo $protocol -- $outsider_name --
                  $outsider_ip_addr -- $insider_ip

                  # Execute iptables (Recording process)
                  # ------------------------------------
                  echo "iptables -A FORWARD -i eth1 -p $protocol -s
                  $insider_ip -d $outsider_ip_addr -j ACCEPT" >>
                  /PROJECT/TEMP/firewall_rule
                  echo "iptables -A FORWARD -i eth0 -p $protocol -s
                  $outsider_ip_addr -d $insider_ip -j ACCEPT" >>
                  /PROJECT/TEMP/firewall_rule
            done
      fi

      # If host list contain only single IP address of Internet side
      # -------------------------------------------------------------
      if [ $outsider_number_of_ip  -eq 1 ]
      then
            echo $protocol -- $outsider_name -- $outsider_ip_addr --
            $insider_ip

            # Execute iptables (Recording process)
            # ------------------------------------
            echo "iptables -A FORWARD -i eth1 -p $protocol -s
            $insider_ip -d $outsider_ip_addr -j ACCEPT" >>
            /PROJECT/TEMP/firewall_rule
            echo "iptables -A FORWARD -i eth0 -p $protocol -s
            $outsider_ip_addr -d $insider_ip -j ACCEPT" >>
            /PROJECT/TEMP/firewall_rule
```

```
      fi

      j=`expr ${j} + 1`
done

##================================================================##
##================================================================##


## Setting up the ending of firewall_rule configuration file
(Recording process)
## ----------------------------------------------------------
echo "## End of configuration"      >> /PROJECT/TEMP/firewall_rule
echo "## ====================="     >> /PROJECT/TEMP/firewall_rule
echo "iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE" >>
/PROJECT/TEMP/firewall_rule
echo "echo "1" > /proc/sys/net/ipv4/ip_forward" >>
/PROJECT/TEMP/firewall_rule
echo "echo Finish Running FIREWALL CONFIGURATION" >>
/PROJECT/TEMP/firewall_rule

## Execute firewall_rule file
## --------------------------
chmod u+x /PROJECT/TEMP/firewall_rule
cd /PROJECT/TEMP
./firewall_rule


# ngrep -d eth1 -e -t > /PROJECT/TEMP/PROXY_BUFF_INT &
# To catch any incoming data from internal network and put it in
PROXY_BUFF using background process

echo "Finish"
echo .
echo .
echo .
echo FOR EXPERIMENT PURPOSES ================================
echo Start at $start
echo Finish at ...
cat /PROJECT/TEMP/TIMESTAMP
# cat /PROJECT/TEMP/MAXCOUNTER

waktu1=`date`
      waktu2=`date`
      #echo $waktu1
      #echo $waktu2
      while [[ $waktu1 = $waktu2 ]]
      do
            cat /PROJECT/TEMP/MAXCOUNTER
            waktu2=`date`
      done
      #echo $waktu2
      hitung=0
      while [ $hitung -le 100 ]
      do
            hitung=`expr ${hitung} + 1`
      done
cat /PROJECT/TEMP/MAXCOUNTER
```

APPENDIX A2

Security Policy Produced by Initialization Process

A2.1 Lattice-Based One Rule

```
PROTECTION LEVEL 3
Trusted Level 3
tcp -- www.google.com -- 63.150.131.40 -- 24.4.74.0/24
tcp -- www.google.com -- 64.233.189.104 -- 24.4.74.0/24
tcp -- www.google.com -- 64.233.161.147 -- 24.4.74.0/24
tcp -- www.google.com -- 64.233.161.104 -- 24.4.74.0/24
tcp -- www.google.com -- 66.94.229.254 -- 24.4.74.0/24
tcp -- www.google.com -- 66.102.7.104 -- 24.4.74.0/24
tcp -- www.google.com -- 66.102.11.104 -- 24.4.74.0/24
tcp -- www.google.com -- 66.102.11.99 -- 24.4.74.0/24
tcp -- www.google.com -- 66.102.9.104 -- 24.4.74.0/24
tcp -- www.google.com -- 66.102.9.99 -- 24.4.74.0/24
Trusted Level 2
tcp -- www.rd.com -- 164.109.22.93 -- 24.4.74.0/24
tcp -- www.rd.com -- 164.109.22.52 -- 24.4.74.0/24
Trusted Level 1
tcp -- www.elsevier.com -- 129.35.76.177 -- 24.4.74.0/24
.
PROTECTION LEVEL 2
Trusted Level 2
tcp -- www.rd.com -- 164.109.22.93 -- 24.4.75.0/24
tcp -- www.rd.com -- 164.109.22.52 -- 24.4.75.0/24
Trusted Level 1
tcp -- www.elsevier.com -- 129.35.76.177 -- 24.4.75.0/24
.
PROTECTION LEVEL 1
Trusted Level 1
tcp -- www.elsevier.com -- 129.35.76.177 -- 24.4.76.0/24
```

A2.2 Two Rules

```
PROTECTION LEVEL 1
Trusted Level 3
tcp -- www.google.com -- 63.150.131.40 -- 24.4.74.0/24
tcp -- www.google.com -- 64.233.189.104 -- 24.4.74.0/24
tcp -- www.google.com -- 64.233.161.147 -- 24.4.74.0/24
tcp -- www.google.com -- 64.233.161.104 -- 24.4.74.0/24
tcp -- www.google.com -- 66.94.229.254 -- 24.4.74.0/24
tcp -- www.google.com -- 66.102.7.104 -- 24.4.74.0/24
tcp -- www.google.com -- 66.102.11.104 -- 24.4.74.0/24
tcp -- www.google.com -- 66.102.11.99 -- 24.4.74.0/24
tcp -- www.google.com -- 66.102.9.104 -- 24.4.74.0/24
tcp -- www.google.com -- 66.102.9.99 -- 24.4.74.0/24
tcp -- www.altavista.com -- 66.94.229.254 -- 24.4.74.0/24
tcp -- www.altavista.com -- 66.94.230.163 -- 24.4.74.0/24
tcp -- www.altavista.com -- 63.150.131.40 -- 24.4.74.0/24
tcp -- www.altavista.com -- 63.150.131.24 -- 24.4.74.0/24
Trusted Level 2
tcp -- www.rd.com -- 164.109.22.93 -- 24.4.74.0/24
tcp -- www.rd.com -- 164.109.22.52 -- 24.4.74.0/24
tcp -- www.kompas.com -- 64.203.71.11 -- 24.4.74.0/24
tcp -- www.kompas.com -- 64.203.71.51 -- 24.4.74.0/24
Trusted Level 1
tcp -- www.elsevier.com -- 129.35.76.177 -- 24.4.74.0/24
tcp -- www.acm.org -- 199.222.69.251 -- 24.4.74.0/24
.
PROTECTION LEVEL 2
Trusted Level 2
tcp -- www.rd.com -- 164.109.22.93 -- 24.4.75.0/24
tcp -- www.rd.com -- 164.109.22.52 -- 24.4.75.0/24
tcp -- www.kompas.com -- 64.203.71.11 -- 24.4.75.0/24
tcp -- www.kompas.com -- 64.203.71.51 -- 24.4.75.0/24
Trusted Level 1
tcp -- www.elsevier.com -- 129.35.76.177 -- 24.4.75.0/24
tcp -- www.acm.org -- 199.222.69.251 -- 24.4.75.0/24
.
PROTECTION LEVEL 3
Trusted Level 1
tcp -- www.elsevier.com -- 129.35.76.177 -- 24.4.76.0/24
tcp -- www.acm.org -- 199.222.69.251 -- 24.4.76.0/24
```

APPENDIX A1

The Script of Lattice-Based Initialization Process

```
#!/bin/sh

echo ============================================================
echo LATTICE-BASED FIREWALL : which is not stated is prohibited
echo ============================================================

start=`cat /PROJECT/TEMP/TIMESTAMP`

# Implementation of Lattice is conducted on initialization
# eth0 = EXTERNAL NETWORK
# eth1 = INTERNAL NETWORK

# ------------------
# 1. Initialization
# ------------------

# INTERNAL_SUBNET="24.4.74"

> /PROJECT/TEMP/PROXY_BUFF_INT

# > /PROJECT/TEMP/PROXY_BUFF_EXT

> /PROJECT/TEMP/IP_SENDER
> /PROJECT/TEMP/IP_ADDR_SENDER
> /PROJECT/TEMP/IP_ADDR_RECEIVER
> /PROJECT/TEMP/TABLEIP
> /PROJECT/TEMP/HISTORY

> /PROJECT/TEMP/firewall_rule

# > /PROJECT/TEMP/DELTATIME
# > /PROJECT/TEMP/TIMEOUTPUT

> /PROJECT/TEMP/LOG

cd /PROJECT/FIREWALL
# To preconfigure firewall machine

echo "0" > /proc/sys/net/ipv4/ip_forward

#### Setting up firewall_rule configuration file
## -------------------------------------------

echo "## Preconfiguration"    >> /PROJECT/TEMP/firewall_rule
echo "## ================"    >> /PROJECT/TEMP/firewall_rule

# Default policy (Recording process)
# ----------------------------------
echo "iptables -P INPUT DROP"    >> /PROJECT/TEMP/firewall_rule
echo "iptables -P OUTPUT ACCEPT" >> /PROJECT/TEMP/firewall_rule
echo "iptables -P FORWARD DROP"  >> /PROJECT/TEMP/firewall_rule

# Flushing the existing rule (Recording process)
# ----------------------------------------------
echo "iptables --flush" >> /PROJECT/TEMP/firewall_rule
echo "iptables --table nat --flush" >> /PROJECT/TEMP/firewall_rule
echo "iptables --delete-chain" >> /PROJECT/TEMP/firewall_rule
echo "iptables --table nat --delete-chain" >>
/PROJECT/TEMP/firewall_rule
```

```
# Setting up Protection Level
# ---------------------------
protection_level1="24.4.74.0/24"    # full-protected
protection_level2="24.4.75.0/24"
protection_level3="24.4.76.0/24"    # unprotected

# Setting up Safety Level
# -----------------------
safety_level1="table1"              # un-trusted
safety_level2="table2"
safety_level3="table3"              # full- trusted

##====================================================================##
##====================================================================##

###### Process Protection Level 1
###### -------------------------
echo .
echo PROTECTION LEVEL 1

echo "## ==================" >> /PROJECT/TEMP/firewall_rule
echo "## Protection Level 1" >> /PROJECT/TEMP/firewall_rule
echo "## ==================" >> /PROJECT/TEMP/firewall_rule

insider_ip=$protection_level1       # Protection level 1

### Process safety level 1
### ----------------------

echo .
echo Safety Level 1

echo "# safety level 1" >> /PROJECT/TEMP/firewall_rule
echo "# --------------" >> /PROJECT/TEMP/firewall_rule

# to get jml baris of table
# -------------------------
baris_table=`wc -l /PROJECT/TABLE/$safety_level1 | awk '{print $1}'`
echo size table linewise "$[baris_table]"

## Process every data in table line-wise
## -------------------------------------
j=1
while [[ $j -le $baris_table ]]
do
      echo .
      echo .
      echo $j
      protocol=`head -n $j /PROJECT/TABLE/$safety_level1 | tail -n 1
      | awk -F, '{print $1}'`
      outsider_name=`head -n $j /PROJECT/TABLE/$safety_level1 | tail
      -n 1 | awk -F, '{print $2}'`

      # Finding IP address of outsider and insider machine
      # --------------------------------------------------
      # insider_ip=$protection_level1           # Protection level 1
      outsider_ip_addr=`grep $outsider_name /PROJECT/TABLE/host_list
      | awk '{print $2}'`

      outsider_number_of_ip=`grep -c $outsider_name
      /PROJECT/TABLE/host_list`
```

```
      # If host list contain more than one IP addresses
      # ------------------------------------------------
      if [ $outsider_number_of_ip  -gt 1 ]
      then
            echo $outsider_ip_addr > /PROJECT/TEMP/OUTSIDER_IP_ADDR
            i=1
            while [[ $i -le $outsider_number_of_ip ]]
            do
                  outsider_ip_addr=`cut -d' ' -f$i
                  /PROJECT/TEMP/OUTSIDER_IP_ADDR`
                  # echo outsider_ip_addr = $outsider_ip_addr
                  i=`expr ${i} + 1`
                  echo $protocol -- $outsider_name --
                  $outsider_ip_addr -- $insider_ip

                  # Execute iptables (Recording process)
                  # ------------------------------------
                  echo "iptables -A FORWARD -i eth1 -p $protocol -s
                  $insider_ip -d $outsider_ip_addr -j ACCEPT" >>
                  /PROJECT/TEMP/firewall_rule
                  echo "iptables -A FORWARD -i eth0 -p $protocol -s
                  $outsider_ip_addr -d $insider_ip -j ACCEPT" >>
                  /PROJECT/TEMP/firewall_rule
            done
      fi

      # If host list contain only single IP address of Internet side
      # -------------------------------------------------------------
      if [ $outsider_number_of_ip  -eq 1 ]
      then
            echo $protocol -- $outsider_name -- $outsider_ip_addr --
            $insider_ip

            # Execute iptables (Recording process)
            # ------------------------------------
            echo "iptables -A FORWARD -i eth1 -p $protocol -s
            $insider_ip -d $outsider_ip_addr -j ACCEPT" >>
            /PROJECT/TEMP/firewall_rule
            echo "iptables -A FORWARD -i eth0 -p $protocol -s
            $outsider_ip_addr -d $insider_ip -j ACCEPT" >>
            /PROJECT/TEMP/firewall_rule
      fi
      j=`expr ${j} + 1`
done

### Process safety level 2
### ----------------------

echo .
echo Safety Level 2

echo "# safety level 2"  >> /PROJECT/TEMP/firewall_rule
echo "# --------------"  >> /PROJECT/TEMP/firewall_rule

# to get jml baris of table
# -------------------------
baris_table=`wc -l /PROJECT/TABLE/$safety_level2 | awk '{print $1}'`
echo size table linewise "$[baris_table]"

## Process every data in table line-wise
## -------------------------------------
```

```
j=1
while [[ $j -le $baris_table ]]
do
      echo .
      echo .
      echo $j
      protocol=`head -n $j /PROJECT/TABLE/$safety_level2 | tail -n 1
      | awk -F, '{print $1}'`
      outsider_name=`head -n $j /PROJECT/TABLE/$safety_level2 | tail
      -n 1 | awk -F, '{print $2}'`

      # Finding IP address of outsider and insider machine
      # --------------------------------------------------
      # insider_ip=$protection_level1           # Protection level 1
      outsider_ip_addr=`grep $outsider_name /PROJECT/TABLE/host_list
      | awk '{print $2}'`

      outsider_number_of_ip=`grep -c $outsider_name
      /PROJECT/TABLE/host_list`

      # If host list contain more than one IP addresses
      # -----------------------------------------------
      if [ $outsider_number_of_ip  -gt 1 ]
      then
            echo $outsider_ip_addr > /PROJECT/TEMP/OUTSIDER_IP_ADDR
            i=1
            while [[ $i -le $outsider_number_of_ip ]]
            do
                  outsider_ip_addr=`cut -d' ' -f$i
                  /PROJECT/TEMP/OUTSIDER_IP_ADDR`
                  # echo outsider_ip_addr = $outsider_ip_addr

                  i=`expr ${i} + 1`

                  echo $protocol -- $outsider_name --
                  $outsider_ip_addr -- $insider_ip

                  # Execute iptables (Recording process)
                  # ------------------------------------
                  echo "iptables -A FORWARD -i eth1 -p $protocol -s
                  $insider_ip -d $outsider_ip_addr -j ACCEPT" >>
                  /PROJECT/TEMP/firewall_rule
                  echo "iptables -A FORWARD -i eth0 -p $protocol -s
                  $outsider_ip_addr -d $insider_ip -j ACCEPT" >>
                  /PROJECT/TEMP/firewall_rule
            done
      fi

      # If host list contain only single IP address of Internet side
      # -------------------------------------------------------------
      if [ $outsider_number_of_ip  -eq 1 ]
      then
            echo $protocol -- $outsider_name -- $outsider_ip_addr --
            $insider_ip

            # Execute iptables (Recording process)
            # ------------------------------------
            echo "iptables -A FORWARD -i eth1 -p $protocol -s
            $insider_ip -d $outsider_ip_addr -j ACCEPT" >>
            /PROJECT/TEMP/firewall_rule
```

```
            echo "iptables -A FORWARD -i eth0 -p $protocol -s
            $outsider_ip_addr -d $insider_ip -j ACCEPT" >>
            /PROJECT/TEMP/firewall_rule
      fi
      j=`expr ${j} + 1`
done

### Process safety level 3
### ----------------------

echo .
echo Safety Level 3

echo "# safety level 3"       >> /PROJECT/TEMP/firewall_rule
echo "# --------------"       >> /PROJECT/TEMP/firewall_rule

# to get jml baris of table
# -------------------------
baris_table=`wc -l /PROJECT/TABLE/$safety_level3 | awk '{print $1}'`
echo size table linewise "$[baris_table]"

## Process every data in table line-wise
## -------------------------------------
j=1
while [[ $j -le $baris_table ]]
do
      echo .
      echo .
      echo $j
      protocol=`head -n $j /PROJECT/TABLE/$safety_level3 | tail -n 1
      | awk -F, '{print $1}'`
      outsider_name=`head -n $j /PROJECT/TABLE/$safety_level3 | tail
      -n 1 | awk -F, '{print $2}'`

      # Finding IP address of outsider and insider machine
      # --------------------------------------------------
      # insider_ip=$protection_level1           # Protection level 1
      outsider_ip_addr=`grep $outsider_name /PROJECT/TABLE/host_list
      | awk '{print $2}'`

      outsider_number_of_ip=`grep -c $outsider_name
      /PROJECT/TABLE/host_list`

      # If host list contain more than one IP addresses
      # -----------------------------------------------
      if [ $outsider_number_of_ip  -gt 1 ]
      then
            echo $outsider_ip_addr > /PROJECT/TEMP/OUTSIDER_IP_ADDR
            i=1
            while [[ $i -le $outsider_number_of_ip ]]
            do
                  outsider_ip_addr=`cut -d' ' -f$i
                  /PROJECT/TEMP/OUTSIDER_IP_ADDR`
                  # echo outsider_ip_addr = $outsider_ip_addr
                  i=`expr ${i} + 1`
                  echo $protocol -- $outsider_name --
                  $outsider_ip_addr -- $insider_ip

                  # Execute iptables (Recording process)
                  # ------------------------------------
```

```
                  echo "iptables -A FORWARD -i eth1 -p $protocol -s
                  $insider_ip -d $outsider_ip_addr -j ACCEPT" >>
                  /PROJECT/TEMP/firewall_rule
                  echo "iptables -A FORWARD -i eth0 -p $protocol -s
                  $outsider_ip_addr -d $insider_ip -j ACCEPT" >>
                  /PROJECT/TEMP/firewall_rule
            done
      fi

      # If host list contain only single IP address of Internet side
      # -------------------------------------------------------------
      if [ $outsider_number_of_ip  -eq 1 ]
      then
            echo $protocol -- $outsider_name -- $outsider_ip_addr --
            $insider_ip

            # Execute iptables (Recording process)
            # ------------------------------------
            echo "iptables -A FORWARD -i eth1 -p $protocol -s
            $insider_ip -d $outsider_ip_addr -j ACCEPT" >>
            /PROJECT/TEMP/firewall_rule
            echo "iptables -A FORWARD -i eth0 -p $protocol -s
            $outsider_ip_addr -d $insider_ip -j ACCEPT" >>
            /PROJECT/TEMP/firewall_rule
      fi
      j=`expr ${j} + 1`
done

##============================================================##
##============================================================##

###### Process Protection Level 2
###### --------------------------
echo .
echo PROTECTION LEVEL 2

echo "## ==================="        >> /PROJECT/TEMP/firewall_rule
echo "## Protection Level 2"         >> /PROJECT/TEMP/firewall_rule
echo "## ==================="        >> /PROJECT/TEMP/firewall_rule

insider_ip=$protection_level2       # Protection level 2

### Process safety level 2
### ----------------------

echo .
echo Safety Level 2

echo "# safety level 2"       >> /PROJECT/TEMP/firewall_rule
echo "# --------------"        >> /PROJECT/TEMP/firewall_rule

# to get jml baris of table
# -------------------------
baris_table=`wc -l /PROJECT/TABLE/$safety_level2 | awk '{print $1}'`
echo size table linewise "$[baris_table]"

## Process every data in table line-wise
## -------------------------------------
j=1
while [[ $j -le $baris_table ]]
do
```

```
echo .
echo .
echo $j
protocol=`head -n $j /PROJECT/TABLE/$safety_level2 | tail -n 1
| awk -F, '{print $1}'`
outsider_name=`head -n $j /PROJECT/TABLE/$safety_level2 | tail
-n 1 | awk -F, '{print $2}'`

# Finding IP address of outsider and insider machine
# ------------------------------------------------
# insider_ip=$protection_level2           # Protection level 2
outsider_ip_addr=`grep $outsider_name /PROJECT/TABLE/host_list
| awk '{print $2}'`

outsider_number_of_ip=`grep -c $outsider_name
/PROJECT/TABLE/host_list`

# If host list contain more than one IP addresses
# ------------------------------------------------
if [ $outsider_number_of_ip  -gt 1 ]
then
      echo $outsider_ip_addr > /PROJECT/TEMP/OUTSIDER_IP_ADDR
      i=1
      while [[ $i -le $outsider_number_of_ip ]]
      do
            outsider_ip_addr=`cut -d' ' -f$i
            /PROJECT/TEMP/OUTSIDER_IP_ADDR`
            # echo outsider_ip_addr = $outsider_ip_addr

            i=`expr ${i} + 1`

            echo $protocol -- $outsider_name --
            $outsider_ip_addr -- $insider_ip

            # Execute iptables (Recording process)
            # ------------------------------------
            echo "iptables -A FORWARD -i eth1 -p $protocol -s
            $insider_ip -d $outsider_ip_addr -j ACCEPT" >>
            /PROJECT/TEMP/firewall_rule
            echo "iptables -A FORWARD -i eth0 -p $protocol -s
            $outsider_ip_addr -d $insider_ip -j ACCEPT" >>
            /PROJECT/TEMP/firewall_rule
      done
fi

# If host list contain only single IP address of Internet side
# -------------------------------------------------------------
if [ $outsider_number_of_ip  -eq 1 ]
then
      echo $protocol -- $outsider_name -- $outsider_ip_addr --
      $insider_ip

      # Execute iptables (Recording process)
      # ------------------------------------
      echo "iptables -A FORWARD -i eth1 -p $protocol -s
      $insider_ip -d $outsider_ip_addr -j ACCEPT" >>
      /PROJECT/TEMP/firewall_rule
      echo "iptables -A FORWARD -i eth0 -p $protocol -s
      $outsider_ip_addr -d $insider_ip -j ACCEPT" >>
      /PROJECT/TEMP/firewall_rule
fi
```

```
      j=`expr ${j} + 1`
done


### Process safety level 3
### ----------------------

echo .
echo Safety Level 3

echo "# safety level 3"       >> /PROJECT/TEMP/firewall_rule
echo "# --------------"       >> /PROJECT/TEMP/firewall_rule

# to get jml baris of table
# -------------------------
baris_table=`wc -l /PROJECT/TABLE/$safety_level3 | awk '{print $1}'`
echo size table linewise "$[baris_table]"

## Process every data in table line-wise
## -------------------------------------
j=1
while [[ $j -le $baris_table ]]
do
      echo .
      echo .
      echo $j
      protocol=`head -n $j /PROJECT/TABLE/$safety_level3 | tail -n 1
      | awk -F, '{print $1}'`
      outsider_name=`head -n $j /PROJECT/TABLE/$safety_level3 | tail
      -n 1 | awk -F, '{print $2}'`

      # Finding IP address of outsider and insider machine
      # --------------------------------------------------
      # insider_ip=$protection_level2           # Protection level 2
      outsider_ip_addr=`grep $outsider_name /PROJECT/TABLE/host_list
      | awk '{print $2}'`

      outsider_number_of_ip=`grep -c $outsider_name
      /PROJECT/TABLE/host_list`

      # If host list contain more than one IP addresses
      # -----------------------------------------------
      if [ $outsider_number_of_ip  -gt 1 ]
      then
            echo $outsider_ip_addr > /PROJECT/TEMP/OUTSIDER_IP_ADDR
            i=1
            while [[ $i -le $outsider_number_of_ip ]]
            do
                  outsider_ip_addr=`cut -d' ' -f$i
                  /PROJECT/TEMP/OUTSIDER_IP_ADDR`
                  # echo outsider_ip_addr = $outsider_ip_addr

                  i=`expr ${i} + 1`

                  echo $protocol -- $outsider_name --
                  $outsider_ip_addr -- $insider_ip

                  # Execute iptables (Recording process)
                  # ------------------------------------
```

```
                  echo "iptables -A FORWARD -i eth1 -p $protocol -s
                  $insider_ip -d $outsider_ip_addr -j ACCEPT" >>
                  /PROJECT/TEMP/firewall_rule
                  echo "iptables -A FORWARD -i eth0 -p $protocol -s
                  $outsider_ip_addr -d $insider_ip -j ACCEPT" >>
                  /PROJECT/TEMP/firewall_rule
            done
      fi

      # If host list contain only single IP address of Internet side
      # -------------------------------------------------------------
      if [ $outsider_number_of_ip  -eq 1 ]
      then
            echo $protocol -- $outsider_name -- $outsider_ip_addr --
            $insider_ip

            # Execute iptables (Recording process)
            # ------------------------------------
            echo "iptables -A FORWARD -i eth1 -p $protocol -s
            $insider_ip -d $outsider_ip_addr -j ACCEPT" >>
            /PROJECT/TEMP/firewall_rule
            echo "iptables -A FORWARD -i eth0 -p $protocol -s
            $outsider_ip_addr -d $insider_ip -j ACCEPT" >>
            /PROJECT/TEMP/firewall_rule
      fi
      j=`expr ${j} + 1`
done


##=============================================================##
##=============================================================##

###### Process Protection Level 3
###### --------------------------
echo .
echo PROTECTION LEVEL 3

echo "## ==================="        >> /PROJECT/TEMP/firewall_rule
echo "## Protection Level 3"         >> /PROJECT/TEMP/firewall_rule
echo "## ==================="        >> /PROJECT/TEMP/firewall_rule

insider_ip=$protection_level3       # Protection level 3

### Process safety level 3
### ----------------------

echo .
echo Safety Level 3

echo "# safety level 3"       >> /PROJECT/TEMP/firewall_rule
echo "# --------------"        >> /PROJECT/TEMP/firewall_rule

# to get jml baris of table
# -------------------------
baris_table=`wc -l /PROJECT/TABLE/$safety_level3 | awk '{print $1}'`
echo size table linewise "$[baris_table]"

## Process every data in table line-wise
## -------------------------------------
j=1
while [[ $j -le $baris_table ]]
```

```
do
      echo .
      echo .
      echo $j
      protocol=`head -n $j /PROJECT/TABLE/$safety_level3 | tail -n 1
      | awk -F, '{print $1}'`
      outsider_name=`head -n $j /PROJECT/TABLE/$safety_level3 | tail
      -n 1 | awk -F, '{print $2}'`

      # Finding IP address of outsider and insider machine
      # ------------------------------------------------
      # insider_ip=$protection_level3           # Protection level 3
      outsider_ip_addr=`grep $outsider_name /PROJECT/TABLE/host_list
      | awk '{print $2}'`

      outsider_number_of_ip=`grep -c $outsider_name
      /PROJECT/TABLE/host_list`

      # If host list contain more than one IP addresses
      # ------------------------------------------------
      if [ $outsider_number_of_ip  -gt 1 ]
      then
            echo $outsider_ip_addr > /PROJECT/TEMP/OUTSIDER_IP_ADDR
            i=1
            while [[ $i -le $outsider_number_of_ip ]]
            do
                  outsider_ip_addr=`cut -d' ' -f$i
                  /PROJECT/TEMP/OUTSIDER_IP_ADDR`
                  # echo outsider_ip_addr = $outsider_ip_addr

                  i=`expr ${i} + 1`

                  echo $protocol -- $outsider_name --
                  $outsider_ip_addr -- $insider_ip

                  # Execute iptables (Recording process)
                  # ------------------------------------
                  echo "iptables -A FORWARD -i eth1 -p $protocol -s
                  $insider_ip -d $outsider_ip_addr -j ACCEPT" >>
                  /PROJECT/TEMP/firewall_rule
                  echo "iptables -A FORWARD -i eth0 -p $protocol -s
                  $outsider_ip_addr -d $insider_ip -j ACCEPT" >>
                  /PROJECT/TEMP/firewall_rule
            done
      fi

      # If host list contain only single IP address of Internet side
      # -------------------------------------------------------------
      if [ $outsider_number_of_ip  -eq 1 ]
      then
            echo $protocol -- $outsider_name -- $outsider_ip_addr --
            $insider_ip

            # Execute iptables (Recording process)
            # ------------------------------------
            echo "iptables -A FORWARD -i eth1 -p $protocol -s
            $insider_ip -d $outsider_ip_addr -j ACCEPT" >>
            /PROJECT/TEMP/firewall_rule
            echo "iptables -A FORWARD -i eth0 -p $protocol -s
            $outsider_ip_addr -d $insider_ip -j ACCEPT" >>
            /PROJECT/TEMP/firewall_rule
```

```
      fi

      j=`expr ${j} + 1`
done

##================================================================##
##================================================================##


## Setting up the ending of firewall_rule configuration file
(Recording process)
## ----------------------------------------------------------
echo "## End of configuration"      >> /PROJECT/TEMP/firewall_rule
echo "## ====================="     >> /PROJECT/TEMP/firewall_rule
echo "iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE" >>
/PROJECT/TEMP/firewall_rule
echo "echo "1" > /proc/sys/net/ipv4/ip_forward" >>
/PROJECT/TEMP/firewall_rule
echo "echo Finish Running FIREWALL CONFIGURATION" >>
/PROJECT/TEMP/firewall_rule

## Execute firewall_rule file
## --------------------------
chmod u+x /PROJECT/TEMP/firewall_rule
cd /PROJECT/TEMP
./firewall_rule


# ngrep -d eth1 -e -t > /PROJECT/TEMP/PROXY_BUFF_INT &
# To catch any incoming data from internal network and put it in
PROXY_BUFF using background process

echo "Finish"
echo .
echo .
echo .
echo FOR EXPERIMENT PURPOSES ================================
echo Start at $start
echo Finish at ...
cat /PROJECT/TEMP/TIMESTAMP
# cat /PROJECT/TEMP/MAXCOUNTER

waktu1=`date`
      waktu2=`date`
      #echo $waktu1
      #echo $waktu2
      while [[ $waktu1 = $waktu2 ]]
      do
            cat /PROJECT/TEMP/MAXCOUNTER
            waktu2=`date`
      done
      #echo $waktu2
      hitung=0
      while [ $hitung -le 100 ]
      do
            hitung=`expr ${hitung} + 1`
      done
cat /PROJECT/TEMP/MAXCOUNTER
```

APPENDIX A2

Security Policy Produced by Initialization Process

A2.1 Lattice-Based One Rule

```
PROTECTION LEVEL 3
Trusted Level 3
tcp -- www.google.com -- 63.150.131.40 -- 24.4.74.0/24
tcp -- www.google.com -- 64.233.189.104 -- 24.4.74.0/24
tcp -- www.google.com -- 64.233.161.147 -- 24.4.74.0/24
tcp -- www.google.com -- 64.233.161.104 -- 24.4.74.0/24
tcp -- www.google.com -- 66.94.229.254 -- 24.4.74.0/24
tcp -- www.google.com -- 66.102.7.104 -- 24.4.74.0/24
tcp -- www.google.com -- 66.102.11.104 -- 24.4.74.0/24
tcp -- www.google.com -- 66.102.11.99 -- 24.4.74.0/24
tcp -- www.google.com -- 66.102.9.104 -- 24.4.74.0/24
tcp -- www.google.com -- 66.102.9.99 -- 24.4.74.0/24
Trusted Level 2
tcp -- www.rd.com -- 164.109.22.93 -- 24.4.74.0/24
tcp -- www.rd.com -- 164.109.22.52 -- 24.4.74.0/24
Trusted Level 1
tcp -- www.elsevier.com -- 129.35.76.177 -- 24.4.74.0/24
.
PROTECTION LEVEL 2
Trusted Level 2
tcp -- www.rd.com -- 164.109.22.93 -- 24.4.75.0/24
tcp -- www.rd.com -- 164.109.22.52 -- 24.4.75.0/24
Trusted Level 1
tcp -- www.elsevier.com -- 129.35.76.177 -- 24.4.75.0/24
.
PROTECTION LEVEL 1
Trusted Level 1
tcp -- www.elsevier.com -- 129.35.76.177 -- 24.4.76.0/24
```

A2.2 Two Rules

```
PROTECTION LEVEL 1
Trusted Level 3
tcp -- www.google.com -- 63.150.131.40 -- 24.4.74.0/24
tcp -- www.google.com -- 64.233.189.104 -- 24.4.74.0/24
tcp -- www.google.com -- 64.233.161.147 -- 24.4.74.0/24
tcp -- www.google.com -- 64.233.161.104 -- 24.4.74.0/24
tcp -- www.google.com -- 66.94.229.254 -- 24.4.74.0/24
tcp -- www.google.com -- 66.102.7.104 -- 24.4.74.0/24
tcp -- www.google.com -- 66.102.11.104 -- 24.4.74.0/24
tcp -- www.google.com -- 66.102.11.99 -- 24.4.74.0/24
tcp -- www.google.com -- 66.102.9.104 -- 24.4.74.0/24
tcp -- www.google.com -- 66.102.9.99 -- 24.4.74.0/24
tcp -- www.altavista.com -- 66.94.229.254 -- 24.4.74.0/24
tcp -- www.altavista.com -- 66.94.230.163 -- 24.4.74.0/24
tcp -- www.altavista.com -- 63.150.131.40 -- 24.4.74.0/24
tcp -- www.altavista.com -- 63.150.131.24 -- 24.4.74.0/24
Trusted Level 2
tcp -- www.rd.com -- 164.109.22.93 -- 24.4.74.0/24
tcp -- www.rd.com -- 164.109.22.52 -- 24.4.74.0/24
tcp -- www.kompas.com -- 64.203.71.11 -- 24.4.74.0/24
tcp -- www.kompas.com -- 64.203.71.51 -- 24.4.74.0/24
Trusted Level 1
tcp -- www.elsevier.com -- 129.35.76.177 -- 24.4.74.0/24
tcp -- www.acm.org -- 199.222.69.251 -- 24.4.74.0/24
.
PROTECTION LEVEL 2
Trusted Level 2
tcp -- www.rd.com -- 164.109.22.93 -- 24.4.75.0/24
tcp -- www.rd.com -- 164.109.22.52 -- 24.4.75.0/24
tcp -- www.kompas.com -- 64.203.71.11 -- 24.4.75.0/24
tcp -- www.kompas.com -- 64.203.71.51 -- 24.4.75.0/24
Trusted Level 1
tcp -- www.elsevier.com -- 129.35.76.177 -- 24.4.75.0/24
tcp -- www.acm.org -- 199.222.69.251 -- 24.4.75.0/24
.
PROTECTION LEVEL 3
Trusted Level 1
tcp -- www.elsevier.com -- 129.35.76.177 -- 24.4.76.0/24
tcp -- www.acm.org -- 199.222.69.251 -- 24.4.76.0/24
```

APPENDIX B
The Implementation of Adaptively Updating Security Rules using Fuzzy Reasoning

```
Function DeltaTime(Waktu1 As String, Waktu2 As String) As Double
   Dim detik1, detik2 As Double
   Dim hasil_split() As String
   hasil_split() = Split(Waktu1, ":")
   detik1 = CDbl(hasil_split(0)) * 3600 + CDbl(hasil_split(0)) * 60 + CDbl(hasil_split(2))
   hasil_split() = Split(Waktu2, ":")
   detik2 = CDbl(hasil_split(0)) * 3600 + CDbl(hasil_split(0)) * 60 + CDbl(hasil_split(2))
   DeltaTime = Abs(detik1 - detik2)
End Function

Function ReadTextFileContents(filename As String) As String
   Dim fnum As Integer, isOpen As Boolean
   On Error GoTo Error_Handler
   ' Get the next free file number.
   fnum = FreeFile()
   Open filename For Input As #fnum
   ' If execution flow got here, the file has been open without error.
   isOpen = True
   ' Read the entire contents in one single operation.
   ReadTextFileContents = Input(LOF(fnum), fnum)
   ' Intentionally flow into the error handler to close the file.
Error_Handler:
   ' Raise the error (if any), but first close the file.
   If isOpen Then Close #fnum
   'If Err Then Err.Raise Err.Number, , Err.Description
End Function
Sub WriteTextFileContents(Text As String, filename As String, Optional AppendMode As Boolean)
   Dim fnum As Integer, isOpen As Boolean
   On Error GoTo Error_Handler
   ' Get the next free file number.
   fnum = FreeFile()
   If AppendMode Then
      Open filename For Append As #fnum
    Else
      Open filename For Output As #fnum
    End If
    ' If execution flow gets here, the file has been opened correctly.
    isOpen = True
    ' Print to the file in one single operation.
    Print #fnum, Text
    ' Intentionally flow into the error handler to close the file.
Error_Handler:
   ' Raise the error (if any), but first close the file.
   If isOpen Then Close #fnum
   'If Err Then Err.Raise Err.Number, , Err.Description
End Sub

Private Sub End_Click()

End Sub

Private Sub EndAnalysis_Click()
   Timer1.Enabled = False
   Run.Enabled = True
   EndAnalysis.Enabled = False
End Sub
```

```
Private Sub Form_Load()
   Timer1.Enabled = False
   EndAnalysis.Enabled = False
End Sub

Private Sub Run_Click()
   EndAnalysis.Enabled = True
   Run.Enabled = False
   Timer1.Enabled = True
   Timer1.Interval = 3000
End Sub

Private Sub Timer1_Timer()
   Dim Nama_Traffic_Buffer As String
   Dim Kumpulan_Kata(1 To 1000000) As String
   Dim List_Outsider(1 To 1000) As String
   Dim List_Internal(1 To 1000) As String
   Dim List_AccessTime(1 To 1000) As String
   'Dim List_Kalimat(1 To 1000) As String
   Dim List_Bahaya1(1 To 1000) As Integer
   Dim List_Bahaya2(1 To 1000) As Integer
   Dim List_Bahaya3(1 To 1000) As Integer
   Dim List_Bahaya4(1 To 1000) As Integer
   Dim List_Bahaya5(1 To 1000) As Integer
   Dim List_Bahaya6(1 To 1000) As Integer
   Dim List_Risk(1 To 1000) As Double
   Dim List_Table(1 To 1000) As String
   Dim Table1_array(1 To 100) As String
   Dim Table2_array(1 To 100) As String
   Dim Table3_array(1 To 100) As String
   Dim Suspicious(1 To 100) As String
   Dim Temporary_array(1 To 100) As String
   Dim Traffic_Buffer, Huruf, Kalimat, Kata As String
   Dim Table1_Buffer, Table2_Buffer, Table3_Buffer As String
   Dim Batas_Table1, Batas_Table2, Batas_Table3 As Long
   Dim fnum1 As Integer
   Dim i, j, k, l, m, Batas_buffer, Batas_kalimat, Batas_kumpulan_kata As Long
   Dim filter_result As String
   Dim hasil_split() As String
   Dim Intranet As String, Outside_Party As String, Internal_Party As String
   Dim Bahaya, batas_index_outsider As Long
   Dim status_exist As Boolean
   Dim status_Threat As Boolean
   Dim Report_File As String
   Dim Table1 As String
   Dim Table2 As String
   Dim Table3 As String
   Dim Total_Outsider, Total_Bahaya1, Total_Bahaya2, Total_Bahaya3, Total_Bahaya5,
Total_Bahaya6 As Long
   Dim Waktu_access1 As String, Waktu_access2 As String
   Dim BedaWaktu As Double
   Dim Forced_Information As Integer
   Dim Risk_Bahaya1, Risk_Bahaya2, Risk_Bahaya3, Risk_Bahaya4, Risk_Bahaya5, Risk_Bahaya6,
Total_Risk As Double

   'If Text3.Text = "" Then
   '    Exit Sub
   'End If
```

```
Text1.Text = "Start at " & Now
Text2.Text = "In progress ... "
Text1.Refresh
Text2.Refresh
List1.Clear
List1.Refresh


'ANALYZING TRAFFIC BUFFER
'========================

Intranet = "24.4.74"
'Suspicious(1) = "script src"
'Suspicious(2) = "Set-Cookie:"
'Suspicious(3) = "<script language="
'Suspicious(2) = ".js"""
'Suspicious(3) = ".js>"
'Suspicious(2) = "<script language ="
'Suspicious(3) = "<script language="

Suspicious(1) = ".exe"""           'EXE
Suspicious(2) = ".exe>"            'EXE
Suspicious(3) = "</script>"        'active script
Suspicious(4) = "Set-Cookie"       'cookie
Suspicious(5) = "Set Cookie"       'cookie
Suspicious(6) = "SetCookie"        'cookie
Suspicious(7) = ".exe "            'EXE
Suspicious(8) = ".exe|"            'EXE
Suspicious(9) = "window.open"      'advertisement
Suspicious(10) = "popup("          'advertisement
Suspicious(11) = "if (exit)"       'forced info
Suspicious(12) = "exit=false"      'forced info
Suspicious(13) = "exit=true"       'forced info
Suspicious(14) = "checkexit"       'forced info
Suspicious(15) = "ControlExit"     'forced info

'Nama_Traffic_Buffer = "D:\Project\Traffic Analyzer\Data\" + Text3.Text
'Nama_Traffic_Buffer = "D:\Project\Data\Threat\" + Text3.Text
Nama_Traffic_Buffer = "D:\Project\Ngrep\fuzzy_buffer"
Traffic_Buffer = ReadTextFileContents(Nama_Traffic_Buffer) 'Copy is traffic buffer
WriteTextFileContents (""), Nama_Traffic_Buffer, False 'Reset traffic buffer
Batas_buffer = Len(Traffic_Buffer)

'Start analyzing buffer file
Outside_Party = ""
batas_index_outsider = 0
i = 1
Do Until i >= Batas_buffer
   StatusBar1.SimpleText = "Browsing buffer at " & i & " round"
   'StatusBar1.Refresh
   Do Until (Huruf = Chr(10)) Or (i >= Batas_buffer)
      Huruf = Mid$(Traffic_Buffer, i, 1)
      If (Huruf = Chr(13)) Or (Huruf = "#") Or (Huruf = Chr(10)) Then
      Else
         Kalimat = Kalimat + Huruf
      End If
      i = i + 1
   Loop 'result is kalimat (character before enter)

   'Start analyzing sentence (character before enter)
```

```
        Batas_kalimat = Len(Kalimat)
        j = 1
        k = 1
        Do Until j > Batas_kalimat
           Do Until (Huruf = " ") Or (j > Batas_buffer)
              'StatusBar1.SimpleText = "Converting sentences into array of words at " & i & " round"
              Huruf = Mid$(Kalimat, j, 1)
              If Huruf = " " Then
              Else
                 Kata = Kata + Huruf
              End If
              j = j + 1
           Loop 'result is kata (character before spasi)
           Kumpulan_Kata(k) = Kata

           Huruf = ""
           Kata = ""
           k = k + 1
        Loop
        Batas_kumpulan_kata = k - 1

        'analyzing kalimat in term of kumpulan_kata array
        If (Kumpulan_Kata(1) = "T" And Batas_kumpulan_kata = 7) Then ' Or (Kumpulan_Kata(1) =
"I" And Batas_kumpulan_kata = 7) Or (Kumpulan_Kata(1) = "U" And Batas_kumpulan_kata = 6)
Then
           'to find out the address of outside party
           'StatusBar1.SimpleText = "Finding outsiders at " & i & " round"
           filter_result = Left$(Kumpulan_Kata(4), 7)
           If filter_result = Intranet Then
              Internal_Party = Kumpulan_Kata(4)   'data leaving intranet
              Outside_Party = Kumpulan_Kata(6)
           Else
              Outside_Party = Kumpulan_Kata(4)    'data coming from outside
              Internal_Party = Kumpulan_Kata(6)
           End If
           'to remove the port number and leave IP address only
           hasil_split() = Split(Outside_Party, ":")
           Outside_Party = hasil_split(0)
           hasil_split() = Split(Internal_Party, ":")
           Internal_Party = hasil_split(0)
           'List1.AddItem Outside_Party
           'Listing the outside party to an array
           filter_result = Left$(Outside_Party, 7)
           If (Outside_Party <> "") And (filter_result <> Intranet) Then 'avoid blank space or internal to
internal communication
              If batas_index_outsider = 0 Then 'list of outsider still empty
                 'List1.AddItem Outside_Party
                 batas_index_outsider = batas_index_outsider + 1 'because it's started from index 0
                 List_Outsider(batas_index_outsider) = Outside_Party
                 List_Internal(batas_index_outsider) = Internal_Party
                 List_AccessTime(batas_index_outsider) = Kumpulan_Kata(3)
                 batas_index_outsider = batas_index_outsider + 1
              Else
                 status_exist = False 'check whether outsider has included in the list
                 For m = 1 To (batas_index_outsider - 1)
                    If Outside_Party = List_Outsider(m) Then
                       status_exist = True
                       Exit For
                    End If
                 Next
```

```
            If status_exist = False Then 'it's a new outsider
               List_Outsider(batas_index_outsider) = Outside_Party
               List_Internal(batas_index_outsider) = Internal_Party
               List_AccessTime(batas_index_outsider) = Kumpulan_Kata(3)
               batas_index_outsider = batas_index_outsider + 1
               'List_Kalimat(batas_index_outsider) = Kalimat
            End If
         End If
      End If
   Else
      'StatusBar1.SimpleText = "Searching suspicious traffic at " & i & " round"
      If Outside_Party <> "" Then
         'checking the threat in the traffic buffer

         'EXE file
         Bahaya = InStr(1, Kalimat, Suspicious(1), vbTextCompare)
         If Bahaya > 0 Then 'if threat is discovered
            For m = 1 To (batas_index_outsider - 1)
               If Outside_Party = List_Outsider(m) Then
                  List_Bahaya1(m) = List_Bahaya1(m) + 1
                  Exit For
               End If
            Next
         End If
         Bahaya = InStr(1, Kalimat, Suspicious(2), vbTextCompare)
         If Bahaya > 0 Then 'if threat is discovered
            For m = 1 To (batas_index_outsider - 1)
               If Outside_Party = List_Outsider(m) Then
                  List_Bahaya1(m) = List_Bahaya1(m) + 1
                  Exit For
               End If
            Next
         End If
         Bahaya = InStr(1, Kalimat, Suspicious(7), vbTextCompare)
         If Bahaya > 0 Then 'if threat is discovered
            For m = 1 To (batas_index_outsider - 1)
               If Outside_Party = List_Outsider(m) Then
                  List_Bahaya1(m) = List_Bahaya1(m) + 1
                  Exit For
               End If
            Next
         End If
         Bahaya = InStr(1, Kalimat, Suspicious(8), vbTextCompare)
         If Bahaya > 0 Then 'if threat is discovered
            For m = 1 To (batas_index_outsider - 1)
               If Outside_Party = List_Outsider(m) Then
                  List_Bahaya1(m) = List_Bahaya1(m) + 1
                  Exit For
               End If
            Next
         End If

         'Active script
         Bahaya = InStr(1, Kalimat, Suspicious(3), vbTextCompare)
         If Bahaya > 0 Then 'if threat is discovered
            For m = 1 To (batas_index_outsider - 1)
               If Outside_Party = List_Outsider(m) Then
                  List_Bahaya2(m) = List_Bahaya2(m) + 1
                  Exit For
               End If
```

```
    Next
End If

'Cookies
Bahaya = InStr(1, Kalimat, Suspicious(4), vbTextCompare)
If Bahaya > 0 Then 'if threat is discovered
    For m = 1 To (batas_index_outsider - 1)
        If Outside_Party = List_Outsider(m) Then
            List_Bahaya3(m) = List_Bahaya3(m) + 1
            Exit For
        End If
    Next
End If
Bahaya = InStr(1, Kalimat, Suspicious(5), vbTextCompare)
If Bahaya > 0 Then 'if threat is discovered
    For m = 1 To (batas_index_outsider - 1)
        If Outside_Party = List_Outsider(m) Then
            List_Bahaya3(m) = List_Bahaya3(m) + 1
            Exit For
        End If
    Next
End If
Bahaya = InStr(1, Kalimat, Suspicious(6), vbTextCompare)
If Bahaya > 0 Then 'if threat is discovered
    For m = 1 To (batas_index_outsider - 1)
        If Outside_Party = List_Outsider(m) Then
            List_Bahaya3(m) = List_Bahaya3(m) + 1
            Exit For
        End If
    Next
End If

'Add
Bahaya = InStr(1, Kalimat, Suspicious(9), vbTextCompare)
If Bahaya > 0 Then 'if threat is discovered
    For m = 1 To (batas_index_outsider - 1)
        If Outside_Party = List_Outsider(m) Then
            List_Bahaya5(m) = List_Bahaya5(m) + 1
            Exit For
        End If
    Next
End If
Bahaya = InStr(1, Kalimat, Suspicious(10), vbTextCompare)
If Bahaya > 0 Then 'if threat is discovered
    For m = 1 To (batas_index_outsider - 1)
        If Outside_Party = List_Outsider(m) Then
            List_Bahaya5(m) = List_Bahaya5(m) + 1
            Exit For
        End If
    Next
End If

'Forced Information
Bahaya = InStr(1, Kalimat, Suspicious(11), vbTextCompare)
If Bahaya > 0 Then 'if threat is discovered
    For m = 1 To (batas_index_outsider - 1)
        If Outside_Party = List_Outsider(m) Then
            List_Bahaya6(m) = List_Bahaya6(m) + 1
            Exit For
        End If
```

```
            Next
         End If
         Bahaya = InStr(1, Kalimat, Suspicious(12), vbTextCompare)
         If Bahaya > 0 Then 'if threat is discovered
            For m = 1 To (batas_index_outsider - 1)
               If Outside_Party = List_Outsider(m) Then
                  List_Bahaya6(m) = List_Bahaya6(m) + 1
                  Exit For
               End If
            Next
         End If
         Bahaya = InStr(1, Kalimat, Suspicious(13), vbTextCompare)
         If Bahaya > 0 Then 'if threat is discovered
            For m = 1 To (batas_index_outsider - 1)
               If Outside_Party = List_Outsider(m) Then
                  List_Bahaya6(m) = List_Bahaya6(m) + 1
                  Exit For
               End If
            Next
         End If
         Bahaya = InStr(1, Kalimat, Suspicious(14), vbTextCompare)
         If Bahaya > 0 Then 'if threat is discovered
            For m = 1 To (batas_index_outsider - 1)
               If Outside_Party = List_Outsider(m) Then
                  List_Bahaya6(m) = List_Bahaya6(m) + 1
                  Exit For
               End If
            Next
         End If
         Bahaya = InStr(1, Kalimat, Suspicious(15), vbTextCompare)
         If Bahaya > 0 Then 'if threat is discovered
            For m = 1 To (batas_index_outsider - 1)
               If Outside_Party = List_Outsider(m) Then
                  List_Bahaya6(m) = List_Bahaya6(m) + 1
                  Exit For
               End If
            Next
         End If

      End If
   End If

   'Text1.Text = Kalimat
   Huruf = ""
   Kata = ""
   Kalimat = ""
Loop 'end of loop for analyzing buffer file

Total_Outsider = batas_index_outsider - 1
Total_Bahaya1 = 0
Total_Bahaya2 = 0
Total_Bahaya3 = 0
Total_Bahaya5 = 0
Total_Bahaya6 = 0
For m = 1 To (batas_index_outsider - 1)
   Total_Bahaya1 = Total_Bahaya1 + List_Bahaya1(m)
   Total_Bahaya2 = Total_Bahaya2 + List_Bahaya2(m)
   Total_Bahaya3 = Total_Bahaya3 + List_Bahaya3(m)
   Total_Bahaya5 = Total_Bahaya5 + List_Bahaya5(m)
   Total_Bahaya6 = Total_Bahaya6 + List_Bahaya6(m)
```

```
Next


'Writing the report to a text file
'Report_File = "D:\Project\Traffic Analyzer\Report.txt"
'WriteTextFileContents ("Buffer name: " + Nama_Traffic_Buffer + ":"), Report_File, False
'For m = 1 To ((batas_index_outsider - 1) * 7) + 5
'    WriteTextFileContents (List1.List(m - 1)), Report_File, True
'Next


'COMPUTING THE TABLE OF SECURITY
'==============================

Table1 = "D:\Project\Fuzzy-based\Table\fuzzy_table1"
Table2 = "D:\Project\Fuzzy-based\Table\fuzzy_table2"
Table3 = "D:\Project\Fuzzy-based\Table\fuzzy_table3"
Table1_Buffer = ReadTextFileContents(Table1)
Table2_Buffer = ReadTextFileContents(Table2)
Table3_Buffer = ReadTextFileContents(Table3)

'creating the array of Table1
Batas_Table1 = 0
i = 1
Do Until i >= Len(Table1_Buffer)
   StatusBar1.SimpleText = "Browsing Table1 at " & i & " round"
   'StatusBar1.Refresh
   Huruf = ""
   Kalimat = ""
   Do Until (Huruf = Chr(10)) Or (i >= Len(Table1_Buffer))
      Huruf = Mid$(Table1_Buffer, i, 1)
      If (Huruf = Chr(13)) Or (Huruf = " ") Or (Huruf = Chr(10)) Then
      Else
         Kalimat = Kalimat + Huruf
      End If
      i = i + 1
   Loop 'result is kalimat (character before enter)

   If Kalimat <> "" Then
      Batas_Table1 = Batas_Table1 + 1
      Table1_array(Batas_Table1) = Kalimat
   End If
Loop       'result is Table1_array

'creating the array of Table2
'List2.AddItem "TABLE2"
'List2.AddItem Len(Table2_Buffer)
Batas_Table2 = 0
i = 1
Do Until i >= Len(Table2_Buffer)
   StatusBar1.SimpleText = "Browsing Table2 at " & i & " round"
   'StatusBar1.Refresh
   Huruf = ""
   Kalimat = ""
   Do Until (Huruf = Chr(10)) Or (i >= Len(Table2_Buffer))
      Huruf = Mid$(Table2_Buffer, i, 1)
      If (Huruf = Chr(13)) Or (Huruf = " ") Or (Huruf = Chr(10)) Then
      Else
         Kalimat = Kalimat + Huruf
      End If
```

```
        i = i + 1
    Loop 'result is kalimat (character before enter)

    If Kalimat <> "" Then
        Batas_Table2 = Batas_Table2 + 1
        Table2_array(Batas_Table2) = Kalimat
    End If
Loop        'result is Table2_array

'creating the array of Table3
'List2.AddItem "TABLE3"
'List2.AddItem Len(Table3_Buffer)
Batas_Table3 = 0
i = 1
Do Until i >= Len(Table3_Buffer)
    StatusBar1.SimpleText = "Browsing Table3 at " & i & " round"
    'StatusBar1.Refresh
    Huruf = ""
    Kalimat = ""
    Do Until (Huruf = Chr(10)) Or (i >= Len(Table3_Buffer))
        Huruf = Mid$(Table3_Buffer, i, 1)
        If (Huruf = Chr(13)) Or (Huruf = " ") Or (Huruf = Chr(10)) Then
        Else
            Kalimat = Kalimat + Huruf
        End If
        i = i + 1
    Loop 'result is kalimat (character before enter)

    If Kalimat <> "" Then
        Batas_Table3 = Batas_Table3 + 1
        Table3_array(Batas_Table3) = Kalimat
    End If
Loop        'result is Table3_array

'Browsing array of outsider created from the buffer
'in order to get any pattern of threat
For i = 1 To Total_Outsider
    'status_Threat = False

    'Is there any effort to force user to read too much information
    'by flooding user with adds from many external servers?
    '-----------------------------------------------------------
    'If (status_Threat = False) Then
    Outside_Party = List_Outsider(i)
    Internal_Party = List_Internal(i)
    Waktu_access1 = List_AccessTime(i)
    Forced_Information = 0
    List_Bahaya4(i) = 0
    For j = 1 To Total_Outsider
        If (Outside_Party <> List_Outsider(j)) And (Internal_Party = List_Internal(j)) Then
            Waktu_access2 = List_AccessTime(j)
            BedaWaktu = DeltaTime(Waktu_access1, Waktu_access2)
            If BedaWaktu < 60 Then  'identifying the servers from the same site
                Forced_Information = Forced_Information + 1
                List_Bahaya4(i) = Forced_Information
            End If
        End If
    Next

    Risk_Bahaya1 = 0
```

```
Risk_Bahaya2 = 0
Risk_Bahaya3 = 0
Risk_Bahaya4 = 0
Risk_Bahaya5 = 0
Risk_Bahaya6 = 0

'EXE file
If List_Bahaya1(i) > 0 Then
   Risk_Bahaya1 = 1
Else
   Risk_Bahaya1 = 0
End If

'Number of active script -> not significant to detect the threat
'example: elsevier has 43 scripts, ieee has 15
'If List_Bahaya2(i) > 0 Then
'    'Risk_Bahaya2 = 1 / (1 + (1 / List_Bahaya2(i)) ^ (1/1.5))
'    Risk_Bahaya2 = 1 - (1 / List_Bahaya2(i) ^ (1 / 1.5))
'End If

'Number of Cookies -> less influence to the threat
'example: cisco has 8 cookies, psz and science direct have 4 cookies
If List_Bahaya3(i) > 0 Then
   'Risk_Bahaya3 = 1 / (1 + (1 / List_Bahaya3(i)))
   Risk_Bahaya3 = 1 - (1 / List_Bahaya3(i) ^ (1 / 2))
End If

'Number of external machines -> less influence to the threat
'example: ieee has 7 machines
If List_Bahaya4(i) > 0 Then
   'Risk_Bahaya4 = 1 / (1 + (1 / List_Bahaya4(i)))
   Risk_Bahaya4 = 1 - (1 / List_Bahaya4(i) ^ (1 / 2))
End If

'Add -> slightly influence the threat
If List_Bahaya5(i) > 0 Then
   Risk_Bahaya5 = 1 - (1 / List_Bahaya5(i) ^ (1.7))
End If

'Number of Forced Info -> extremely influence the threat
If List_Bahaya6(i) > 0 Then
   Risk_Bahaya6 = 1 - (1 / List_Bahaya6(i) ^ (3))
End If


'Computing total risk
If (Risk_Bahaya4 - Risk_Bahaya1) >= 0 Then
   Total_Risk = Risk_Bahaya4
Else
   Total_Risk = Risk_Bahaya1
End If
If (Total_Risk - Risk_Bahaya2) < 0 Then
   Total_Risk = Risk_Bahaya2
End If
If (Total_Risk - Risk_Bahaya3) < 0 Then
   Total_Risk = Risk_Bahaya3
End If
If (Total_Risk - Risk_Bahaya5) < 0 Then
   Total_Risk = Risk_Bahaya5
End If
```

```
If (Total_Risk - Risk_Bahaya6) < 0 Then
   Total_Risk = Risk_Bahaya6
End If
List_Risk(i) = Total_Risk

Select Case List_Risk(i)
   Case Is < 0.25
      List_Table(i) = "Table1"
      'Table1 Registration
      status_exist = False
      For j = 1 To Batas_Table1
         If Table1_array(j) = List_Outsider(i) Then
            status_exist = True
         End If
      Next
      If status_exist = False Then
         Batas_Table1 = Batas_Table1 + 1
         Table1_array(Batas_Table1) = List_Outsider(i)
      End If

      'delete the membership from other table
      k = 0
      For j = 1 To Batas_Table2       'Table2
         If Table2_array(j) = List_Outsider(i) Then
         Else
            k = k + 1
            Temporary_array(k) = Table2_array(j)
         End If
      Next
      Batas_Table2 = k
      For j = 1 To Batas_Table2
         Table2_array(j) = Temporary_array(j)
      Next

      k = 0
      For j = 1 To Batas_Table3       'Table3
         If Table3_array(j) = List_Outsider(i) Then
         Else
            k = k + 1
            Temporary_array(k) = Table3_array(j)
         End If
      Next
      Batas_Table3 = k
      For j = 1 To Batas_Table3
         Table3_array(j) = Temporary_array(j)
      Next

   Case 0.25 To 0.75
      List_Table(i) = "Table2"
      'Table2 Registration
      status_exist = False
      For j = 1 To Batas_Table2
         If Table2_array(j) = List_Outsider(i) Then
            status_exist = True
         End If
      Next
      If status_exist = False Then
         Batas_Table2 = Batas_Table2 + 1
         Table2_array(Batas_Table2) = List_Outsider(i)
      End If
```

```
    'delete the membership from other table
    k = 0
    For j = 1 To Batas_Table1       'Table1
       If Table1_array(j) = List_Outsider(i) Then
       Else
          k = k + 1
          Temporary_array(k) = Table1_array(j)
       End If
    Next
    Batas_Table1 = k
    For j = 1 To Batas_Table1
       Table1_array(j) = Temporary_array(j)
    Next

    k = 0
    For j = 1 To Batas_Table3       'Table3
       If Table3_array(j) = List_Outsider(i) Then
       Else
          k = k + 1
          Temporary_array(k) = Table3_array(j)
       End If
    Next
    Batas_Table3 = k
    For j = 1 To Batas_Table3
       Table3_array(j) = Temporary_array(j)
    Next

Case Is > 0.75
    List_Table(i) = "Table3"
    'Table3 Registration
    status_exist = False
    For j = 1 To Batas_Table3
       If Table3_array(j) = List_Outsider(i) Then
          status_exist = True
       End If
    Next
    If status_exist = False Then
       Batas_Table3 = Batas_Table3 + 1
       Table3_array(Batas_Table3) = List_Outsider(i)
    End If

    'delete the membership from other table
    k = 0
    For j = 1 To Batas_Table1       'Table1
       If Table1_array(j) = List_Outsider(i) Then
       Else
          k = k + 1
          Temporary_array(k) = Table1_array(j)
       End If
    Next
    Batas_Table1 = k
    For j = 1 To Batas_Table1
       Table1_array(j) = Temporary_array(j)
    Next

    k = 0
    For j = 1 To Batas_Table2       'Table2
       If Table2_array(j) = List_Outsider(i) Then
       Else
```

```
            k = k + 1
            Temporary_array(k) = Table2_array(j)
          End If
        Next
        Batas_Table2 = k
        For j = 1 To Batas_Table2
          Table2_array(j) = Temporary_array(j)
        Next

    End Select
Next

'Send the result to Table files
Report_File = "D:\Project\Fuzzy-based\Table\fuzzy_table1"
WriteTextFileContents (""), Report_File, False
For m = 1 To Batas_Table1
    WriteTextFileContents (Table1_array(m) + " "), Report_File, True
Next
Report_File = "D:\Project\Fuzzy-based\Table\fuzzy_table2"
WriteTextFileContents (""), Report_File, False
For m = 1 To Batas_Table2
    WriteTextFileContents (Table2_array(m) + " "), Report_File, True
Next
Report_File = "D:\Project\Fuzzy-based\Table\fuzzy_table3"
WriteTextFileContents (""), Report_File, False
For m = 1 To Batas_Table3
    WriteTextFileContents (Table3_array(m) + " "), Report_File, True
Next

'Send the results to firewall
FileCopy "D:\Project\Fuzzy-based\Table\fuzzy_table1", "Z:\fuzzy_table1"
FileCopy "D:\Project\Fuzzy-based\Table\fuzzy_table2", "Z:\fuzzy_table2"
FileCopy "D:\Project\Fuzzy-based\Table\fuzzy_table3", "Z:\fuzzy_table3"

'Presenting the result
List1.AddItem "Summary:"
List1.AddItem "Number of external server machine: " & Total_Outsider
List1.AddItem "Number of EXE file: " & Total_Bahaya1
List1.AddItem "Number of active script: " & Total_Bahaya2
List1.AddItem "Number of cookies: " & Total_Bahaya3
List1.AddItem "Number of add: " & Total_Bahaya5
List1.AddItem "Number of forced info: " & Total_Bahaya6

For m = 1 To (batas_index_outsider - 1)
    List1.AddItem " "
    List1.AddItem List_Outsider(m)
    List1.AddItem "Internal user          " & List_Internal(m)
    List1.AddItem "First access           " & List_AccessTime(m)
    'List1.AddItem List_Kalimat(m)
    List1.AddItem "Risk Value             " & List_Risk(m)
    List1.AddItem List_Table(m)
    List1.AddItem "EXE Files              " & List_Bahaya1(m)
    List1.AddItem "Other involved machines  " & List_Bahaya4(m)
    List1.AddItem "Active Scripts         " & List_Bahaya2(m)
    List1.AddItem "Cookies                " & List_Bahaya3(m)
    List1.AddItem "Adds                   " & List_Bahaya5(m)
    List1.AddItem "Forced Info            " & List_Bahaya6(m)
Next

'Writing the report to a text file
```

```
    'Report_File = "D:\Project\Fuzzy Traffic Analyzer\Report\" + File1.filename
    Report_File = "D:\Project\Fuzzy-based\Table\fuzzy_report"
    WriteTextFileContents ("Buffer name: " + Nama_Traffic_Buffer), Report_File, False
    For m = 1 To ((batas_index_outsider - 1) * 12) + 7
        WriteTextFileContents (List1.List(m - 1)), Report_File, True
    Next

    'testing only
    List2.Clear
    List2.AddItem "TABLE1"
    For i = 1 To Batas_Table1
        List2.AddItem Table1_array(i)
    Next
    List2.AddItem " "
    List2.AddItem "TABLE2"
    For i = 1 To Batas_Table2
        List2.AddItem Table2_array(i)
    Next
    List2.AddItem " "
    List2.AddItem "TABLE3"
    For i = 1 To Batas_Table3
        List2.AddItem Table3_array(i)
    Next

    'closing the process
    Erase Kumpulan_Kata
    Erase List_Outsider
    'Erase List_Kalimat
    Erase List_Bahaya1
    Erase List_Bahaya2
    Erase List_Bahaya3
    Erase List_Bahaya4
    Erase List_Bahaya5
    Erase List_Bahaya6
    Erase List_Risk
    Erase List_Table
    Erase Suspicious
    Erase Table1_array
    Erase Table2_array
    Erase Table3_array
    Erase Temporary_array

    Text2.Text = "End at " & Now

    ' End reporting
    Report_File = "D:\Project\Fuzzy-based\Table\fuzzy_report"
    WriteTextFileContents (" "), Report_File, True
    WriteTextFileContents (Text1.Text), Report_File, True
    WriteTextFileContents (Text2.Text), Report_File, True
    WriteTextFileContents ("File size " & FileLen(Nama_Traffic_Buffer) & " Bytes"), Report_File,
True

End Sub
```

APPENDIX B
The Implementation of Adaptively Updating Security Rules using Fuzzy Reasoning

```
Function DeltaTime(Waktu1 As String, Waktu2 As String) As Double
   Dim detik1, detik2 As Double
   Dim hasil_split() As String
   hasil_split() = Split(Waktu1, ":")
   detik1 = CDbl(hasil_split(0)) * 3600 + CDbl(hasil_split(0)) * 60 + CDbl(hasil_split(2))
   hasil_split() = Split(Waktu2, ":")
   detik2 = CDbl(hasil_split(0)) * 3600 + CDbl(hasil_split(0)) * 60 + CDbl(hasil_split(2))
   DeltaTime = Abs(detik1 - detik2)
End Function

Function ReadTextFileContents(filename As String) As String
   Dim fnum As Integer, isOpen As Boolean
   On Error GoTo Error_Handler
   ' Get the next free file number.
   fnum = FreeFile()
   Open filename For Input As #fnum
   ' If execution flow got here, the file has been open without error.
   isOpen = True
   ' Read the entire contents in one single operation.
   ReadTextFileContents = Input(LOF(fnum), fnum)
   ' Intentionally flow into the error handler to close the file.
Error_Handler:
   ' Raise the error (if any), but first close the file.
   If isOpen Then Close #fnum
   'If Err Then Err.Raise Err.Number, , Err.Description
End Function
Sub WriteTextFileContents(Text As String, filename As String, Optional AppendMode As Boolean)
   Dim fnum As Integer, isOpen As Boolean
   On Error GoTo Error_Handler
   ' Get the next free file number.
   fnum = FreeFile()
   If AppendMode Then
      Open filename For Append As #fnum
    Else
      Open filename For Output As #fnum
    End If
    ' If execution flow gets here, the file has been opened correctly.
    isOpen = True
    ' Print to the file in one single operation.
    Print #fnum, Text
    ' Intentionally flow into the error handler to close the file.
Error_Handler:
   ' Raise the error (if any), but first close the file.
   If isOpen Then Close #fnum
   'If Err Then Err.Raise Err.Number, , Err.Description
End Sub

Private Sub End_Click()

End Sub

Private Sub EndAnalysis_Click()
   Timer1.Enabled = False
   Run.Enabled = True
   EndAnalysis.Enabled = False
End Sub
```

```
Private Sub Form_Load()
   Timer1.Enabled = False
   EndAnalysis.Enabled = False
End Sub

Private Sub Run_Click()
   EndAnalysis.Enabled = True
   Run.Enabled = False
   Timer1.Enabled = True
   Timer1.Interval = 3000
End Sub

Private Sub Timer1_Timer()
   Dim Nama_Traffic_Buffer As String
   Dim Kumpulan_Kata(1 To 1000000) As String
   Dim List_Outsider(1 To 1000) As String
   Dim List_Internal(1 To 1000) As String
   Dim List_AccessTime(1 To 1000) As String
   'Dim List_Kalimat(1 To 1000) As String
   Dim List_Bahaya1(1 To 1000) As Integer
   Dim List_Bahaya2(1 To 1000) As Integer
   Dim List_Bahaya3(1 To 1000) As Integer
   Dim List_Bahaya4(1 To 1000) As Integer
   Dim List_Bahaya5(1 To 1000) As Integer
   Dim List_Bahaya6(1 To 1000) As Integer
   Dim List_Risk(1 To 1000) As Double
   Dim List_Table(1 To 1000) As String
   Dim Table1_array(1 To 100) As String
   Dim Table2_array(1 To 100) As String
   Dim Table3_array(1 To 100) As String
   Dim Suspicious(1 To 100) As String
   Dim Temporary_array(1 To 100) As String
   Dim Traffic_Buffer, Huruf, Kalimat, Kata As String
   Dim Table1_Buffer, Table2_Buffer, Table3_Buffer As String
   Dim Batas_Table1, Batas_Table2, Batas_Table3 As Long
   Dim fnum1 As Integer
   Dim i, j, k, l, m, Batas_buffer, Batas_kalimat, Batas_kumpulan_kata As Long
   Dim filter_result As String
   Dim hasil_split() As String
   Dim Intranet As String, Outside_Party As String, Internal_Party As String
   Dim Bahaya, batas_index_outsider As Long
   Dim status_exist As Boolean
   Dim status_Threat As Boolean
   Dim Report_File As String
   Dim Table1 As String
   Dim Table2 As String
   Dim Table3 As String
   Dim Total_Outsider, Total_Bahaya1, Total_Bahaya2, Total_Bahaya3, Total_Bahaya5,
Total_Bahaya6 As Long
   Dim Waktu_access1 As String, Waktu_access2 As String
   Dim BedaWaktu As Double
   Dim Forced_Information As Integer
   Dim Risk_Bahaya1, Risk_Bahaya2, Risk_Bahaya3, Risk_Bahaya4, Risk_Bahaya5, Risk_Bahaya6,
Total_Risk As Double

   'If Text3.Text = "" Then
   '    Exit Sub
   'End If
```

```
Text1.Text = "Start at " & Now
Text2.Text = "In progress ... "
Text1.Refresh
Text2.Refresh
List1.Clear
List1.Refresh


'ANALYZING TRAFFIC BUFFER
'========================

Intranet = "24.4.74"
'Suspicious(1) = "script src"
'Suspicious(2) = "Set-Cookie:"
'Suspicious(3) = "<script language="
'Suspicious(2) = ".js"""
'Suspicious(3) = ".js>"
'Suspicious(2) = "<script language ="
'Suspicious(3) = "<script language="

Suspicious(1) = ".exe"""           'EXE
Suspicious(2) = ".exe>"            'EXE
Suspicious(3) = "</script>"        'active script
Suspicious(4) = "Set-Cookie"       'cookie
Suspicious(5) = "Set Cookie"       'cookie
Suspicious(6) = "SetCookie"        'cookie
Suspicious(7) = ".exe "            'EXE
Suspicious(8) = ".exe|"            'EXE
Suspicious(9) = "window.open"      'advertisement
Suspicious(10) = "popup("          'advertisement
Suspicious(11) = "if (exit)"       'forced info
Suspicious(12) = "exit=false"      'forced info
Suspicious(13) = "exit=true"       'forced info
Suspicious(14) = "checkexit"       'forced info
Suspicious(15) = "ControlExit"     'forced info

'Nama_Traffic_Buffer = "D:\Project\Traffic Analyzer\Data\" + Text3.Text
'Nama_Traffic_Buffer = "D:\Project\Data\Threat\" + Text3.Text
Nama_Traffic_Buffer = "D:\Project\Ngrep\fuzzy_buffer"
Traffic_Buffer = ReadTextFileContents(Nama_Traffic_Buffer) 'Copy is traffic buffer
WriteTextFileContents (""), Nama_Traffic_Buffer, False 'Reset traffic buffer
Batas_buffer = Len(Traffic_Buffer)

'Start analyzing buffer file
Outside_Party = ""
batas_index_outsider = 0
i = 1
Do Until i >= Batas_buffer
   StatusBar1.SimpleText = "Browsing buffer at " & i & " round"
   'StatusBar1.Refresh
   Do Until (Huruf = Chr(10)) Or (i >= Batas_buffer)
      Huruf = Mid$(Traffic_Buffer, i, 1)
      If (Huruf = Chr(13)) Or (Huruf = "#") Or (Huruf = Chr(10)) Then
      Else
         Kalimat = Kalimat + Huruf
      End If
      i = i + 1
   Loop 'result is kalimat (character before enter)

   'Start analyzing sentence (character before enter)
```

```
        Batas_kalimat = Len(Kalimat)
        j = 1
        k = 1
        Do Until j > Batas_kalimat
           Do Until (Huruf = " ") Or (j > Batas_buffer)
              'StatusBar1.SimpleText = "Converting sentences into array of words at " & i & " round"
              Huruf = Mid$(Kalimat, j, 1)
              If Huruf = " " Then
              Else
                 Kata = Kata + Huruf
              End If
              j = j + 1
           Loop 'result is kata (character before spasi)
           Kumpulan_Kata(k) = Kata

           Huruf = ""
           Kata = ""
           k = k + 1
        Loop
        Batas_kumpulan_kata = k - 1

        'analyzing kalimat in term of kumpulan_kata array
        If (Kumpulan_Kata(1) = "T" And Batas_kumpulan_kata = 7) Then ' Or (Kumpulan_Kata(1) =
"I" And Batas_kumpulan_kata = 7) Or (Kumpulan_Kata(1) = "U" And Batas_kumpulan_kata = 6)
Then
           'to find out the address of outside party
           'StatusBar1.SimpleText = "Finding outsiders at " & i & " round"
           filter_result = Left$(Kumpulan_Kata(4), 7)
           If filter_result = Intranet Then
              Internal_Party = Kumpulan_Kata(4)   'data leaving intranet
              Outside_Party = Kumpulan_Kata(6)
           Else
              Outside_Party = Kumpulan_Kata(4)    'data coming from outside
              Internal_Party = Kumpulan_Kata(6)
           End If
           'to remove the port number and leave IP address only
           hasil_split() = Split(Outside_Party, ":")
           Outside_Party = hasil_split(0)
           hasil_split() = Split(Internal_Party, ":")
           Internal_Party = hasil_split(0)
           'List1.AddItem Outside_Party
           'Listing the outside party to an array
           filter_result = Left$(Outside_Party, 7)
           If (Outside_Party <> "") And (filter_result <> Intranet) Then 'avoid blank space or internal to
internal communication
              If batas_index_outsider = 0 Then 'list of outsider still empty
                 'List1.AddItem Outside_Party
                 batas_index_outsider = batas_index_outsider + 1 'because it's started from index 0
                 List_Outsider(batas_index_outsider) = Outside_Party
                 List_Internal(batas_index_outsider) = Internal_Party
                 List_AccessTime(batas_index_outsider) = Kumpulan_Kata(3)
                 batas_index_outsider = batas_index_outsider + 1
              Else
                 status_exist = False 'check whether outsider has included in the list
                 For m = 1 To (batas_index_outsider - 1)
                    If Outside_Party = List_Outsider(m) Then
                       status_exist = True
                       Exit For
                    End If
                 Next
```

```
            If status_exist = False Then 'it's a new outsider
               List_Outsider(batas_index_outsider) = Outside_Party
               List_Internal(batas_index_outsider) = Internal_Party
               List_AccessTime(batas_index_outsider) = Kumpulan_Kata(3)
               batas_index_outsider = batas_index_outsider + 1
               'List_Kalimat(batas_index_outsider) = Kalimat
            End If
         End If
      End If
   Else
      'StatusBar1.SimpleText = "Searching suspicious traffic at " & i & " round"
      If Outside_Party <> "" Then
         'checking the threat in the traffic buffer

         'EXE file
         Bahaya = InStr(1, Kalimat, Suspicious(1), vbTextCompare)
         If Bahaya > 0 Then 'if threat is discovered
            For m = 1 To (batas_index_outsider - 1)
               If Outside_Party = List_Outsider(m) Then
                  List_Bahaya1(m) = List_Bahaya1(m) + 1
                  Exit For
               End If
            Next
         End If
         Bahaya = InStr(1, Kalimat, Suspicious(2), vbTextCompare)
         If Bahaya > 0 Then 'if threat is discovered
            For m = 1 To (batas_index_outsider - 1)
               If Outside_Party = List_Outsider(m) Then
                  List_Bahaya1(m) = List_Bahaya1(m) + 1
                  Exit For
               End If
            Next
         End If
         Bahaya = InStr(1, Kalimat, Suspicious(7), vbTextCompare)
         If Bahaya > 0 Then 'if threat is discovered
            For m = 1 To (batas_index_outsider - 1)
               If Outside_Party = List_Outsider(m) Then
                  List_Bahaya1(m) = List_Bahaya1(m) + 1
                  Exit For
               End If
            Next
         End If
         Bahaya = InStr(1, Kalimat, Suspicious(8), vbTextCompare)
         If Bahaya > 0 Then 'if threat is discovered
            For m = 1 To (batas_index_outsider - 1)
               If Outside_Party = List_Outsider(m) Then
                  List_Bahaya1(m) = List_Bahaya1(m) + 1
                  Exit For
               End If
            Next
         End If

         'Active script
         Bahaya = InStr(1, Kalimat, Suspicious(3), vbTextCompare)
         If Bahaya > 0 Then 'if threat is discovered
            For m = 1 To (batas_index_outsider - 1)
               If Outside_Party = List_Outsider(m) Then
                  List_Bahaya2(m) = List_Bahaya2(m) + 1
                  Exit For
               End If
```

```
    Next
End If

'Cookies
Bahaya = InStr(1, Kalimat, Suspicious(4), vbTextCompare)
If Bahaya > 0 Then 'if threat is discovered
    For m = 1 To (batas_index_outsider - 1)
        If Outside_Party = List_Outsider(m) Then
            List_Bahaya3(m) = List_Bahaya3(m) + 1
            Exit For
        End If
    Next
End If
Bahaya = InStr(1, Kalimat, Suspicious(5), vbTextCompare)
If Bahaya > 0 Then 'if threat is discovered
    For m = 1 To (batas_index_outsider - 1)
        If Outside_Party = List_Outsider(m) Then
            List_Bahaya3(m) = List_Bahaya3(m) + 1
            Exit For
        End If
    Next
End If
Bahaya = InStr(1, Kalimat, Suspicious(6), vbTextCompare)
If Bahaya > 0 Then 'if threat is discovered
    For m = 1 To (batas_index_outsider - 1)
        If Outside_Party = List_Outsider(m) Then
            List_Bahaya3(m) = List_Bahaya3(m) + 1
            Exit For
        End If
    Next
End If

'Add
Bahaya = InStr(1, Kalimat, Suspicious(9), vbTextCompare)
If Bahaya > 0 Then 'if threat is discovered
    For m = 1 To (batas_index_outsider - 1)
        If Outside_Party = List_Outsider(m) Then
            List_Bahaya5(m) = List_Bahaya5(m) + 1
            Exit For
        End If
    Next
End If
Bahaya = InStr(1, Kalimat, Suspicious(10), vbTextCompare)
If Bahaya > 0 Then 'if threat is discovered
    For m = 1 To (batas_index_outsider - 1)
        If Outside_Party = List_Outsider(m) Then
            List_Bahaya5(m) = List_Bahaya5(m) + 1
            Exit For
        End If
    Next
End If

'Forced Information
Bahaya = InStr(1, Kalimat, Suspicious(11), vbTextCompare)
If Bahaya > 0 Then 'if threat is discovered
    For m = 1 To (batas_index_outsider - 1)
        If Outside_Party = List_Outsider(m) Then
            List_Bahaya6(m) = List_Bahaya6(m) + 1
            Exit For
        End If
```

```
            Next
         End If
         Bahaya = InStr(1, Kalimat, Suspicious(12), vbTextCompare)
         If Bahaya > 0 Then 'if threat is discovered
            For m = 1 To (batas_index_outsider - 1)
               If Outside_Party = List_Outsider(m) Then
                  List_Bahaya6(m) = List_Bahaya6(m) + 1
                  Exit For
               End If
            Next
         End If
         Bahaya = InStr(1, Kalimat, Suspicious(13), vbTextCompare)
         If Bahaya > 0 Then 'if threat is discovered
            For m = 1 To (batas_index_outsider - 1)
               If Outside_Party = List_Outsider(m) Then
                  List_Bahaya6(m) = List_Bahaya6(m) + 1
                  Exit For
               End If
            Next
         End If
         Bahaya = InStr(1, Kalimat, Suspicious(14), vbTextCompare)
         If Bahaya > 0 Then 'if threat is discovered
            For m = 1 To (batas_index_outsider - 1)
               If Outside_Party = List_Outsider(m) Then
                  List_Bahaya6(m) = List_Bahaya6(m) + 1
                  Exit For
               End If
            Next
         End If
         Bahaya = InStr(1, Kalimat, Suspicious(15), vbTextCompare)
         If Bahaya > 0 Then 'if threat is discovered
            For m = 1 To (batas_index_outsider - 1)
               If Outside_Party = List_Outsider(m) Then
                  List_Bahaya6(m) = List_Bahaya6(m) + 1
                  Exit For
               End If
            Next
         End If

      End If
   End If

   'Text1.Text = Kalimat
   Huruf = ""
   Kata = ""
   Kalimat = ""
Loop 'end of loop for analyzing buffer file

Total_Outsider = batas_index_outsider - 1
Total_Bahaya1 = 0
Total_Bahaya2 = 0
Total_Bahaya3 = 0
Total_Bahaya5 = 0
Total_Bahaya6 = 0
For m = 1 To (batas_index_outsider - 1)
   Total_Bahaya1 = Total_Bahaya1 + List_Bahaya1(m)
   Total_Bahaya2 = Total_Bahaya2 + List_Bahaya2(m)
   Total_Bahaya3 = Total_Bahaya3 + List_Bahaya3(m)
   Total_Bahaya5 = Total_Bahaya5 + List_Bahaya5(m)
   Total_Bahaya6 = Total_Bahaya6 + List_Bahaya6(m)
```

```
Next


'Writing the report to a text file
'Report_File = "D:\Project\Traffic Analyzer\Report.txt"
'WriteTextFileContents ("Buffer name: " + Nama_Traffic_Buffer + ":"), Report_File, False
'For m = 1 To ((batas_index_outsider - 1) * 7) + 5
'    WriteTextFileContents (List1.List(m - 1)), Report_File, True
'Next


'COMPUTING THE TABLE OF SECURITY
'==============================

Table1 = "D:\Project\Fuzzy-based\Table\fuzzy_table1"
Table2 = "D:\Project\Fuzzy-based\Table\fuzzy_table2"
Table3 = "D:\Project\Fuzzy-based\Table\fuzzy_table3"
Table1_Buffer = ReadTextFileContents(Table1)
Table2_Buffer = ReadTextFileContents(Table2)
Table3_Buffer = ReadTextFileContents(Table3)

'creating the array of Table1
Batas_Table1 = 0
i = 1
Do Until i >= Len(Table1_Buffer)
   StatusBar1.SimpleText = "Browsing Table1 at " & i & " round"
   'StatusBar1.Refresh
   Huruf = ""
   Kalimat = ""
   Do Until (Huruf = Chr(10)) Or (i >= Len(Table1_Buffer))
      Huruf = Mid$(Table1_Buffer, i, 1)
      If (Huruf = Chr(13)) Or (Huruf = " ") Or (Huruf = Chr(10)) Then
      Else
         Kalimat = Kalimat + Huruf
      End If
      i = i + 1
   Loop 'result is kalimat (character before enter)

   If Kalimat <> "" Then
      Batas_Table1 = Batas_Table1 + 1
      Table1_array(Batas_Table1) = Kalimat
   End If
Loop       'result is Table1_array

'creating the array of Table2
'List2.AddItem "TABLE2"
'List2.AddItem Len(Table2_Buffer)
Batas_Table2 = 0
i = 1
Do Until i >= Len(Table2_Buffer)
   StatusBar1.SimpleText = "Browsing Table2 at " & i & " round"
   'StatusBar1.Refresh
   Huruf = ""
   Kalimat = ""
   Do Until (Huruf = Chr(10)) Or (i >= Len(Table2_Buffer))
      Huruf = Mid$(Table2_Buffer, i, 1)
      If (Huruf = Chr(13)) Or (Huruf = " ") Or (Huruf = Chr(10)) Then
      Else
         Kalimat = Kalimat + Huruf
      End If
```

```
        i = i + 1
    Loop 'result is kalimat (character before enter)

    If Kalimat <> "" Then
        Batas_Table2 = Batas_Table2 + 1
        Table2_array(Batas_Table2) = Kalimat
    End If
Loop        'result is Table2_array

'creating the array of Table3
'List2.AddItem "TABLE3"
'List2.AddItem Len(Table3_Buffer)
Batas_Table3 = 0
i = 1
Do Until i >= Len(Table3_Buffer)
    StatusBar1.SimpleText = "Browsing Table3 at " & i & " round"
    'StatusBar1.Refresh
    Huruf = ""
    Kalimat = ""
    Do Until (Huruf = Chr(10)) Or (i >= Len(Table3_Buffer))
        Huruf = Mid$(Table3_Buffer, i, 1)
        If (Huruf = Chr(13)) Or (Huruf = " ") Or (Huruf = Chr(10)) Then
        Else
            Kalimat = Kalimat + Huruf
        End If
        i = i + 1
    Loop 'result is kalimat (character before enter)

    If Kalimat <> "" Then
        Batas_Table3 = Batas_Table3 + 1
        Table3_array(Batas_Table3) = Kalimat
    End If
Loop        'result is Table3_array

'Browsing array of outsider created from the buffer
'in order to get any pattern of threat
For i = 1 To Total_Outsider
    'status_Threat = False

    'Is there any effort to force user to read too much information
    'by flooding user with adds from many external servers?
    '-----------------------------------------------------------
    'If (status_Threat = False) Then
    Outside_Party = List_Outsider(i)
    Internal_Party = List_Internal(i)
    Waktu_access1 = List_AccessTime(i)
    Forced_Information = 0
    List_Bahaya4(i) = 0
    For j = 1 To Total_Outsider
        If (Outside_Party <> List_Outsider(j)) And (Internal_Party = List_Internal(j)) Then
            Waktu_access2 = List_AccessTime(j)
            BedaWaktu = DeltaTime(Waktu_access1, Waktu_access2)
            If BedaWaktu < 60 Then  'identifying the servers from the same site
                Forced_Information = Forced_Information + 1
                List_Bahaya4(i) = Forced_Information
            End If
        End If
    Next

    Risk_Bahaya1 = 0
```

```
Risk_Bahaya2 = 0
Risk_Bahaya3 = 0
Risk_Bahaya4 = 0
Risk_Bahaya5 = 0
Risk_Bahaya6 = 0

'EXE file
If List_Bahaya1(i) > 0 Then
   Risk_Bahaya1 = 1
Else
   Risk_Bahaya1 = 0
End If

'Number of active script -> not significant to detect the threat
'example: elsevier has 43 scripts, ieee has 15
'If List_Bahaya2(i) > 0 Then
'    'Risk_Bahaya2 = 1 / (1 + (1 / List_Bahaya2(i)) ^ (1/1.5))
'    Risk_Bahaya2 = 1 - (1 / List_Bahaya2(i) ^ (1 / 1.5))
'End If

'Number of Cookies -> less influence to the threat
'example: cisco has 8 cookies, psz and science direct have 4 cookies
If List_Bahaya3(i) > 0 Then
   'Risk_Bahaya3 = 1 / (1 + (1 / List_Bahaya3(i)))
   Risk_Bahaya3 = 1 - (1 / List_Bahaya3(i) ^ (1 / 2))
End If

'Number of external machines -> less influence to the threat
'example: ieee has 7 machines
If List_Bahaya4(i) > 0 Then
   'Risk_Bahaya4 = 1 / (1 + (1 / List_Bahaya4(i)))
   Risk_Bahaya4 = 1 - (1 / List_Bahaya4(i) ^ (1 / 2))
End If

'Add -> slightly influence the threat
If List_Bahaya5(i) > 0 Then
   Risk_Bahaya5 = 1 - (1 / List_Bahaya5(i) ^ (1.7))
End If

'Number of Forced Info -> extremely influence the threat
If List_Bahaya6(i) > 0 Then
   Risk_Bahaya6 = 1 - (1 / List_Bahaya6(i) ^ (3))
End If


'Computing total risk
If (Risk_Bahaya4 - Risk_Bahaya1) >= 0 Then
   Total_Risk = Risk_Bahaya4
Else
   Total_Risk = Risk_Bahaya1
End If
If (Total_Risk - Risk_Bahaya2) < 0 Then
   Total_Risk = Risk_Bahaya2
End If
If (Total_Risk - Risk_Bahaya3) < 0 Then
   Total_Risk = Risk_Bahaya3
End If
If (Total_Risk - Risk_Bahaya5) < 0 Then
   Total_Risk = Risk_Bahaya5
End If
```

```
If (Total_Risk - Risk_Bahaya6) < 0 Then
   Total_Risk = Risk_Bahaya6
End If
List_Risk(i) = Total_Risk

Select Case List_Risk(i)
   Case Is < 0.25
      List_Table(i) = "Table1"
      'Table1 Registration
      status_exist = False
      For j = 1 To Batas_Table1
         If Table1_array(j) = List_Outsider(i) Then
            status_exist = True
         End If
      Next
      If status_exist = False Then
         Batas_Table1 = Batas_Table1 + 1
         Table1_array(Batas_Table1) = List_Outsider(i)
      End If

      'delete the membership from other table
      k = 0
      For j = 1 To Batas_Table2       'Table2
         If Table2_array(j) = List_Outsider(i) Then
         Else
            k = k + 1
            Temporary_array(k) = Table2_array(j)
         End If
      Next
      Batas_Table2 = k
      For j = 1 To Batas_Table2
         Table2_array(j) = Temporary_array(j)
      Next

      k = 0
      For j = 1 To Batas_Table3       'Table3
         If Table3_array(j) = List_Outsider(i) Then
         Else
            k = k + 1
            Temporary_array(k) = Table3_array(j)
         End If
      Next
      Batas_Table3 = k
      For j = 1 To Batas_Table3
         Table3_array(j) = Temporary_array(j)
      Next

   Case 0.25 To 0.75
      List_Table(i) = "Table2"
      'Table2 Registration
      status_exist = False
      For j = 1 To Batas_Table2
         If Table2_array(j) = List_Outsider(i) Then
            status_exist = True
         End If
      Next
      If status_exist = False Then
         Batas_Table2 = Batas_Table2 + 1
         Table2_array(Batas_Table2) = List_Outsider(i)
      End If
```

```
    'delete the membership from other table
    k = 0
    For j = 1 To Batas_Table1       'Table1
       If Table1_array(j) = List_Outsider(i) Then
       Else
          k = k + 1
          Temporary_array(k) = Table1_array(j)
       End If
    Next
    Batas_Table1 = k
    For j = 1 To Batas_Table1
       Table1_array(j) = Temporary_array(j)
    Next

    k = 0
    For j = 1 To Batas_Table3       'Table3
       If Table3_array(j) = List_Outsider(i) Then
       Else
          k = k + 1
          Temporary_array(k) = Table3_array(j)
       End If
    Next
    Batas_Table3 = k
    For j = 1 To Batas_Table3
       Table3_array(j) = Temporary_array(j)
    Next

Case Is > 0.75
    List_Table(i) = "Table3"
    'Table3 Registration
    status_exist = False
    For j = 1 To Batas_Table3
       If Table3_array(j) = List_Outsider(i) Then
          status_exist = True
       End If
    Next
    If status_exist = False Then
       Batas_Table3 = Batas_Table3 + 1
       Table3_array(Batas_Table3) = List_Outsider(i)
    End If

    'delete the membership from other table
    k = 0
    For j = 1 To Batas_Table1       'Table1
       If Table1_array(j) = List_Outsider(i) Then
       Else
          k = k + 1
          Temporary_array(k) = Table1_array(j)
       End If
    Next
    Batas_Table1 = k
    For j = 1 To Batas_Table1
       Table1_array(j) = Temporary_array(j)
    Next

    k = 0
    For j = 1 To Batas_Table2       'Table2
       If Table2_array(j) = List_Outsider(i) Then
       Else
```

```
            k = k + 1
            Temporary_array(k) = Table2_array(j)
          End If
        Next
        Batas_Table2 = k
        For j = 1 To Batas_Table2
          Table2_array(j) = Temporary_array(j)
        Next

    End Select
Next

'Send the result to Table files
Report_File = "D:\Project\Fuzzy-based\Table\fuzzy_table1"
WriteTextFileContents (""), Report_File, False
For m = 1 To Batas_Table1
    WriteTextFileContents (Table1_array(m) + " "), Report_File, True
Next
Report_File = "D:\Project\Fuzzy-based\Table\fuzzy_table2"
WriteTextFileContents (""), Report_File, False
For m = 1 To Batas_Table2
    WriteTextFileContents (Table2_array(m) + " "), Report_File, True
Next
Report_File = "D:\Project\Fuzzy-based\Table\fuzzy_table3"
WriteTextFileContents (""), Report_File, False
For m = 1 To Batas_Table3
    WriteTextFileContents (Table3_array(m) + " "), Report_File, True
Next

'Send the results to firewall
FileCopy "D:\Project\Fuzzy-based\Table\fuzzy_table1", "Z:\fuzzy_table1"
FileCopy "D:\Project\Fuzzy-based\Table\fuzzy_table2", "Z:\fuzzy_table2"
FileCopy "D:\Project\Fuzzy-based\Table\fuzzy_table3", "Z:\fuzzy_table3"

'Presenting the result
List1.AddItem "Summary:"
List1.AddItem "Number of external server machine: " & Total_Outsider
List1.AddItem "Number of EXE file: " & Total_Bahaya1
List1.AddItem "Number of active script: " & Total_Bahaya2
List1.AddItem "Number of cookies: " & Total_Bahaya3
List1.AddItem "Number of add: " & Total_Bahaya5
List1.AddItem "Number of forced info: " & Total_Bahaya6

For m = 1 To (batas_index_outsider - 1)
    List1.AddItem " "
    List1.AddItem List_Outsider(m)
    List1.AddItem "Internal user          " & List_Internal(m)
    List1.AddItem "First access           " & List_AccessTime(m)
    'List1.AddItem List_Kalimat(m)
    List1.AddItem "Risk Value             " & List_Risk(m)
    List1.AddItem List_Table(m)
    List1.AddItem "EXE Files              " & List_Bahaya1(m)
    List1.AddItem "Other involved machines  " & List_Bahaya4(m)
    List1.AddItem "Active Scripts         " & List_Bahaya2(m)
    List1.AddItem "Cookies                " & List_Bahaya3(m)
    List1.AddItem "Adds                   " & List_Bahaya5(m)
    List1.AddItem "Forced Info            " & List_Bahaya6(m)
Next

'Writing the report to a text file
```

```
    'Report_File = "D:\Project\Fuzzy Traffic Analyzer\Report\" + File1.filename
    Report_File = "D:\Project\Fuzzy-based\Table\fuzzy_report"
    WriteTextFileContents ("Buffer name: " + Nama_Traffic_Buffer), Report_File, False
    For m = 1 To ((batas_index_outsider - 1) * 12) + 7
        WriteTextFileContents (List1.List(m - 1)), Report_File, True
    Next

    'testing only
    List2.Clear
    List2.AddItem "TABLE1"
    For i = 1 To Batas_Table1
        List2.AddItem Table1_array(i)
    Next
    List2.AddItem " "
    List2.AddItem "TABLE2"
    For i = 1 To Batas_Table2
        List2.AddItem Table2_array(i)
    Next
    List2.AddItem " "
    List2.AddItem "TABLE3"
    For i = 1 To Batas_Table3
        List2.AddItem Table3_array(i)
    Next

    'closing the process
    Erase Kumpulan_Kata
    Erase List_Outsider
    'Erase List_Kalimat
    Erase List_Bahaya1
    Erase List_Bahaya2
    Erase List_Bahaya3
    Erase List_Bahaya4
    Erase List_Bahaya5
    Erase List_Bahaya6
    Erase List_Risk
    Erase List_Table
    Erase Suspicious
    Erase Table1_array
    Erase Table2_array
    Erase Table3_array
    Erase Temporary_array

    Text2.Text = "End at " & Now

    ' End reporting
    Report_File = "D:\Project\Fuzzy-based\Table\fuzzy_report"
    WriteTextFileContents (" "), Report_File, True
    WriteTextFileContents (Text1.Text), Report_File, True
    WriteTextFileContents (Text2.Text), Report_File, True
    WriteTextFileContents ("File size " & FileLen(Nama_Traffic_Buffer) & " Bytes"), Report_File,
True

End Sub
```

APPENDIX C1
The Implementation of Distributed Agent-Based Module

```
Option Explicit
Public wS_FileName          As New Class1
Public wSuspendMode         As Boolean
Dim milidetik As Double
Dim waktu_skr_ref As Long
Public wkt_skr_real As Double
Public wkt_skr_string As String

Private Sub Clear_Click()
   List1.Clear
   Text1.Text = ""
   Text2.Text = ""
End Sub

Private Sub Form_Activate()

   Dim Number_processes

   milidetik = 0
   waktu_skr_ref = Second(Time) + 60 * Minute(Time) + 3600 * Hour(Time)

   Timer1.Interval = 500 'approx 500 equal to 1 sec based on E-book
   'Timer2.Interval = 5000
   Timer2.Enabled = False
   Timer3.Interval = 1

   Start.Enabled = True
   Stop1.Enabled = False

   Number_processes = Module2.List_ActiveProcess(TreeView1)
   Module2.List_ActiveModules TreeView1
   'Text1.Text = ReadTextFileContents("D:\\Project\Memory Inspector\Virus_list.txt")

   Label29.Caption = Number_processes
   Label30.Caption = TreeView1.Nodes(1).FirstSibling.Text
   Label31.Caption = TreeView1.Nodes(1).LastSibling.Text
   Label32.Caption = TreeView1.Nodes(1).LastSibling.Children

   Dim hi As SYSTEMINFO
      GetSystemInfo hi
      Label12.Caption = hi.dwNumberOfProcessors
      Label11.Caption = hi.dwProcessorType
      Label10.Caption = Hex$(hi.lpMinimumApplicationAddress) & "H"
      Label9.Caption = Hex$(hi.lpMaximumApplicationAddress) & "H"

   Dim si&
   #If Win32 Then
      myVer.dwOSVersionInfoSize = 148
      si& = GetVersionEx&(myVer)
      If myVer.dwPlatformId <= 4 Then
         Label21.Caption = "Windows NT"
      ElseIf myVer.dwPlatformId = 5 Then
         Label21.Caption = "Windows 2000"
      End If
      Label22.Caption = myVer.dwPlatformId
      Label23.Caption = myVer.szCSDVersion
      'Label24.Caption = myVer.dwBuildNumber
   #End If
```

```
    Label24.Caption = Winsock1.LocalIP 'For software info

    'Dim vernum&
        'vernum& = GetVersion&()
        'Print vernum&

    'Print Second(Time)
    'Print Time

End Sub

Private Sub Form_Click()
    'Dim ms As MEMORYSTATUS
        'ms.dwLength = Len(ms)
        'GlobalMemoryStatus ms
        'Print "Total physical memory: "; ms.dwTotalPhys
        'Print "Available physical memory: "; ms.dwAvailPhys
        'Label5.Caption = ms.dwTotalPhys
        'Label6.Caption = ms.dwTotalPageFile
        'Label7.Caption = ms.dwAvailPhys
        'Label8.Caption = ms.dwLength
End Sub

Private Sub Start_Click()
    'Dim i As Integer
    Start.Enabled = False
    Stop1.Enabled = True
    Timer2.Interval = 500
    Timer2.Enabled = True

    'i = 0
    'Do Until i = 1000000
    '    Module8.Timer_Action
    '    Form1.Refresh
    '    If Stop1.Value = True Then
    '        Exit Do
    '    End If
    'Loop

End Sub

Private Sub Stop1_Click()
    'Dim taskID As Long
    Start.Enabled = True
    Stop1.Enabled = False
    Timer2.Enabled = False
    'taskID = ExecuteTask("C:\Program Files\Microsoft Office\Office\Winword.exe")

End Sub

Private Sub Timer1_Timer()
    On Error Resume Next
    Dim ms As MEMORYSTATUS 'For physical memory info
        ms.dwLength = Len(ms)
        GlobalMemoryStatus ms
        Label5.Caption = (ms.dwTotalPhys \ 1000)
        Label6.Caption = ((ms.dwTotalPhys - ms.dwAvailPhys) \ 1000)
        Label7.Caption = (ms.dwAvailPhys \ 1000)
        Label8.Caption = ms.dwMemoryLoad
```

```
End Sub

Private Sub Timer2_Timer()
   'On Error Resume Next
   Module9.Timer_Action   'written in module8.bas
End Sub
Function ReadTextFileContents(filename As String) As String
   Dim fnum As Integer, isOpen As Boolean
   On Error GoTo Error_Handler
   ' Get the next free file number.
   fnum = FreeFile()
   Open filename For Input As #fnum
   ' If execution flow got here, the file has been open without error.
   isOpen = True
   ' Read the entire contents in one single operation.
   ReadTextFileContents = Input(LOF(fnum), fnum)
   ' Intentionally flow into the error handler to close the file.
Error_Handler:
   ' Raise the error (if any), but first close the file.
   If isOpen Then Close #fnum
   'If Err Then Err.Raise Err.Number, , Err.Description
End Function
Sub WriteTextFileContents(Text As String, filename As String, Optional AppendMode As Boolean)
   Dim fnum As Integer, isOpen As Boolean
   On Error GoTo Error_Handler
   ' Get the next free file number.
   fnum = FreeFile()
   If AppendMode Then
      Open filename For Append As #fnum
    Else
      Open filename For Output As #fnum
    End If
    ' If execution flow gets here, the file has been opened correctly.
    isOpen = True
    ' Print to the file in one single operation.
    Print #fnum, Text
    ' Intentionally flow into the error handler to close the file.
Error_Handler:
   ' Raise the error (if any), but first close the file.
   If isOpen Then Close #fnum
   'If Err Then Err.Raise Err.Number, , Err.Description
End Sub

Private Sub Timer3_Timer()
   Dim wkt_skr As Long
   Dim detik, menit, jam As Long

   detik = Second(Time)
   menit = Minute(Time)
   jam = Hour(Time)
   wkt_skr = detik + 60 * menit + 3600 * jam
   If wkt_skr = waktu_skr_ref Then
      milidetik = milidetik + 1.587
   Else
      'Text2.Text = milidetik
      waktu_skr_ref = wkt_skr
      milidetik = 0
   End If

   wkt_skr_real = wkt_skr + milidetik / 100
```

```
    wkt_skr_string = jam & ":" & menit & ":" & detik & "." & milidetik '/ 100
End Sub

Option Compare Text
Option Explicit
Dim ukuran_aplikasi_lg_jalan As Long
Public Sub Timer_Action()
    Dim Number_processes
    Dim i, j As Integer
    Dim application_list As String
    Dim number_application_match As Integer
    Dim running_application As String
    Dim running_application_list As String
    Dim data_running As String
    Dim data_secure As String
    Dim index_applikasi_lgjln As Integer
    Dim wkt_mulai As Date
    Dim elapsed_time As Integer

    wkt_mulai = Time
    Form1.Stop1.Enabled = False

    Number_processes = Module2.List_ActiveProcess(Form1.TreeView1)
    'Module2.List_ActiveModules Form1.TreeView1

    Form1.Label29.Caption = Number_processes
    Form1.Label30.Caption = Form1.TreeView1.Nodes(1).FirstSibling.Text
    Form1.Label31.Caption = Form1.TreeView1.Nodes(1).LastSibling.Text
    Form1.Label32.Caption = Form1.TreeView1.Nodes(1).LastSibling.Children

    'Creating the database of running application using array
    Dim aplikasi_lg_jalan(1 To 1000) As String
    For i = 3 To Number_processes 'i=3 for avoiding the content of [System Process] written in the file
        aplikasi_lg_jalan(i - 2) = Form1.TreeView1.Nodes(i).Text
    Next
    'Getting the file size of running applications by calling Delphi application
    'Dim taskID As Long 'the product is result_carifile.txt
    'taskID = ExecuteTask("D:\Project\Finding_file\Project1.exe")
    'Getting the file size of running applications by calling FindFilesandSize procedure
    For i = 1 To Number_processes - 2
       elapsed_time = (Time - wkt_mulai) * 100000
       Form1.StatusBar1.SimpleText = "Analyzing " + aplikasi_lg_jalan(i) + " with elapsed time " &
elapsed_time & " seconds"
       ukuran_aplikasi_lg_jalan = FindFileSizefromDB(aplikasi_lg_jalan(i))
       'FindFilesandSize "c:", aplikasi_lg_jalan(i)
       aplikasi_lg_jalan(i) = aplikasi_lg_jalan(i) + " " & ukuran_aplikasi_lg_jalan 'modifying array
running appl
    Next i
    'only for documentation
    Dim doc_applikasi_lg_jln As String
    doc_applikasi_lg_jln = "D:\Project\Active Program Inspector\doc_applikasi_lg_jln.txt"
    Kill (doc_applikasi_lg_jln)
    For i = 1 To Number_processes - 2
       Form1.WriteTextFileContents (aplikasi_lg_jalan(i)), doc_applikasi_lg_jln, True
    Next i
    'Reading the database of secure application (consist of appl name and size)
    application_list = "D:\Project\Building_Application_DB\ApplicationDBNS.txt"
    'running_application_list = "D:\Project\Finding_file\result_carifile.txt"
    'Comparing the list of running application and the reference (available application software)
    Dim fnum1, fnum2 As Integer
```

```
    fnum1 = FreeFile()
    Open application_list For Input As #fnum1
    'fnum2 = FreeFile()
    'Open running_application_list For Input As #fnum2
    Dim Isi_applList, Isi_runningAppl As String
    Dim status As Boolean
    Dim MgknVirus(1 To 1000) As String
    Dim MgknNormal(1 To 1000) As String
    Dim isi_reference(1 To 100000) As String
    Dim hitung As Integer
    Dim ngitung As Integer
    Dim index_ref, jml_ref As Integer
    index_ref = 1
    Do Until EOF(fnum1)
       Line Input #fnum1, isi_reference(index_ref)
       index_ref = index_ref + 1
    Loop
    jml_ref = index_ref - 1
    hitung = 1
    ngitung = 1
    'Do Until EOF(fnum2)
    index_applikasi_lgjln = 1
    Do Until index_applikasi_lgjln = (Number_processes - 1)
       status = False
       Isi_runningAppl = aplikasi_lg_jalan(index_applikasi_lgjln)
       index_applikasi_lgjln = index_applikasi_lgjln + 1
       For index_ref = 1 To jml_ref
          Isi_applList = isi_reference(index_ref)
          If Isi_runningAppl = Isi_applList Then
             status = True
          End If
          If status = True Then          'file is matched with reference
             MgknNormal(ngitung) = Isi_runningAppl + " " + Isi_applList 'list of ok appl
             ngitung = ngitung + 1
             Exit For
          End If
       Next
       If status = False Then              'MALICIOUS PROGRAM - running file is not matched with
reference
          'Text1.Text = "virus" + " " + Isi_applList + " " + Isi_runningAppl
          'MgknVirus(hitung) = Isi_runningAppl     'suspected as virus
          MgknVirus(hitung) = Isi_runningAppl      'list of suspected virus
          Form1.List1.AddItem (Isi_runningAppl + " at " & Form1.wkt_skr_real)
          hitung = hitung + 1
       End If
    Loop
    Dim suspected_virus_list As String          'Reporting suspicious process
    Dim agent_report As String
    suspected_virus_list = "D:\Project\Active Program Inspector\suspected_virus_list.txt"
    agent_report = "Z:\agent_report"
    'Kill (suspected_virus_list) 'reset/rewrite mode
    If hitung > 1 Then
       Form1.Text1.Text = "threat" + " " & Now
       For i = 1 To (hitung - 1)
          Form1.WriteTextFileContents (Form1.Label24.Caption + " " + MgknVirus(i)),
suspected_virus_list, True 'append mode
          Form1.WriteTextFileContents (Form1.Label24.Caption + " " + MgknVirus(i)), agent_report,
True 'append mode
          'MgknVirus(i) is suspicious process and Label24 is IPaddr of the machine
       Next
```

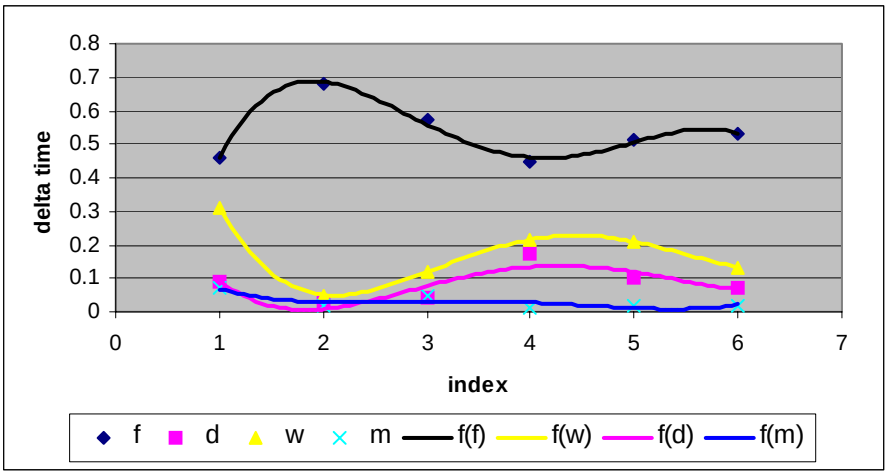
```
    Else
       'Form1.Text1.Text = "secure"
    End If
    Dim normal_appl_list As String              'Reporting normal application
    normal_appl_list = "D:\Project\Active Program Inspector\normal_appl_list.txt"
    Kill (normal_appl_list) 'reset/rewrite mode
    If ngitung > 1 Then
       For i = 1 To (ngitung - 1)
          Form1.WriteTextFileContents (MgknNormal(i)), normal_appl_list, True 'append mode
       Next
    End If
    Close #fnum1
    Erase MgknVirus
    Erase MgknNormal
    Erase isi_reference
    Erase aplikasi_lg_jalan
    Form1.Stop1.Enabled = True
End Sub

Private Function FindFileSizefromDB(nama_file As String) As Long
    Dim statuscari As Boolean
    Dim fnum1, fnum2 As Integer
    Dim namafilename, namafilepath As String
    Dim filename, filepath As String
    Dim filesize As Long
    namafilename = "D:\Project\Building_Application_DB\ApplicationDBN.txt"
    namafilepath = "D:\Project\Building_Application_DB\ApplicationDBP.txt"
    fnum1 = FreeFile()
    Open namafilename For Input As #fnum1
    fnum2 = FreeFile()
    Open namafilepath For Input As #fnum2
    statuscari = False
    Do Until EOF(fnum1)
       Line Input #fnum1, filename
       Line Input #fnum2, filepath
       If filename = nama_file Then                 'folder of the file has been found
          filesize = FileLen(filepath + nama_file)   'file size is obtained
          statuscari = True
          Exit Do
       End If
    Loop
    Close #fnum1
    Close #fnum2
    If statuscari = True Then
       FindFileSizefromDB = filesize 'file size is obtained
    Else
       FindFileSizefromDB = 0
    End If
End Function
```

```
#!/bin/sh

i=1
while [ $i -lt 4 ]
do
        echo ""
        i=`expr ${i} + 1`
done

echo "################################################"
echo "#                                              #"
echo "#  ACTIVE FIREWALL                             #"
echo "#  Mode of operation: DISTRIBUTED AGENT-BASED  #"
echo "#                                              #"
echo "################################################"

i=1
while [ $i -lt 2 ]
do
        echo ""
        i=`expr ${i} + 1`
done

# eth0 = EXTERNAL NETWORK
# eth1 = INTERNAL NETWORK
# Need to run ./PROJECT/FIREWALL/timestamp in other console to get the timing of the firewall
# To simulate communication, pls run ./PROJECT/FIREWALL/Agent_FW_communication

# ------------------
# 1. Initialization
# ------------------

INTERNAL_SUBNET="24.4.74"

> /PROJECT/TEMP/AGENT_HISTORY

cd /PROJECT/FIREWALL
./agent_based_preconfiguration
# To preconfigure firewall machine, default open for start up

echo "Security Status          : Secure"
echo "Firewall Status          : Running Normally"
i=1
while [ $i -lt 3 ]
do
        echo ""
        i=`expr ${i} + 1`
done


# ------------------------------------------------------------------------------------------------
# 2. Monitoring the result of Distributed Agent-Based Security Modules
# ------------------------------------------------------------------------------------------------

while [ 1 ]
do
        # Checking intrusion
        # ------------------
        # cp -f /UMUM/NEWS/report.txt /PROJECT/TEMP/report.txt
        # > /UMUM/NEWS/report.txt
```

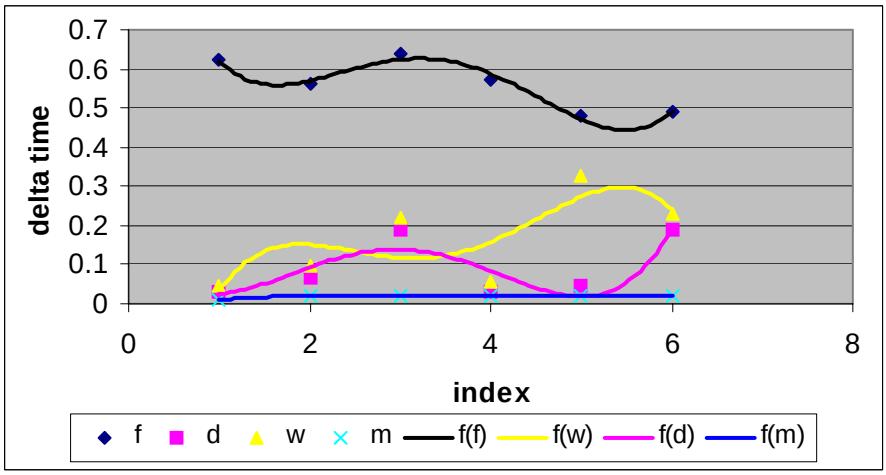
```
        cp -f /UMUM/NEWS/agent_report /PROJECT/TEMP/agent_report
        > /UMUM/NEWS/agent_report

        JUMLAH_BARIS=`wc -l /PROJECT/TEMP/agent_report | awk '{print $1}'`
        #echo "$JUMLAH_BARIS"

        # There is/are intrusions if (JUMLAH_BARIS = > 1)
        if [[ $JUMLAH_BARIS -ge 1 ]]
        then
                # Get the computer being attacked and close communication line of this machine

                echo "################################################"
                echo "#                                              #"
                echo "#  ACTIVE FIREWALL                             #"
                echo "#  Mode of operation: DISTRIBUTED AGENT-BASED   #"
                echo "#                                               #"
                echo "################################################"

                echo ""

                # date
                # cat /PROJECT/TEMP/TIMESTAMP #for experiment
                # echo doing action -----
                # echo .
                # echo .
                i=1
                while [[ ${i:=1} -le $JUMLAH_BARIS ]]
                do
                        victim_PC=`head -n $i /PROJECT/TEMP/agent_report >
                        /PROJECT/TEMP/agentreportTEMP | tail -n 1
                        /PROJECT/TEMP/agentreportTEMP | awk '{print $1}'`
                        program_name=`head -n $i /PROJECT/TEMP/agent_report >
                        /PROJECT/TEMP/agentreportTEMP | tail -n 1
                        /PROJECT/TEMP/agentreportTEMP | awk '{print $2}'`

                        echo "Suspicious program $i $program_name running in $victim_PC"

                        status_victim=""
                        status_victim=`cat /PROJECT/TEMP/AGENT_HISTORY | grep
                        $victim_PC`
                        if [ "$status_victim" = "" ]
                        then
                                echo "Communication is closed for $victim_PC"
                                echo $victim_PC >> /PROJECT/TEMP/AGENT_HISTORY
                                iptables -I FORWARD 1 -i eth1 -p tcp -s $victim_PC -d 0/0 -j
                                DROP
                                iptables -I FORWARD 1 -i eth0 -p tcp -s 0/0 -d $victim_PC -j
                                DROP
                                date
                        fi
                        i=`expr ${i} + 1`
                        # echo $i
                done
                echo "=====> Action to isolate victim machine has been accomplished"
                > /PROJECT/TEMP/agent_report
        fi
done

echo "Finish"
```

APPENDIX C3
Graph Presentation of the Speed of Attacks, Closing Canals, Threat Detections and Starting Unauthorized Information Flows

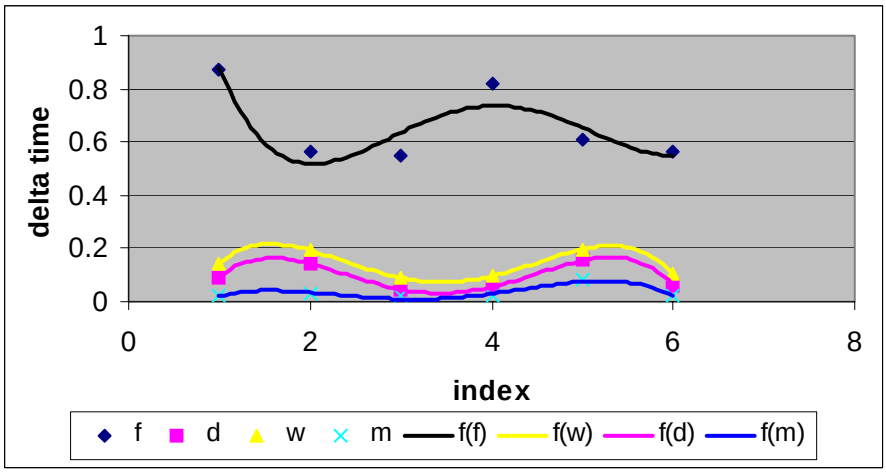
GROUP 1: Experiment 1 - 6



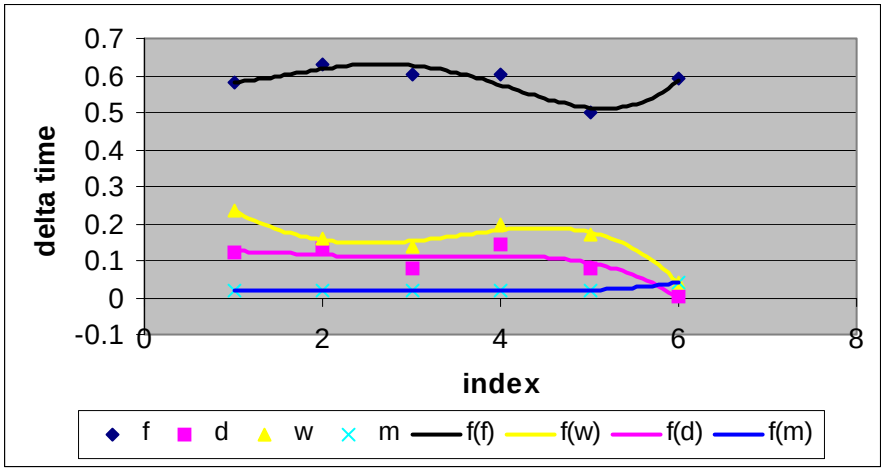
GROUP 2: Experiment 7 – 12



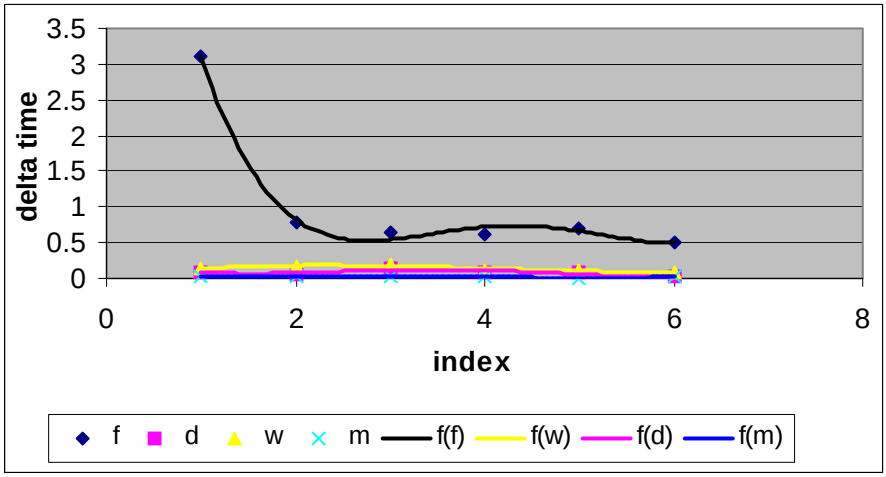
GROUP 3: Experiment 13 – 18



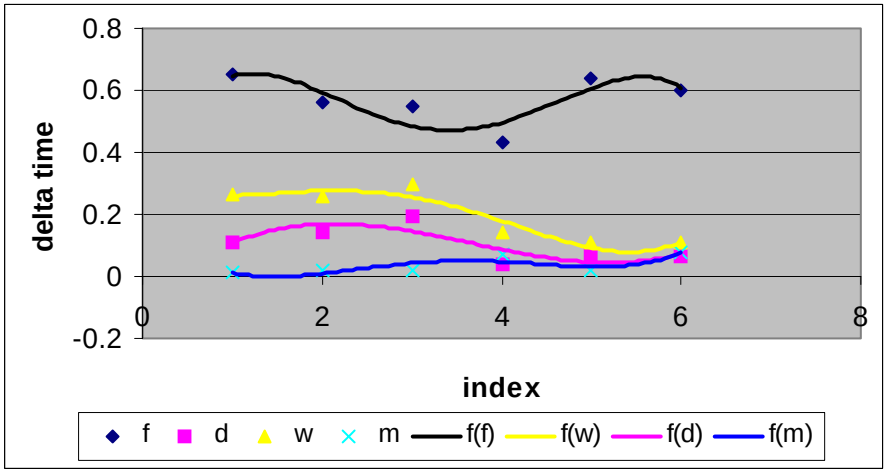
GROUP 4: Experiment 19 – 24



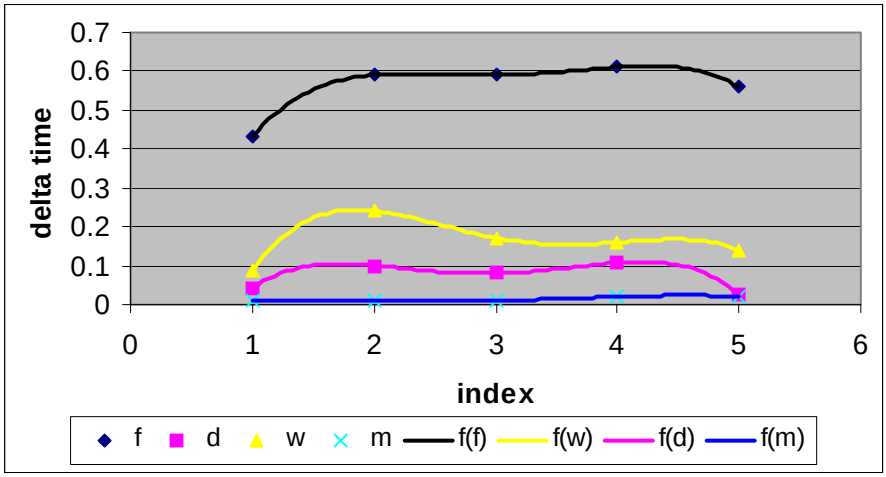
GROUP 5: Experiment 25 – 30



GROUP 6: Experiment 31 – 36



GROUP 7: Experiment 37 – 41



APPENDIX C4
Calculations for Experimental Results of Agent-Based Firewall

Table C2: Function of experimental results

Experiment Index	$f(x)$	$F = \int f(x)dx$
Group 1	$f_f(x)$ = -0.0113x4 + 0.1768x3 - 0.9557x2 + 2.0257x - 0.7757 R2 = 0.9907 $f_w(x)$ = 0.007x4 - 0.1196x3 + 0.7085x2 - 1.6539x + 1.3694 R2 = 0.9994 $f_d(x)$ = 0.0042x4 - 0.0692x3 + 0.3861x2 - 0.8213x + 0.5931 R2 = 0.7404 $f_m(x)$ = 0.0019x4 - 0.0271x3 + 0.1369x2 - 0.2885x + 0.245 R2 = 0.7005	2.746917 0.844083 0.39075 0.168292
Group 2	$f_f(x)$ = 0.0086x4 - 0.118x3 + 0.5459x2 - 0.9897x + 1.1725 R2 = 0.9651 $f_w(x)$ = -0.0093x4 + 0.1308x3 - 0.6265x2 + 1.2218x - 0.6832 R2 = 0.5792 $f_d(x)$ = 0.0069x4 - 0.0817x3 + 0.3071x2 - 0.386x + 0.1803 R2 = 0.7705 $f_m(x)$ = -0.0003x4 + 0.0043x3 - 0.0245x2 + 0.058x - 0.0275 R2 = 0.9898	2.836083 0.951333 0.434458 0.047292
Group 3	$f_f(x)$ = 0.0137x4 - 0.2201x3 + 1.216x2 - 2.6745x + 2.5443 R2 = 0.8257 $f_w(x)$ = -0.0115x4 + 0.1585x3 - 0.7395x2 + 1.3289x - 0.5906 R2 = 0.9995 $f_d(x)$ = -0.0111x4 + 0.1531x3 - 0.7138x2 + 1.2858x - 0.6205 R2 = 1 $f_m(x)$ = -0.0044x4 + 0.0588x3 - 0.2626x2 + 0.4587x - 0.2315 R2 = 0.9134	3.110542 0.737125 0.548958 0.244583
Group 4	$f_f(x)$ = 0.0038x4 - 0.0441x3 + 0.157x2 - 0.184x + 0.65 R2 = 0.8224 $f_w(x)$= -0.0011x4 + 0.0033x3 + 0.0452x2 - 0.2231x + 0.4113 R2 = 0.9776 $f_d(x)$ = -0.0013x4 + 0.0136x3 - 0.0465x2 + 0.053x + 0.1065 R2 = 0.8128 $f_m(x)$ = 0.0004x4 - 0.0049x3 + 0.0199x2 - 0.032x + 0.0367 R2 = 0.9952	2.913292 0.749458 0.509 0.085292

Table C2: cont.

Group 5	$f_f(x) = 0.0345x4 - 0.5986x3 + 3.7242x2 - 9.7742x + 9.7193$ $R2 = 0.9953$ $f_w(x) = -0.0004x4 + 0.0087x3 - 0.0693x2 + 0.1979x - 0.0004$ $R2 = 0.8791$ $f_d(x) = 0.0037x4 - 0.054x3 + 0.2581x2 - 0.4679x + 0.3532$ $R2 = 0.6576$ $f_m(x) = 0.0006x4 - 0.0081x3 + 0.0351x2 - 0.0591x + 0.0517$ $R2 = 0.8809$	4.29975 0.689375 0.345917 0.050375
Group 6	$f_f(x) = -0.0082x4 + 0.1113x3 - 0.4929x2 + 0.7715x + 0.2627$ $R2 = 0.6642$ $f_w(x) = 0.0029x4 - 0.0332x3 + 0.1084x2 - 0.1217x + 0.3041$ $R2 = 0.8894$ $f_d(x) = -2E-05x4 + 0.0088x3 - 0.0953x2 + 0.2863x - 0.0923$ $R2 = 0.6321$ $f_m(x) = 0.0031x4 - 0.0424x3 + 0.1948x2 - 0.3383x + 0.195$ $R2 = 0.7156$	2.772625 0.920417 0.536817 0.108917
Group 7	$f_f(x) = -0.0113x4 + 0.1429x3 - 0.6552x2 + 1.2946x - 0.34$ $R2 = 1$ $f_w(x) = -0.0146x4 + 0.1932x3 - 0.9072x2 + 1.744x - 0.93$ $R2 = 1$ $f_d(x) = -0.0116x4 + 0.1364x3 - 0.5672x2 + 0.977x - 0.4912$ $R2 = 1$ $f_m(x) = -0.0013x4 + 0.0142x3 - 0.0538x2 + 0.0808x - 0.03$ $R2 = 1$	2.32576 0.72752 0.345653 0.028827

APPENDIX C1
The Implementation of Distributed Agent-Based Module

```
Option Explicit
Public wS_FileName          As New Class1
Public wSuspendMode         As Boolean
Dim milidetik As Double
Dim waktu_skr_ref As Long
Public wkt_skr_real As Double
Public wkt_skr_string As String

Private Sub Clear_Click()
   List1.Clear
   Text1.Text = ""
   Text2.Text = ""
End Sub

Private Sub Form_Activate()

   Dim Number_processes

   milidetik = 0
   waktu_skr_ref = Second(Time) + 60 * Minute(Time) + 3600 * Hour(Time)

   Timer1.Interval = 500 'approx 500 equal to 1 sec based on E-book
   'Timer2.Interval = 5000
   Timer2.Enabled = False
   Timer3.Interval = 1

   Start.Enabled = True
   Stop1.Enabled = False

   Number_processes = Module2.List_ActiveProcess(TreeView1)
   Module2.List_ActiveModules TreeView1
   'Text1.Text = ReadTextFileContents("D:\\Project\Memory Inspector\Virus_list.txt")

   Label29.Caption = Number_processes
   Label30.Caption = TreeView1.Nodes(1).FirstSibling.Text
   Label31.Caption = TreeView1.Nodes(1).LastSibling.Text
   Label32.Caption = TreeView1.Nodes(1).LastSibling.Children

   Dim hi As SYSTEMINFO
      GetSystemInfo hi
      Label12.Caption = hi.dwNumberOfProcessors
      Label11.Caption = hi.dwProcessorType
      Label10.Caption = Hex$(hi.lpMinimumApplicationAddress) & "H"
      Label9.Caption = Hex$(hi.lpMaximumApplicationAddress) & "H"

   Dim si&
   #If Win32 Then
      myVer.dwOSVersionInfoSize = 148
      si& = GetVersionEx&(myVer)
      If myVer.dwPlatformId <= 4 Then
         Label21.Caption = "Windows NT"
      ElseIf myVer.dwPlatformId = 5 Then
         Label21.Caption = "Windows 2000"
      End If
      Label22.Caption = myVer.dwPlatformId
      Label23.Caption = myVer.szCSDVersion
      'Label24.Caption = myVer.dwBuildNumber
   #End If
```

```
    Label24.Caption = Winsock1.LocalIP 'For software info

    'Dim vernum&
        'vernum& = GetVersion&()
        'Print vernum&

    'Print Second(Time)
    'Print Time

End Sub

Private Sub Form_Click()
    'Dim ms As MEMORYSTATUS
        'ms.dwLength = Len(ms)
        'GlobalMemoryStatus ms
        'Print "Total physical memory: "; ms.dwTotalPhys
        'Print "Available physical memory: "; ms.dwAvailPhys
        'Label5.Caption = ms.dwTotalPhys
        'Label6.Caption = ms.dwTotalPageFile
        'Label7.Caption = ms.dwAvailPhys
        'Label8.Caption = ms.dwLength
End Sub

Private Sub Start_Click()
    'Dim i As Integer
    Start.Enabled = False
    Stop1.Enabled = True
    Timer2.Interval = 500
    Timer2.Enabled = True

    'i = 0
    'Do Until i = 1000000
    '    Module8.Timer_Action
    '    Form1.Refresh
    '    If Stop1.Value = True Then
    '        Exit Do
    '    End If
    'Loop

End Sub

Private Sub Stop1_Click()
    'Dim taskID As Long
    Start.Enabled = True
    Stop1.Enabled = False
    Timer2.Enabled = False
    'taskID = ExecuteTask("C:\Program Files\Microsoft Office\Office\Winword.exe")

End Sub

Private Sub Timer1_Timer()
    On Error Resume Next
    Dim ms As MEMORYSTATUS 'For physical memory info
        ms.dwLength = Len(ms)
        GlobalMemoryStatus ms
        Label5.Caption = (ms.dwTotalPhys \ 1000)
        Label6.Caption = ((ms.dwTotalPhys - ms.dwAvailPhys) \ 1000)
        Label7.Caption = (ms.dwAvailPhys \ 1000)
        Label8.Caption = ms.dwMemoryLoad
```

```
End Sub

Private Sub Timer2_Timer()
   'On Error Resume Next
   Module9.Timer_Action   'written in module8.bas
End Sub
Function ReadTextFileContents(filename As String) As String
   Dim fnum As Integer, isOpen As Boolean
   On Error GoTo Error_Handler
   ' Get the next free file number.
   fnum = FreeFile()
   Open filename For Input As #fnum
   ' If execution flow got here, the file has been open without error.
   isOpen = True
   ' Read the entire contents in one single operation.
   ReadTextFileContents = Input(LOF(fnum), fnum)
   ' Intentionally flow into the error handler to close the file.
Error_Handler:
   ' Raise the error (if any), but first close the file.
   If isOpen Then Close #fnum
   'If Err Then Err.Raise Err.Number, , Err.Description
End Function
Sub WriteTextFileContents(Text As String, filename As String, Optional AppendMode As Boolean)
   Dim fnum As Integer, isOpen As Boolean
   On Error GoTo Error_Handler
   ' Get the next free file number.
   fnum = FreeFile()
   If AppendMode Then
      Open filename For Append As #fnum
    Else
      Open filename For Output As #fnum
    End If
    ' If execution flow gets here, the file has been opened correctly.
    isOpen = True
    ' Print to the file in one single operation.
    Print #fnum, Text
    ' Intentionally flow into the error handler to close the file.
Error_Handler:
   ' Raise the error (if any), but first close the file.
   If isOpen Then Close #fnum
   'If Err Then Err.Raise Err.Number, , Err.Description
End Sub

Private Sub Timer3_Timer()
   Dim wkt_skr As Long
   Dim detik, menit, jam As Long

   detik = Second(Time)
   menit = Minute(Time)
   jam = Hour(Time)
   wkt_skr = detik + 60 * menit + 3600 * jam
   If wkt_skr = waktu_skr_ref Then
      milidetik = milidetik + 1.587
   Else
      'Text2.Text = milidetik
      waktu_skr_ref = wkt_skr
      milidetik = 0
   End If

   wkt_skr_real = wkt_skr + milidetik / 100
```

```
    wkt_skr_string = jam & ":" & menit & ":" & detik & "." & milidetik '/ 100
End Sub

Option Compare Text
Option Explicit
Dim ukuran_aplikasi_lg_jalan As Long
Public Sub Timer_Action()
    Dim Number_processes
    Dim i, j As Integer
    Dim application_list As String
    Dim number_application_match As Integer
    Dim running_application As String
    Dim running_application_list As String
    Dim data_running As String
    Dim data_secure As String
    Dim index_applikasi_lgjln As Integer
    Dim wkt_mulai As Date
    Dim elapsed_time As Integer

    wkt_mulai = Time
    Form1.Stop1.Enabled = False

    Number_processes = Module2.List_ActiveProcess(Form1.TreeView1)
    'Module2.List_ActiveModules Form1.TreeView1

    Form1.Label29.Caption = Number_processes
    Form1.Label30.Caption = Form1.TreeView1.Nodes(1).FirstSibling.Text
    Form1.Label31.Caption = Form1.TreeView1.Nodes(1).LastSibling.Text
    Form1.Label32.Caption = Form1.TreeView1.Nodes(1).LastSibling.Children

    'Creating the database of running application using array
    Dim aplikasi_lg_jalan(1 To 1000) As String
    For i = 3 To Number_processes 'i=3 for avoiding the content of [System Process] written in the file
        aplikasi_lg_jalan(i - 2) = Form1.TreeView1.Nodes(i).Text
    Next
    'Getting the file size of running applications by calling Delphi application
    'Dim taskID As Long 'the product is result_carifile.txt
    'taskID = ExecuteTask("D:\Project\Finding_file\Project1.exe")
    'Getting the file size of running applications by calling FindFilesandSize procedure
    For i = 1 To Number_processes - 2
       elapsed_time = (Time - wkt_mulai) * 100000
       Form1.StatusBar1.SimpleText = "Analyzing " + aplikasi_lg_jalan(i) + " with elapsed time " &
elapsed_time & " seconds"
       ukuran_aplikasi_lg_jalan = FindFileSizefromDB(aplikasi_lg_jalan(i))
       'FindFilesandSize "c:", aplikasi_lg_jalan(i)
       aplikasi_lg_jalan(i) = aplikasi_lg_jalan(i) + " " & ukuran_aplikasi_lg_jalan 'modifying array
running appl
    Next i
    'only for documentation
    Dim doc_applikasi_lg_jln As String
    doc_applikasi_lg_jln = "D:\Project\Active Program Inspector\doc_applikasi_lg_jln.txt"
    Kill (doc_applikasi_lg_jln)
    For i = 1 To Number_processes - 2
       Form1.WriteTextFileContents (aplikasi_lg_jalan(i)), doc_applikasi_lg_jln, True
    Next i
    'Reading the database of secure application (consist of appl name and size)
    application_list = "D:\Project\Building_Application_DB\ApplicationDBNS.txt"
    'running_application_list = "D:\Project\Finding_file\result_carifile.txt"
    'Comparing the list of running application and the reference (available application software)
    Dim fnum1, fnum2 As Integer
```

```
    fnum1 = FreeFile()
    Open application_list For Input As #fnum1
    'fnum2 = FreeFile()
    'Open running_application_list For Input As #fnum2
    Dim Isi_applList, Isi_runningAppl As String
    Dim status As Boolean
    Dim MgknVirus(1 To 1000) As String
    Dim MgknNormal(1 To 1000) As String
    Dim isi_reference(1 To 100000) As String
    Dim hitung As Integer
    Dim ngitung As Integer
    Dim index_ref, jml_ref As Integer
    index_ref = 1
    Do Until EOF(fnum1)
       Line Input #fnum1, isi_reference(index_ref)
       index_ref = index_ref + 1
    Loop
    jml_ref = index_ref - 1
    hitung = 1
    ngitung = 1
    'Do Until EOF(fnum2)
    index_applikasi_lgjln = 1
    Do Until index_applikasi_lgjln = (Number_processes - 1)
       status = False
       Isi_runningAppl = aplikasi_lg_jalan(index_applikasi_lgjln)
       index_applikasi_lgjln = index_applikasi_lgjln + 1
       For index_ref = 1 To jml_ref
          Isi_applList = isi_reference(index_ref)
          If Isi_runningAppl = Isi_applList Then
             status = True
          End If
          If status = True Then          'file is matched with reference
             MgknNormal(ngitung) = Isi_runningAppl + " " + Isi_applList 'list of ok appl
             ngitung = ngitung + 1
             Exit For
          End If
       Next
       If status = False Then              'MALICIOUS PROGRAM - running file is not matched with
reference
          'Text1.Text = "virus" + " " + Isi_applList + " " + Isi_runningAppl
          'MgknVirus(hitung) = Isi_runningAppl     'suspected as virus
          MgknVirus(hitung) = Isi_runningAppl      'list of suspected virus
          Form1.List1.AddItem (Isi_runningAppl + " at " & Form1.wkt_skr_real)
          hitung = hitung + 1
       End If
    Loop
    Dim suspected_virus_list As String          'Reporting suspicious process
    Dim agent_report As String
    suspected_virus_list = "D:\Project\Active Program Inspector\suspected_virus_list.txt"
    agent_report = "Z:\agent_report"
    'Kill (suspected_virus_list) 'reset/rewrite mode
    If hitung > 1 Then
       Form1.Text1.Text = "threat" + " " & Now
       For i = 1 To (hitung - 1)
          Form1.WriteTextFileContents (Form1.Label24.Caption + " " + MgknVirus(i)),
suspected_virus_list, True 'append mode
          Form1.WriteTextFileContents (Form1.Label24.Caption + " " + MgknVirus(i)), agent_report,
True 'append mode
          'MgknVirus(i) is suspicious process and Label24 is IPaddr of the machine
       Next
```

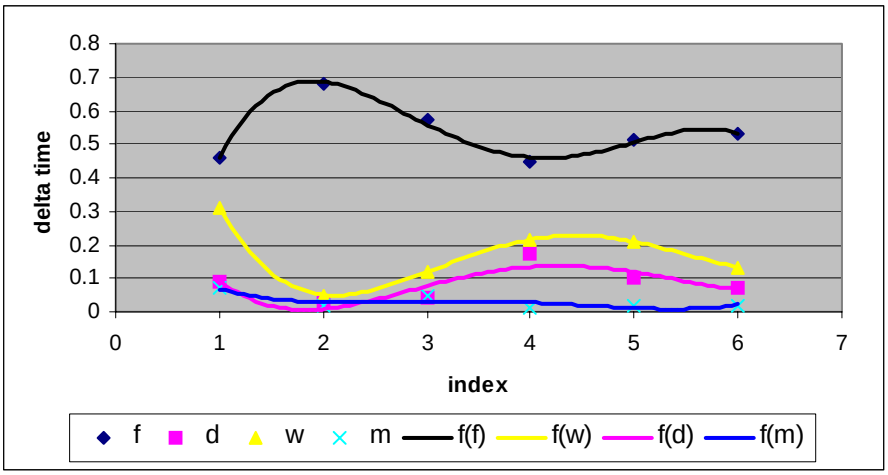
```
    Else
       'Form1.Text1.Text = "secure"
    End If
    Dim normal_appl_list As String              'Reporting normal application
    normal_appl_list = "D:\Project\Active Program Inspector\normal_appl_list.txt"
    Kill (normal_appl_list) 'reset/rewrite mode
    If ngitung > 1 Then
       For i = 1 To (ngitung - 1)
          Form1.WriteTextFileContents (MgknNormal(i)), normal_appl_list, True 'append mode
       Next
    End If
    Close #fnum1
    Erase MgknVirus
    Erase MgknNormal
    Erase isi_reference
    Erase aplikasi_lg_jalan
    Form1.Stop1.Enabled = True
End Sub

Private Function FindFileSizefromDB(nama_file As String) As Long
    Dim statuscari As Boolean
    Dim fnum1, fnum2 As Integer
    Dim namafilename, namafilepath As String
    Dim filename, filepath As String
    Dim filesize As Long
    namafilename = "D:\Project\Building_Application_DB\ApplicationDBN.txt"
    namafilepath = "D:\Project\Building_Application_DB\ApplicationDBP.txt"
    fnum1 = FreeFile()
    Open namafilename For Input As #fnum1
    fnum2 = FreeFile()
    Open namafilepath For Input As #fnum2
    statuscari = False
    Do Until EOF(fnum1)
       Line Input #fnum1, filename
       Line Input #fnum2, filepath
       If filename = nama_file Then                 'folder of the file has been found
          filesize = FileLen(filepath + nama_file)   'file size is obtained
          statuscari = True
          Exit Do
       End If
    Loop
    Close #fnum1
    Close #fnum2
    If statuscari = True Then
       FindFileSizefromDB = filesize 'file size is obtained
    Else
       FindFileSizefromDB = 0
    End If
End Function
```

```
#!/bin/sh

i=1
while [ $i -lt 4 ]
do
        echo ""
        i=`expr ${i} + 1`
done

echo "################################################"
echo "#                                              #"
echo "#  ACTIVE FIREWALL                             #"
echo "#  Mode of operation: DISTRIBUTED AGENT-BASED  #"
echo "#                                              #"
echo "################################################"

i=1
while [ $i -lt 2 ]
do
        echo ""
        i=`expr ${i} + 1`
done

# eth0 = EXTERNAL NETWORK
# eth1 = INTERNAL NETWORK
# Need to run ./PROJECT/FIREWALL/timestamp in other console to get the timing of the firewall
# To simulate communication, pls run ./PROJECT/FIREWALL/Agent_FW_communication

# ------------------
# 1. Initialization
# ------------------

INTERNAL_SUBNET="24.4.74"

> /PROJECT/TEMP/AGENT_HISTORY

cd /PROJECT/FIREWALL
./agent_based_preconfiguration
# To preconfigure firewall machine, default open for start up

echo "Security Status          : Secure"
echo "Firewall Status          : Running Normally"
i=1
while [ $i -lt 3 ]
do
        echo ""
        i=`expr ${i} + 1`
done


# ------------------------------------------------------------------------------------------------
# 2. Monitoring the result of Distributed Agent-Based Security Modules
# ------------------------------------------------------------------------------------------------

while [ 1 ]
do
        # Checking intrusion
        # ------------------
        # cp -f /UMUM/NEWS/report.txt /PROJECT/TEMP/report.txt
        # > /UMUM/NEWS/report.txt
```

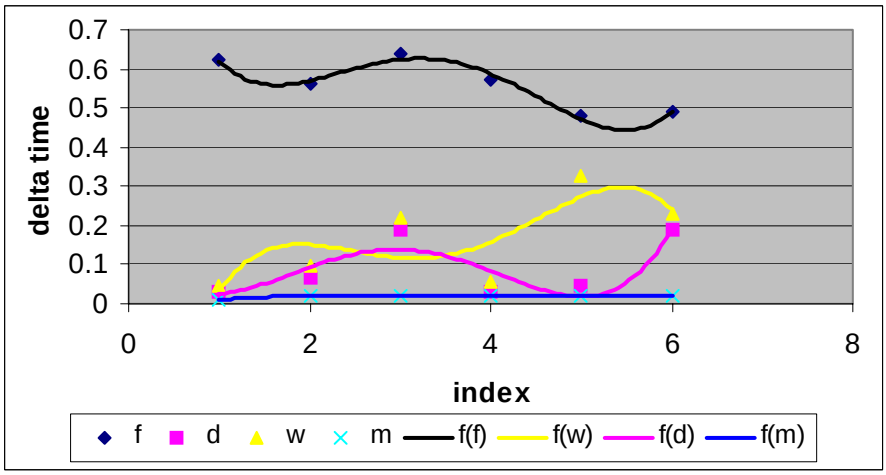
```
        cp -f /UMUM/NEWS/agent_report /PROJECT/TEMP/agent_report
        > /UMUM/NEWS/agent_report

        JUMLAH_BARIS=`wc -l /PROJECT/TEMP/agent_report | awk '{print $1}'`
        #echo "$JUMLAH_BARIS"

        # There is/are intrusions if (JUMLAH_BARIS = > 1)
        if [[ $JUMLAH_BARIS -ge 1 ]]
        then
                # Get the computer being attacked and close communication line of this machine

                echo "################################################"
                echo "#                                              #"
                echo "#  ACTIVE FIREWALL                             #"
                echo "#  Mode of operation: DISTRIBUTED AGENT-BASED   #"
                echo "#                                               #"
                echo "################################################"

                echo ""

                # date
                # cat /PROJECT/TEMP/TIMESTAMP #for experiment
                # echo doing action -----
                # echo .
                # echo .
                i=1
                while [[ ${i:=1} -le $JUMLAH_BARIS ]]
                do
                        victim_PC=`head -n $i /PROJECT/TEMP/agent_report >
                        /PROJECT/TEMP/agentreportTEMP | tail -n 1
                        /PROJECT/TEMP/agentreportTEMP | awk '{print $1}'`
                        program_name=`head -n $i /PROJECT/TEMP/agent_report >
                        /PROJECT/TEMP/agentreportTEMP | tail -n 1
                        /PROJECT/TEMP/agentreportTEMP | awk '{print $2}'`

                        echo "Suspicious program $i $program_name running in $victim_PC"

                        status_victim=""
                        status_victim=`cat /PROJECT/TEMP/AGENT_HISTORY | grep
                        $victim_PC`
                        if [ "$status_victim" = "" ]
                        then
                                echo "Communication is closed for $victim_PC"
                                echo $victim_PC >> /PROJECT/TEMP/AGENT_HISTORY
                                iptables -I FORWARD 1 -i eth1 -p tcp -s $victim_PC -d 0/0 -j
                                DROP
                                iptables -I FORWARD 1 -i eth0 -p tcp -s 0/0 -d $victim_PC -j
                                DROP
                                date
                        fi
                        i=`expr ${i} + 1`
                        # echo $i
                done
                echo "=====> Action to isolate victim machine has been accomplished"
                > /PROJECT/TEMP/agent_report
        fi
done

echo "Finish"
```

APPENDIX C3
Graph Presentation of the Speed of Attacks, Closing Canals, Threat Detections and Starting Unauthorized Information Flows

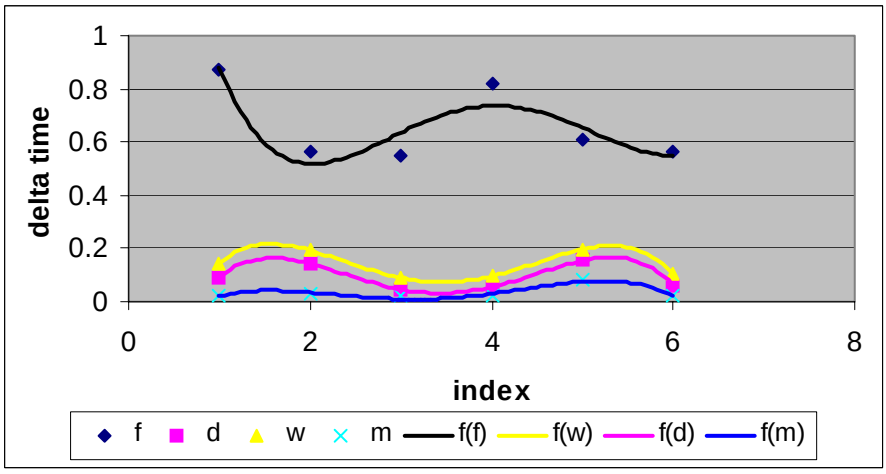
GROUP 1: Experiment 1 - 6



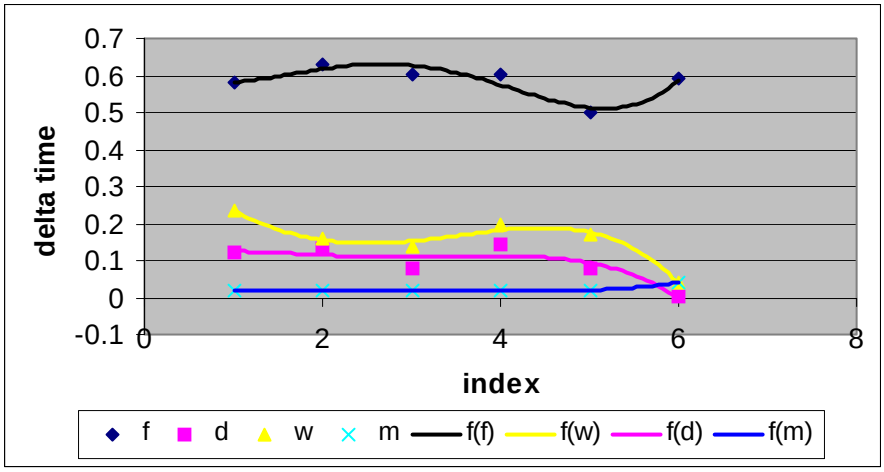
GROUP 2: Experiment 7 – 12



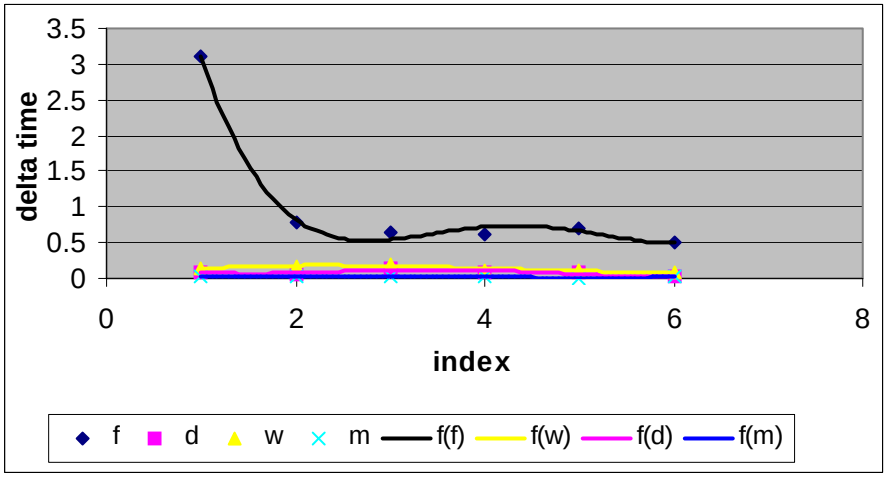
GROUP 3: Experiment 13 – 18



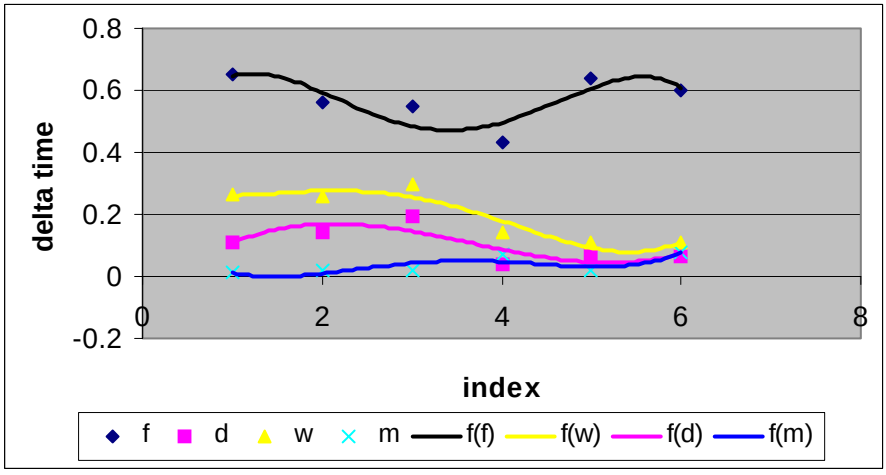
GROUP 4: Experiment 19 – 24



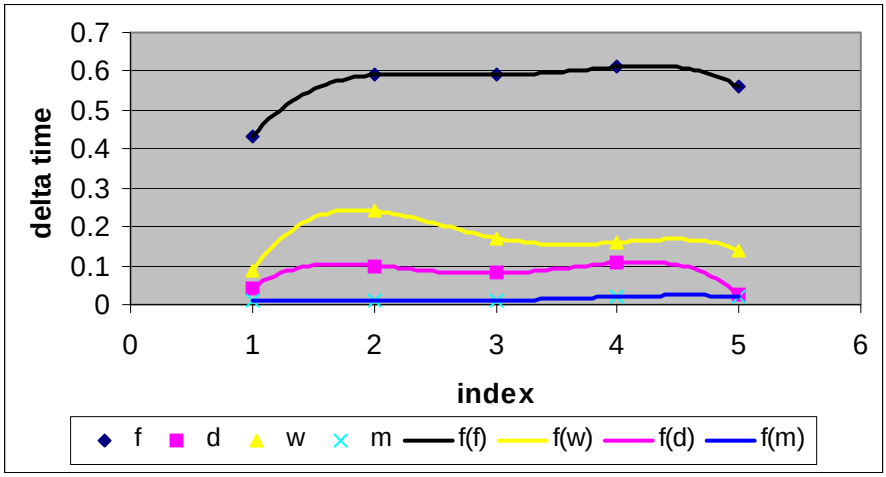
GROUP 5: Experiment 25 – 30



GROUP 6: Experiment 31 – 36



GROUP 7: Experiment 37 – 41



APPENDIX C4
Calculations for Experimental Results of Agent-Based Firewall

Table C2: Function of experimental results

Experiment Index	$f(x)$	$F = \int f(x)dx$
Group 1	$f_f(x)$ = -0.0113x4 + 0.1768x3 - 0.9557x2 + 2.0257x - 0.7757 R2 = 0.9907 $f_w(x)$ = 0.007x4 - 0.1196x3 + 0.7085x2 - 1.6539x + 1.3694 R2 = 0.9994 $f_d(x)$ = 0.0042x4 - 0.0692x3 + 0.3861x2 - 0.8213x + 0.5931 R2 = 0.7404 $f_m(x)$ = 0.0019x4 - 0.0271x3 + 0.1369x2 - 0.2885x + 0.245 R2 = 0.7005	2.746917 0.844083 0.39075 0.168292
Group 2	$f_f(x)$ = 0.0086x4 - 0.118x3 + 0.5459x2 - 0.9897x + 1.1725 R2 = 0.9651 $f_w(x)$ = -0.0093x4 + 0.1308x3 - 0.6265x2 + 1.2218x - 0.6832 R2 = 0.5792 $f_d(x)$ = 0.0069x4 - 0.0817x3 + 0.3071x2 - 0.386x + 0.1803 R2 = 0.7705 $f_m(x)$ = -0.0003x4 + 0.0043x3 - 0.0245x2 + 0.058x - 0.0275 R2 = 0.9898	2.836083 0.951333 0.434458 0.047292
Group 3	$f_f(x)$ = 0.0137x4 - 0.2201x3 + 1.216x2 - 2.6745x + 2.5443 R2 = 0.8257 $f_w(x)$ = -0.0115x4 + 0.1585x3 - 0.7395x2 + 1.3289x - 0.5906 R2 = 0.9995 $f_d(x)$ = -0.0111x4 + 0.1531x3 - 0.7138x2 + 1.2858x - 0.6205 R2 = 1 $f_m(x)$ = -0.0044x4 + 0.0588x3 - 0.2626x2 + 0.4587x - 0.2315 R2 = 0.9134	3.110542 0.737125 0.548958 0.244583
Group 4	$f_f(x)$ = 0.0038x4 - 0.0441x3 + 0.157x2 - 0.184x + 0.65 R2 = 0.8224 $f_w(x)$= -0.0011x4 + 0.0033x3 + 0.0452x2 - 0.2231x + 0.4113 R2 = 0.9776 $f_d(x)$ = -0.0013x4 + 0.0136x3 - 0.0465x2 + 0.053x + 0.1065 R2 = 0.8128 $f_m(x)$ = 0.0004x4 - 0.0049x3 + 0.0199x2 - 0.032x + 0.0367 R2 = 0.9952	2.913292 0.749458 0.509 0.085292

Table C2: cont.

Group 5	$f_f(x) = 0.0345x4 - 0.5986x3 + 3.7242x2 - 9.7742x + 9.7193$ $R2 = 0.9953$ $f_w(x) = -0.0004x4 + 0.0087x3 - 0.0693x2 + 0.1979x - 0.0004$ $R2 = 0.8791$ $f_d(x) = 0.0037x4 - 0.054x3 + 0.2581x2 - 0.4679x + 0.3532$ $R2 = 0.6576$ $f_m(x) = 0.0006x4 - 0.0081x3 + 0.0351x2 - 0.0591x + 0.0517$ $R2 = 0.8809$	4.29975 0.689375 0.345917 0.050375
Group 6	$f_f(x) = -0.0082x4 + 0.1113x3 - 0.4929x2 + 0.7715x + 0.2627$ $R2 = 0.6642$ $f_w(x) = 0.0029x4 - 0.0332x3 + 0.1084x2 - 0.1217x + 0.3041$ $R2 = 0.8894$ $f_d(x) = -2E-05x4 + 0.0088x3 - 0.0953x2 + 0.2863x - 0.0923$ $R2 = 0.6321$ $f_m(x) = 0.0031x4 - 0.0424x3 + 0.1948x2 - 0.3383x + 0.195$ $R2 = 0.7156$	2.772625 0.920417 0.536817 0.108917
Group 7	$f_f(x) = -0.0113x4 + 0.1429x3 - 0.6552x2 + 1.2946x - 0.34$ $R2 = 1$ $f_w(x) = -0.0146x4 + 0.1932x3 - 0.9072x2 + 1.744x - 0.93$ $R2 = 1$ $f_d(x) = -0.0116x4 + 0.1364x3 - 0.5672x2 + 0.977x - 0.4912$ $R2 = 1$ $f_m(x) = -0.0013x4 + 0.0142x3 - 0.0538x2 + 0.0808x - 0.03$ $R2 = 1$	2.32576 0.72752 0.345653 0.028827

APPENDIX D1
Table of External Parties for Zero-Configuration

Table D1: List of external parties

```
fsksm.utm.my 161.139.68.251 02:30.2
fsksm.utm.my 161.139.250.2 02:30.2
fsksm.utm.my 161.139.68.247 02:30.2
fsksm.utm.my 216.239.57.103 02:30.2
fsksm.utm.my 216.239.57.104 02:30.2
fsksm.utm.my 216.239.57.99 02:30.2

web.utm.my 161.139.18.91 02:40.1
www.utm.my 161.139.18.99 02:40.1

www.ukm.my 202.185.33.79 00:17.9

www.elsevier.com 129.35.76.177 01:15.5

www.gatra.com 64.156.138.148 02:56.8

www.google.com 63.150.131.40 00:30.1
www.google.com 64.233.189.104 00:30.1
www.google.com 64.233.161.147 00:30.1
www.google.com 64.233.161.104 00:30.1
www.google.com 66.94.229.254 00:30.1
www.google.com 66.102.7.104 00:30.1
www.google.com 66.102.9.104 00:30.1
www.google.com 66.102.9.99 00:30.1
www.google.com 66.102.11.104 00:30.1
www.google.com 66.102.11.99 00:30.1

www.kompas.com 64.203.71.11 02:20.6
www.kompas.com 64.203.71.51 02:20.6

www.rd.com 164.109.22.52 03:03.0
www.rd.com 164.109.22.155 03:03.0
```

APPENDIX D2
Experimental Results of Zero-Configuration

Table D2: Experimental result for opening www.fsksm.utm.my

Name of External Party	Δt_{oc}	$\Delta t_{transaction}$	**Cumulative Mean**	**Cumulative Std Dev**	**Life-time of canals**	$\frac{Size}{\Delta t_{transaction}}$	**Access Results**
www.fsksm.utm.my (Size 30 Kbytes)	00:02.4	00:22.4	00:22.4	00:22.4	01:29.6	8.64468	Success
	00:02.7	00:30.1	00:26.3	00:05.4	00:42.6	11.61574	Success
	00:00.7	00:30.7	00:27.7	00:04.6	00:41.6	11.84992	Success
	00:01.7	00:13.3	00:24.1	00:08.2	00:48.6	5.11420	Success
	00:09.1	00:31.3	00:25.5	00:07.8	**00:48.8**	12.05748	Success
	00:36.0	**01:55.8**	00:40.6	00:37.5	02:33.1	44.69444	**Denied**
	00:01.5	00:09.7	00:36.2	00:36.2	02:24.7	3.74576	Success
	00:09.4	00:35.3	00:36.1	00:33.5	02:16.6	13.61111	Success
	00:37.5	01:34.4	00:42.6	00:36.9	02:33.2	36.43248	Success
	00:03.3	00:05.9	00:38.9	00:36.7	02:28.8	2.28665	Success
	00:04.7	00:23.7	00:37.5	00:35.1	02:22.7	9.13002	Success
	00:18.8	01:07.1	00:40.0	00:34.5	02:23.5	25.90316	Success
	01:16.1	01:56.4	00:45.9	00:39.3	02:43.7	44.91898	Success
	00:02.4	00:09.9	00:43.3	00:38.9	02:40.1	3.82986	Success
	00:09.1	00:36.7	00:42.9	00:37.6	02:35.5	14.15123	Success
	00:36.7	01:35.0	00:46.1	00:38.5	02:41.8	36.64853	Success
	00:01.7	00:09.4	00:44.0	00:38.4	02:39.1	3.62461	Success
	00:09.2	00:36.8	00:43.6	00:37.3	02:35.3	14.19252	Success
	00:02.3	00:10.5	00:41.8	00:37.0	02:32.8	4.05748	Success
	00:08.5	00:31.4	00:41.3	00:36.1	02:29.6	12.09992	Success
	00:02.4	00:24.6	00:40.5	00:35.4	02:26.6	9.49306	Success
	00:02.2	00:24.6	00:39.8	00:34.7	02:23.8	9.48225	Success
	00:01.8	00:10.5	00:38.5	00:34.4	02:21.8	4.05517	Success
	00:08.6	00:31.3	00:38.2	00:33.7	02:19.3	12.07446	Success
	00:36.8	02:00.0	00:41.5	00:36.8	02:32.0	46.31289	Success
	00:34.2	01:44.7	00:43.9	00:38.2	02:38.4	40.40085	Success
	00:02.6	00:24.7	00:43.2	00:37.6	02:36.0	9.51003	Success
	00:01.5	00:10.4	00:42.0	00:37.4	02:34.3	4.02855	Success
	00:02.4	00:12.9	00:41.0	00:37.1	02:32.4	4.99074	Success
	00:17.6	00:59.0	00:41.6	00:36.6	02:31.5	22.74460	Success
	00:00.3	00:06.9	00:40.5	00:36.6	02:30.2	2.66898	Success
Average	00:12.4	00:40.5	-	-	-	15.62485	-
Standard Deviation	00:17.2	00:36.6	-	-	-	14.10173	-
Denial of Service	1 of 31 transactions						

Table D3: Experimental result for opening www.utm.my

Name of External Party	Δt_{oc}	$\Delta t_{transaction}$	**Cumulative Mean**	**Cumulative Std Dev**	**Life-time of canals**	$\frac{Size}{\Delta t_{transaction}}$	**Access Results**
www.utm.my (Size 9 Kbytes)	00:02.2	00:18.7	00:18.7	00:18.7	01:14.8	24.04064	Success
	00:02.2	00:10.5	00:14.6	00:05.8	00:32.0	13.47608	Success
	00:08.9	00:28.9	00:19.4	00:09.3	**00:47.1**	37.22094	Success
	00:36.3	**01:54.5**	00:43.2	00:48.2	03:07.6	147.25309	**Denied**
	00:02.3	00:10.5	00:36.6	00:44.2	02:49.2	13.49537	Success
	00:08.5	00:34.0	00:36.2	00:39.5	02:34.8	43.71914	Success
	00:36.4	01:53.6	00:47.2	00:46.5	03:06.6	146.05838	Success
	00:35.7	01:31.9	00:52.8	00:45.8	03:10.3	118.22402	Success
	00:03.0	00:15.8	00:48.7	00:44.6	03:02.5	20.28035	Success
	00:00.5	00:11.3	00:45.0	00:43.7	02:56.0	14.56147	Success
	00:09.2	00:28.2	00:43.4	00:41.8	02:48.7	36.28601	Success
	00:24.0	01:10.7	00:45.7	00:40.6	02:47.5	90.96836	Success
	01:13.5	01:54.5	00:51.0	00:43.3	03:00.8	147.19264	Success
	00:09.0	00:23.9	00:49.1	00:42.2	02:55.7	30.73174	Success
	00:05.2	00:12.8	00:46.7	00:41.7	02:51.9	16.48405	Success
	00:05.4	00:28.1	00:45.5	00:40.6	02:47.3	36.08410	Success
	00:35.9	01:53.5	00:49.5	00:42.6	02:57.4	145.98637	Success
	00:18.4	01:04.8	00:50.3	00:41.5	02:54.9	83.36034	Success
	00:01.2	00:09.6	00:48.2	00:41.4	02:52.4	12.40612	Success
	00:04.1	00:18.1	00:46.7	00:40.9	02:49.3	23.28061	Success
	00:02.8	00:18.7	00:45.4	00:40.3	02:46.2	24.03164	Success
	00:01.6	00:10.6	00:43.8	00:40.0	02:43.8	13.59825	Success
	00:00.6	00:46.2	00:43.9	00:39.1	02:41.2	59.43287	Success
	00:02.0	00:16.0	00:42.7	00:38.7	02:38.7	20.61214	Success
	00:01.9	00:10.4	00:41.4	00:38.4	02:36.6	13.35520	Success
	00:09.1	00:43.7	00:41.5	00:37.6	02:34.4	56.21271	Success
	00:10.0	00:40.3	00:41.5	00:36.9	02:32.1	51.87114	Success
	00:03.6	02:17.1	00:44.9	00:40.5	02:46.3	176.35031	Success
	00:02.5	00:18.5	00:44.0	00:40.0	02:44.1	23.83102	Success
	00:02.3	00:10.6	00:42.9	00:39.8	02:42.3	13.59439	Success
	00:08.5	00:31.2	00:42.5	00:39.2	02:40.1	40.11960	Success
Average	00:11.8	00:42.5	-	-	-	54.64900	-
Std Deviation	0.000188	00:39.2	-	-	-	50.40374	-
Denial of Service	1 of 31 transactions						

Table D4: Experimental result for opening www.ukm.my

Name of External Party	Δt_{oc}	$\Delta t_{transaction}$	**Cumulative Mean**	**Cumulative Std Dev**	**Life-time of canals**	$\frac{Size}{\Delta t_{transaction}}$	**Access Results**
www.ukm.my (Size 26 Kbytes)	00:00.5	00:04.6	00:04.6	00:04.6	00:18.2	2.02947	Success
	00:00.9	00:10.8	00:07.7	00:04.4	00:21.0	4.82906	Success
	00:00.8	00:10.3	00:08.6	00:03.5	00:19.0	4.56998	Success
	00:00.7	00:04.6	00:07.6	00:03.5	00:18.0	2.03036	Success
	00:02.6	00:12.7	00:08.6	00:03.8	00:19.9	5.63925	Success
	00:03.0	00:19.0	00:10.3	00:05.4	00:26.6	8.43616	Success
	00:00.2	00:04.5	00:09.5	00:05.4	00:25.7	2.00855	Success
	00:00.9	00:04.2	00:08.8	00:05.3	00:24.9	1.86254	Success
	00:00.7	00:04.5	00:08.3	00:05.2	00:23.9	2.00009	Success
	00:00.7	00:10.6	00:08.6	00:05.0	00:23.4	4.70397	Success
	00:00.2	00:10.9	00:08.8	00:04.8	00:23.0	4.87135	Success
	00:00.6	00:04.5	00:08.4	00:04.7	00:22.5	2.02235	Success
	00:00.7	00:04.5	00:08.1	00:04.6	00:22.0	2.02324	Success
	00:00.5	00:04.5	00:07.9	00:04.5	00:21.5	2.00276	Success
	00:01.0	00:11.0	00:08.1	00:04.5	00:21.4	4.90518	Success
	00:00.2	00:04.5	00:07.9	00:04.4	00:21.0	2.01300	Success
	00:00.5	00:05.0	00:07.7	00:04.3	00:20.6	2.23736	Success
	00:00.3	00:10.5	00:07.8	00:04.2	00:20.5	4.66569	Success
	00:00.9	00:07.9	00:07.8	00:04.1	00:20.2	3.51407	Success
	00:00.7	00:04.5	00:07.7	00:04.1	00:19.9	2.01077	Success
	00:00.7	00:04.5	00:07.5	00:04.0	00:19.6	2.00677	Success
	00:00.3	00:04.5	00:07.4	00:04.0	00:19.3	2.01790	Success
	00:00.8	00:04.6	00:07.3	00:03.9	00:19.1	2.03659	Success
	00:00.6	00:04.8	00:07.2	00:03.9	00:18.8	2.14432	Success
	00:00.7	00:06.0	00:07.1	00:03.8	00:18.5	2.69186	Success
	00:00.0	00:04.6	00:07.0	00:03.8	00:18.3	2.03259	Success
	00:00.1	00:04.1	00:06.9	00:03.7	00:18.1	1.84295	Success
	00:00.5	00:04.2	00:06.8	00:03.7	00:17.9	1.85452	Success
	00:00.4	00:10.5	00:06.9	00:03.7	00:18.0	4.68705	Success
	00:00.7	00:04.1	00:06.9	00:03.7	00:17.9	1.83093	Success
	00:00.4	00:10.3	00:07.0	00:03.7	00:17.9	4.57131	Success
Average	00:00.7	00:07.0	-	-	-	3.09974	-
Std Deviation	7.13E-06	00:03.7	-	-	-	1.62958	-
Denial of Service	0 of 31 transactions						

Table D5: Experimental result for opening www.elsevier.com

Name of External Party	Δt_{oc}	$\Delta t_{transaction}$	**Cumulative Mean**	**Cumulative Std Dev**	**Life-time of canals**	$\frac{Size}{\Delta t_{transaction}}$	**Access Results**
www.elsevier.com (Size 146 Kbytes)	00:00.3	00:46.5	00:46.5	00:46.5	03:06.1	3.68817	Success
	00:00.2	00:44.8	00:45.7	00:01.2	**00:49.3**	3.55411	Success
	00:01.5	**00:49.8**	00:47.1	00:02.5	00:54.7	3.94938	**Denied**
	00:00.2	00:46.2	00:46.8	00:02.1	**00:53.2**	3.66042	Success
	00:02.6	**01:12.5**	00:52.0	00:11.6	01:26.8	5.74526	**Denied**
	00:00.4	00:47.8	00:51.3	00:10.5	01:22.8	3.78924	Success
	00:03.1	00:59.2	00:52.4	00:10.1	01:22.6	4.69408	Success
	00:00.9	00:44.7	00:51.4	00:09.7	01:20.6	3.54270	Success
	00:00.6	00:46.7	00:50.9	00:09.2	01:18.6	3.70410	Success
	00:00.3	00:44.6	00:50.3	00:08.9	01:17.0	3.53374	Success
	00:00.6	00:48.0	00:50.1	00:08.5	01:15.5	3.80858	Success
Average	00:01.0	00:50.1	-	-	-	3.96998	-
Std Deviation	1.18E-05	00:08.5	-	-	-	0.67282	-
Denial of Service	2 of 11 transactions						

Table D6: Experimental result for opening www.gatra.com

Name of External Party	Δt_{oc}	$\Delta t_{transaction}$	**Cumulative Mean**	**Cumulative Std Dev**	**Life-time of canals**	$\frac{Size}{\Delta t_{transaction}}$	**Access Results**
www.gatra.com (Size 64 Kbytes)	00:02.9	00:33.5	00:33.5	00:33.5	02:14.0	6.05650	Success
	00:00.5	00:40.8	00:37.2	00:05.2	00:52.7	7.38263	Success
	00:00.4	00:43.1	00:39.1	00:05.0	**00:54.2**	7.78863	Success
	00:00.7	**01:32.4**	00:52.5	00:27.0	02:13.4	16.71893	**Denied**
	00:02.0	00:46.7	00:51.3	00:23.5	02:01.8	8.43660	Success
	00:02.9	01:52.7	01:01.5	00:32.7	02:39.7	20.38610	Success
	00:02.4	01:57.3	01:09.5	00:36.6	02:59.2	21.21944	Success
	00:02.7	00:33.4	01:05.0	00:36.2	02:53.6	6.03172	Success
	00:00.1	01:32.6	01:08.1	00:35.1	02:53.3	16.74226	Success
	00:02.5	01:57.4	01:13.0	00:36.6	03:02.7	21.22667	Success
	00:02.5	00:52.5	01:11.1	00:35.2	02:56.8	9.49797	Success
Average	00:01.8	01:11.1	-	-	-	12.86250	-
Std Deviation	1.29E-05	00:35.2	-	-	-	6.37173	-
Denial of Service	1 of 11 transactions						

Table D7: Experimental result for opening www.google.com

Name of External Party	Δt_{oc}	$\Delta t_{transaction}$	**Cumulative Mean**	**Cumulative Std Dev**	**Life-time of canals**	$\frac{Size}{\Delta t_{transaction}}$	**Access Results**
www.google.com (Size 10 Kbytes)	00:00.1	00:12.4	00:12.4	00:12.4	00:49.7	14.37500	Success
	00:02.3	00:19.3	00:15.8	00:04.8	00:30.4	22.29398	Success
	00:02.4	00:25.1	00:18.9	00:06.4	00:38.0	29.07292	Success
	00:02.7	00:19.3	00:19.0	00:05.2	00:34.6	22.34722	Success
	00:03.1	00:19.2	00:19.1	00:04.5	00:32.6	22.27431	Success
	00:02.2	00:19.2	00:19.1	00:04.0	00:31.2	22.27199	Success
	00:02.7	00:19.1	00:19.1	00:03.7	00:30.1	22.16319	Success
	00:02.7	00:19.2	00:19.1	00:03.4	00:29.3	22.24421	Success
	00:02.7	00:19.3	00:19.1	00:03.2	00:28.7	22.36111	Success
	00:02.8	00:25.6	00:19.8	00:03.6	00:30.7	29.67824	Success
	00:02.7	00:19.2	00:19.7	00:03.5	00:30.1	22.26389	Success
Average	00:02.4	00:12.4	-	-	-	22.84964	-
Std Deviation	9.14E-06	00:19.3	-	-	-	3.99780	-
Denial of Service	0 of 11 transactions						

Table D8: Experimental result for opening www.kompas.com

Name of External Party	Δt_{oc}	$\Delta t_{transaction}$	**Cumulative Mean**	**Cumulative Std Dev**	**Life-time of canals**	$\frac{Size}{\Delta t_{transaction}}$	**Access Results**
www.kompas.com (Size 43 Kbytes)	00:06.6	01:30.0	01:30.0	01:30.0	06:00.1	24.23342	Success
	00:11.1	00:54.0	01:12.0	00:25.4	02:28.4	14.54753	Success
	00:01.1	00:27.1	00:57.1	00:31.6	02:31.8	7.30163	Success
	00:01.0	00:34.4	00:51.4	00:28.1	02:15.8	9.26814	Success
	00:01.0	00:30.1	00:47.1	00:26.2	02:05.7	8.09566	Success
	00:10.9	01:42.7	00:56.4	00:32.6	02:34.2	27.64185	Success
	00:02.9	01:36.3	01:02.1	00:33.4	02:42.2	25.91301	Success
	00:03.0	00:33.8	00:58.6	00:32.5	02:35.9	9.10557	Success
	00:01.0	00:28.8	00:55.3	00:31.9	02:31.1	7.75086	Success
	00:02.4	00:33.9	00:53.1	00:30.9	02:25.7	9.11445	Success
	00:02.4	00:40.3	00:52.0	00:29.5	02:20.6	10.85164	Success
Average	00:03.9	00:52.0	-	-	-	13.98398	-
Std Deviation	4.44E-05	00:29.5	-	-	-	7.94951	-
Denial of Service	0 of 11 transactions						

APPENDIX D2
Experimental Results of Zero-Configuration

Table D7: Experimental result for opening www.rd.com

Name of External Party	Δt_{oc}	$\Delta t_{transaction}$	**Cumulative Mean**	**Cumulative Std Dev**	**Life-time of canals**	$\frac{Size}{\Delta t_{transaction}}$	**Access Results**
www.rd.com (Size 138 Kbytes)	00:20.6	01:42.1	01:42.1	01:42.1	06:48.3	8.56104	Success
	00:02.4	00:41.5	01:11.8	00:42.8	03:20.2	3.48422	Success
	00:00.7	00:24.8	00:56.1	00:40.6	02:58.1	2.08090	Success
	00:09.3	00:44.1	00:53.1	00:33.7	02:34.3	3.69951	Success
	00:09.7	00:58.5	00:54.2	00:29.3	02:22.1	4.90296	Success
	00:01.2	01:25.3	00:59.4	00:29.1	02:26.8	7.15269	Success
	00:02.6	01:23.9	01:02.9	00:28.2	02:27.4	7.03897	Success
	00:02.5	02:02.9	01:10.4	00:33.6	02:51.3	10.31015	Success
	00:02.8	02:02.2	01:16.1	00:35.9	03:03.8	10.24666	Success
	00:01.2	01:57.5	01:20.3	00:36.3	03:09.1	9.85390	Success
	00:02.7	01:08.0	01:19.2	00:34.6	03:03.0	5.70535	Success
Average	00:05.1	01:19.2	-	-	-	6.63967	-
Std Deviation	6.95E-05	00:34.6	-	-	-	2.90175	-
Denial of Service	0 of 11 transactions						

APPENDIX D1
Table of External Parties for Zero-Configuration

Table D1: List of external parties

```
fsksm.utm.my 161.139.68.251 02:30.2
fsksm.utm.my 161.139.250.2 02:30.2
fsksm.utm.my 161.139.68.247 02:30.2
fsksm.utm.my 216.239.57.103 02:30.2
fsksm.utm.my 216.239.57.104 02:30.2
fsksm.utm.my 216.239.57.99 02:30.2

web.utm.my 161.139.18.91 02:40.1
www.utm.my 161.139.18.99 02:40.1

www.ukm.my 202.185.33.79 00:17.9

www.elsevier.com 129.35.76.177 01:15.5

www.gatra.com 64.156.138.148 02:56.8

www.google.com 63.150.131.40 00:30.1
www.google.com 64.233.189.104 00:30.1
www.google.com 64.233.161.147 00:30.1
www.google.com 64.233.161.104 00:30.1
www.google.com 66.94.229.254 00:30.1
www.google.com 66.102.7.104 00:30.1
www.google.com 66.102.9.104 00:30.1
www.google.com 66.102.9.99 00:30.1
www.google.com 66.102.11.104 00:30.1
www.google.com 66.102.11.99 00:30.1

www.kompas.com 64.203.71.11 02:20.6
www.kompas.com 64.203.71.51 02:20.6

www.rd.com 164.109.22.52 03:03.0
www.rd.com 164.109.22.155 03:03.0
```

APPENDIX D2
Experimental Results of Zero-Configuration

Table D2: Experimental result for opening www.fsksm.utm.my

Name of External Party	Δt_{oc}	$\Delta t_{transaction}$	**Cumulative Mean**	**Cumulative Std Dev**	**Life-time of canals**	**Access Results**
www.fsksm.utm.my	00:02.4	00:22.4	00:22.4	00:22.4	01:29.6	Success
	00:02.7	00:30.1	00:26.3	00:05.4	00:42.6	Success
	00:00.7	00:30.7	00:27.7	00:04.6	00:41.6	Success
	00:01.7	00:13.3	00:24.1	00:08.2	00:48.6	Success
	00:09.1	00:31.3	00:25.5	00:07.8	**00:48.8**	Success
	00:36.0	**01:55.8**	00:40.6	00:37.5	02:33.1	**Denied**
	00:01.5	00:09.7	00:36.2	00:36.2	02:24.7	Success
	00:09.4	00:35.3	00:36.1	00:33.5	02:16.6	Success
	00:37.5	01:34.4	00:42.6	00:36.9	02:33.2	Success
	00:03.3	00:05.9	00:38.9	00:36.7	02:28.8	Success
	00:04.7	00:23.7	00:37.5	00:35.1	02:22.7	Success
	00:18.8	01:07.1	00:40.0	00:34.5	02:23.5	Success
	01:16.1	01:56.4	00:45.9	00:39.3	02:43.7	Success
	00:02.4	00:09.9	00:43.3	00:38.9	02:40.1	Success
	00:09.1	00:36.7	00:42.9	00:37.6	02:35.5	Success
	00:36.7	01:35.0	00:46.1	00:38.5	02:41.8	Success
	00:01.7	00:09.4	00:44.0	00:38.4	02:39.1	Success
	00:09.2	00:36.8	00:43.6	00:37.3	02:35.3	Success
	00:02.3	00:10.5	00:41.8	00:37.0	02:32.8	Success
	00:08.5	00:31.4	00:41.3	00:36.1	02:29.6	Success
	00:02.4	00:24.6	00:40.5	00:35.4	02:26.6	Success
	00:02.2	00:24.6	00:39.8	00:34.7	02:23.8	Success
	00:01.8	00:10.5	00:38.5	00:34.4	02:21.8	Success
	00:08.6	00:31.3	00:38.2	00:33.7	02:19.3	Success
	00:36.8	02:00.0	00:41.5	00:36.8	02:32.0	Success
	00:34.2	01:44.7	00:43.9	00:38.2	02:38.4	Success
	00:02.6	00:24.7	00:43.2	00:37.6	02:36.0	Success
	00:01.5	00:10.4	00:42.0	00:37.4	02:34.3	Success
	00:02.4	00:12.9	00:41.0	00:37.1	02:32.4	Success
	00:17.6	00:59.0	00:41.6	00:36.6	02:31.5	Success
	00:00.3	00:06.9	00:40.5	00:36.6	02:30.2	Success
Average	00:12.4	00:40.5	-	-	-	-
Standard Deviation	00:17.2	00:36.6	-	-	-	-
Denial of Service	1 of 31 transactions					

Table D3: Experimental result for opening www.utm.my

Name of External Party	Δt_{oc}	$\Delta t_{transaction}$	**Cumulative Mean**	**Cumulative Std Dev**	**Life-time of canals**	**Access Results**
www.utm.my	00:02.2	00:18.7	00:18.7	00:18.7	01:14.8	Success
	00:02.2	00:10.5	00:14.6	00:05.8	00:32.0	Success
	00:08.9	00:28.9	00:19.4	00:09.3	**00:47.1**	Success
	00:36.3	**01:54.5**	00:43.2	00:48.2	03:07.6	**Denied**
	00:02.3	00:10.5	00:36.6	00:44.2	02:49.2	Success
	00:08.5	00:34.0	00:36.2	00:39.5	02:34.8	Success
	00:36.4	01:53.6	00:47.2	00:46.5	03:06.6	Success
	00:35.7	01:31.9	00:52.8	00:45.8	03:10.3	Success
	00:03.0	00:15.8	00:48.7	00:44.6	03:02.5	Success
	00:00.5	00:11.3	00:45.0	00:43.7	02:56.0	Success
	00:09.2	00:28.2	00:43.4	00:41.8	02:48.7	Success
	00:24.0	01:10.7	00:45.7	00:40.6	02:47.5	Success
	01:13.5	01:54.5	00:51.0	00:43.3	03:00.8	Success
	00:09.0	00:23.9	00:49.1	00:42.2	02:55.7	Success
	00:05.2	00:12.8	00:46.7	00:41.7	02:51.9	Success
	00:05.4	00:28.1	00:45.5	00:40.6	02:47.3	Success
	00:35.9	01:53.5	00:49.5	00:42.6	02:57.4	Success
	00:18.4	01:04.8	00:50.3	00:41.5	02:54.9	Success
	00:01.2	00:09.6	00:48.2	00:41.4	02:52.4	Success
	00:04.1	00:18.1	00:46.7	00:40.9	02:49.3	Success
	00:02.8	00:18.7	00:45.4	00:40.3	02:46.2	Success
	00:01.6	00:10.6	00:43.8	00:40.0	02:43.8	Success
	00:00.6	00:46.2	00:43.9	00:39.1	02:41.2	Success
	00:02.0	00:16.0	00:42.7	00:38.7	02:38.7	Success
	00:01.9	00:10.4	00:41.4	00:38.4	02:36.6	Success
	00:09.1	00:43.7	00:41.5	00:37.6	02:34.4	Success
	00:10.0	00:40.3	00:41.5	00:36.9	02:32.1	Success
	00:03.6	02:17.1	00:44.9	00:40.5	02:46.3	Success
	00:02.5	00:18.5	00:44.0	00:40.0	02:44.1	Success
	00:02.3	00:10.6	00:42.9	00:39.8	02:42.3	Success
	00:08.5	00:31.2	00:42.5	00:39.2	02:40.1	Success
Average	00:11.8	00:42.5	-	-	-	-
Std Deviation	0.000188	00:39.2	-	-	-	-
Denial of Service	1 of 31 transactions					

Table D4: Experimental result for opening www.ukm.my

Name of External Party	Δt_{oc}	$\Delta t_{transaction}$	**Cumulative Mean**	**Cumulative Std Dev**	**Life-time of canals**	**Access Results**
www.ukm.my	00:00.5	00:04.6	00:04.6	00:04.6	00:18.2	Success
	00:00.9	00:10.8	00:07.7	00:04.4	00:21.0	Success
	00:00.8	00:10.3	00:08.6	00:03.5	00:19.0	Success
	00:00.7	00:04.6	00:07.6	00:03.5	00:18.0	Success
	00:02.6	00:12.7	00:08.6	00:03.8	00:19.9	Success
	00:03.0	00:19.0	00:10.3	00:05.4	00:26.6	Success
	00:00.2	00:04.5	00:09.5	00:05.4	00:25.7	Success
	00:00.9	00:04.2	00:08.8	00:05.3	00:24.9	Success
	00:00.7	00:04.5	00:08.3	00:05.2	00:23.9	Success
	00:00.7	00:10.6	00:08.6	00:05.0	00:23.4	Success
	00:00.2	00:10.9	00:08.8	00:04.8	00:23.0	Success
	00:00.6	00:04.5	00:08.4	00:04.7	00:22.5	Success
	00:00.7	00:04.5	00:08.1	00:04.6	00:22.0	Success
	00:00.5	00:04.5	00:07.9	00:04.5	00:21.5	Success
	00:01.0	00:11.0	00:08.1	00:04.5	00:21.4	Success
	00:00.2	00:04.5	00:07.9	00:04.4	00:21.0	Success
	00:00.5	00:05.0	00:07.7	00:04.3	00:20.6	Success
	00:00.3	00:10.5	00:07.8	00:04.2	00:20.5	Success
	00:00.9	00:07.9	00:07.8	00:04.1	00:20.2	Success
	00:00.7	00:04.5	00:07.7	00:04.1	00:19.9	Success
	00:00.7	00:04.5	00:07.5	00:04.0	00:19.6	Success
	00:00.3	00:04.5	00:07.4	00:04.0	00:19.3	Success
	00:00.8	00:04.6	00:07.3	00:03.9	00:19.1	Success
	00:00.6	00:04.8	00:07.2	00:03.9	00:18.8	Success
	00:00.7	00:06.0	00:07.1	00:03.8	00:18.5	Success
	00:00.0	00:04.6	00:07.0	00:03.8	00:18.3	Success
	00:00.1	00:04.1	00:06.9	00:03.7	00:18.1	Success
	00:00.5	00:04.2	00:06.8	00:03.7	00:17.9	Success
	00:00.4	00:10.5	00:06.9	00:03.7	00:18.0	Success
	00:00.7	00:04.1	00:06.9	00:03.7	00:17.9	Success
	00:00.4	00:10.3	00:07.0	00:03.7	00:17.9	Success
Average	00:00.7	00:07.0	-	-	-	-
Std Deviation	7.13E-06	00:03.7	-	-	-	-
Denial of Service	0 of 31 transactions					

Table D5: Experimental result for opening www.elsevier.com

Name of External Party	Δt_{oc}	$\Delta t_{transaction}$	**Cumulative Mean**	**Cumulative Std Dev**	**Life-time of canals**	**Access Results**
www.elsevier.com	00:00.3	00:46.5	00:46.5	00:46.5	03:06.1	Success
	00:00.2	00:44.8	00:45.7	00:01.2	**00:49.3**	Success
	00:01.5	**00:49.8**	00:47.1	00:02.5	00:54.7	**Denied**
	00:00.2	00:46.2	00:46.8	00:02.1	**00:53.2**	Success
	00:02.6	**01:12.5**	00:52.0	00:11.6	01:26.8	**Denied**
	00:00.4	00:47.8	00:51.3	00:10.5	01:22.8	Success
	00:03.1	00:59.2	00:52.4	00:10.1	01:22.6	Success
	00:00.9	00:44.7	00:51.4	00:09.7	01:20.6	Success
	00:00.6	00:46.7	00:50.9	00:09.2	01:18.6	Success
	00:00.3	00:44.6	00:50.3	00:08.9	01:17.0	Success
	00:00.6	00:48.0	00:50.1	00:08.5	01:15.5	Success
Average	00:01.0	00:50.1	-	-	-	-
Std Deviation	1.18E-05	00:08.5	-	-	-	-
Denial of Service	2 of 11 transactions					

Table D6: Experimental result for opening www.gatra.com

Name of External Party	Δt_{oc}	$\Delta t_{transaction}$	**Cumulative Mean**	**Cumulative Std Dev**	**Life-time of canals**	**Access Results**
www.gatra.com	00:02.9	00:33.5	00:33.5	00:33.5	02:14.0	Success
	00:00.5	00:40.8	00:37.2	00:05.2	00:52.7	Success
	00:00.4	00:43.1	00:39.1	00:05.0	**00:54.2**	Success
	00:00.7	**01:32.4**	00:52.5	00:27.0	02:13.4	**Denied**
	00:02.0	00:46.7	00:51.3	00:23.5	02:01.8	Success
	00:02.9	01:52.7	01:01.5	00:32.7	02:39.7	Success
	00:02.4	01:57.3	01:09.5	00:36.6	02:59.2	Success
	00:02.7	00:33.4	01:05.0	00:36.2	02:53.6	Success
	00:00.1	01:32.6	01:08.1	00:35.1	02:53.3	Success
	00:02.5	01:57.4	01:13.0	00:36.6	03:02.7	Success
	00:02.5	00:52.5	01:11.1	00:35.2	02:56.8	Success
Average	00:01.8	01:11.1	-	-	-	-
Std Deviation	1.29E-05	00:35.2	-	-	-	-
Denial of Service	1 of 11 transactions					

Table D7: Experimental result for opening www.google.com

Name of External Party	Δt_{oc}	$\Delta t_{transaction}$	**Cumulative Mean**	**Cumulative Std Dev**	**Life-time of canals**	**Access Results**
www.google.com	00:00.1	00:12.4	00:12.4	00:12.4	00:49.7	Success
	00:02.3	00:19.3	00:15.8	00:04.8	00:30.4	Success
	00:02.4	00:25.1	00:18.9	00:06.4	00:38.0	Success
	00:02.7	00:19.3	00:19.0	00:05.2	00:34.6	Success
	00:03.1	00:19.2	00:19.1	00:04.5	00:32.6	Success
	00:02.2	00:19.2	00:19.1	00:04.0	00:31.2	Success
	00:02.7	00:19.1	00:19.1	00:03.7	00:30.1	Success
	00:02.7	00:19.2	00:19.1	00:03.4	00:29.3	Success
	00:02.7	00:19.3	00:19.1	00:03.2	00:28.7	Success
	00:02.8	00:25.6	00:19.8	00:03.6	00:30.7	Success
	00:02.7	00:19.2	00:19.7	00:03.5	00:30.1	Success
Average	00:02.4	00:12.4	-	-	-	-
Std Deviation	9.14E-06	00:19.3	-	-	-	-
Denial of Service	0 of 11 transactions					

Table D8: Experimental result for opening www.kompas.com

Name of External Party	Δt_{oc}	$\Delta t_{transaction}$	**Cumulative Mean**	**Cumulative Std Dev**	**Life-time of canals**	**Access Results**
www.kompas.com	00:06.6	01:30.0	01:30.0	01:30.0	06:00.1	Success
	00:11.1	00:54.0	01:12.0	00:25.4	02:28.4	Success
	00:01.1	00:27.1	00:57.1	00:31.6	02:31.8	Success
	00:01.0	00:34.4	00:51.4	00:28.1	02:15.8	Success
	00:01.0	00:30.1	00:47.1	00:26.2	02:05.7	Success
	00:10.9	01:42.7	00:56.4	00:32.6	02:34.2	Success
	00:02.9	01:36.3	01:02.1	00:33.4	02:42.2	Success
	00:03.0	00:33.8	00:58.6	00:32.5	02:35.9	Success
	00:01.0	00:28.8	00:55.3	00:31.9	02:31.1	Success
	00:02.4	00:33.9	00:53.1	00:30.9	02:25.7	Success
	00:02.4	00:40.3	00:52.0	00:29.5	02:20.6	Success
Average	00:03.9	00:52.0	-	-	-	-
Std Deviation	4.44E-05	00:29.5	-	-	-	-
Denial of Service	0 of 11 transactions					

Table D7: Experimental result for opening www.rd.com

Name of External Party	Δt_{oc}	$\Delta t_{transaction}$	**Cumulative Mean**	**Cumulative Std Dev**	**Life-time of canals**	**Access Results**
www.rd.com	00:20.6	01:42.1	01:42.1	01:42.1	06:48.3	Success
	00:02.4	00:41.5	01:11.8	00:42.8	03:20.2	Success
	00:00.7	00:24.8	00:56.1	00:40.6	02:58.1	Success
	00:09.3	00:44.1	00:53.1	00:33.7	02:34.3	Success
	00:09.7	00:58.5	00:54.2	00:29.3	02:22.1	Success
	00:01.2	01:25.3	00:59.4	00:29.1	02:26.8	Success
	00:02.6	01:23.9	01:02.9	00:28.2	02:27.4	Success
	00:02.5	02:02.9	01:10.4	00:33.6	02:51.3	Success
	00:02.8	02:02.2	01:16.1	00:35.9	03:03.8	Success
	00:01.2	01:57.5	01:20.3	00:36.3	03:09.1	Success
	00:02.7	01:08.0	01:19.2	00:34.6	03:03.0	Success
Average	00:05.1	01:19.2	-	-	-	-
Std Deviation	6.95E-05	00:34.6	-	-	-	-
Denial of Service	0 of 11 transactions					

APPENDIX D3
Comparison of Processing Time and Number of Packets of Zero-Configuration

Table D8: Processing time versus number of packets

No.	**Number of packets**	Δt **(min: seconds)**	No.	**Number of packets**	Δt **(min: seconds)**
1	1	00:00.0	31	137	00:09.0
2	2	00:00.0	32	166	00:11.0
3	2	00:01.0	33	176	00:09.0
4	3	00:00.0	34	178	00:09.0
5	3	00:00.0	35	194	00:13.0
6	3	00:00.0	36	205	00:12.0
7	6	00:01.0	37	232	00:13.0
8	8	00:00.0	38	249	00:18.0
9	8	00:01.0	39	293	00:17.0
10	8	00:01.0	40	308	00:20.0
11	9	00:00.0	41	313	00:20.0
12	14	00:01.0	42	321	00:23.0
13	16	00:01.0	43	329	00:18.0
14	18	00:01.0	44	335	00:18.0
15	19	00:02.0	45	349	00:18.0
16	19	00:03.0	46	418	00:30.0
17	20	00:02.0	47	455	00:27.0
18	22	00:01.0	48	484	00:29.0
19	29	00:02.0	49	487	00:29.0
20	35	00:02.0	50	524	00:28.0
21	43	00:03.0	51	561	00:38.0
22	43	00:03.0	52	569	00:40.0
23	45	00:03.0	53	618	00:39.0
24	66	00:05.0	54	651	00:38.0
25	71	00:05.0	55	730	00:35.0
26	83	00:06.0	56	887	01:03.0
27	94	00:07.0	57	942	01:10.0
28	107	00:06.0	58	1065	01:03.0
29	121	00:06.0	59	1072	01:07.0
30	123	00:07.0	60	1326	01:23.0

APPENDIX D4
Foreground Algorithm of Zero-Configuration

```
#!/bin/sh
# FIREWALL BASED ON ZERO CONFIGURATION #
# eth0 = EXTERNAL NETWORK
# eth1 = INTERNAL NETWORK
# ------------------
# 1. Initialization
# ------------------
INTERNAL_SUBNET="24.4.74"
> /PROJECT/TEMP/PROXY_BUFF_INT
> /PROJECT/TEMP/PROXY_BUFF_EXT
> /PROJECT/TEMP/IP_SENDER
> /PROJECT/TEMP/IP_ADDR_SENDER
> /PROJECT/TEMP/IP_ADDR_RECEIVER
> /PROJECT/TEMP/TABLEIP
> /PROJECT/TEMP/HISTORY_ZERO
> /PROJECT/TEMP/DELTATIME
> /PROJECT/TEMP/TIMEOUTPUT
> /PROJECT/TEMP/LOG
cd /PROJECT/FIREWALL
./firewall_preconfiguration
# To preconfigure firewall machine
echo "1" > /proc/sys/net/ipv4/ip_forward
ngrep -d eth1 -t -e > /PROJECT/TEMP/PROXY_BUFF_INT &                    # To catch any
incoming data from internal network and put it in PROXY_BUFF using background process
# ngrep -d eth0 -t > /PROJECT/TEMP/PROXY_BUFF_EXT &          # To catch any incoming data
from outside and put it in PROXY_BUFF using background process

# ---------------------
# 2. Packet monitoring
# ---------------------

while [ 1 ]
do
      #### HANDLING PACKET FROM INTERNAL NETWORK ####
      #### ------------------------------------ ####

      SIZE_BUFF_INT=`wc -c /PROJECT/TEMP/PROXY_BUFF_INT | awk '{print $1}'`
      # echo size buffer "$[SIZE_BUFF_INT]"
      if [ "$SIZE_BUFF_INT"  != "0" ]
      then
                  cat /PROJECT/TEMP/PROXY_BUFF_INT >
                  /PROJECT/TEMP/PROXY_BUFF_INT_RAW
                  > /PROJECT/TEMP/PROXY_BUFF_INT

                  # to get jml baris of buffer
                  # -------------------------
                  BUFF_baris=`wc -l /PROJECT/TEMP/PROXY_BUFF_INT_RAW | awk '{print
                  $1}'`
                  SIZE_BUFF_INT=`wc -c /PROJECT/TEMP/PROXY_BUFF_INT_RAW | awk
                  '{print $1}'`
                  #echo size buffer bytewise "$[SIZE_BUFF_INT]"
                  #echo size buffer linewise "$[BUFF_baris]"
                  j=1
                  ### 1 ###
                  #########
                  ## Process every data in buffer line-wise
                  ## -------------------------------------
                  while [[ $j -le $BUFF_baris ]]                    # &&[[ $j -le 10 ]]
                  do
```

```
TRANSACTION_CODE=""
DELTA_TIME=""
## To identify the type of network transaction
## ----------------------------------------
TRANSACTION_CODE=`head -n $j
/PROJECT/TEMP/PROXY_BUFF_INT_RAW >
/PROJECT/TEMP/TRANSACTIONTEMP | tail -n 1
/PROJECT/TEMP/TRANSACTIONTEMP | awk '{print $1}'`
SENDER=`head -n $j /PROJECT/TEMP/PROXY_BUFF_INT_RAW >
/PROJECT/TEMP/TRANSACTIONTEMP | tail -n 1
/PROJECT/TEMP/TRANSACTIONTEMP | awk '{print $4}'| awk -F:
'{print $1}'`
RECEIVER=`head -n $j /PROJECT/TEMP/PROXY_BUFF_INT_RAW >
/PROJECT/TEMP/TRANSACTIONTEMP | tail -n 1
/PROJECT/TEMP/TRANSACTIONTEMP | awk '{print $6}'| awk -F:
'{print $1}'`

#DELTA_TIME=`head -n 2
/PROJECT/TEMP/PROXY_BUFF_INT_RAW >
/PROJECT/TEMP/TRANSACTIONTEMP | tail -n 1
/PROJECT/TEMP/TRANSACTIONTEMP | awk '{print $3}'`
#ACCESS_CODE=`head -n 2
/PROJECT/TEMP/PROXY_BUFF_INT_RAW >
/PROJECT/TEMP/TRANSACTIONTEMP | tail -n 1
/PROJECT/TEMP/TRANSACTIONTEMP | awk '{print $7}'`

### 2 ###
#########

if [[ $TRANSACTION_CODE = U ]] || [[ $TRANSACTION_CODE = T
]] || [[ $TRANSACTION_CODE = I ]]
then
  if [[ $SENDER != "" ]] && [[ $RECEIVER != "" ]]
  then
        # echo $j - $TRANSACTION_CODE - $SENDER -
        $RECEIVER

        # Define the protocol
        if [[ $TRANSACTION_CODE = U ]]
        then
                TRANSACTION_CODE=udp
        fi
        if [[ $TRANSACTION_CODE = T ]]
        then
                TRANSACTION_CODE=tcp
        fi
        if [[ $TRANSACTION_CODE = I ]]
        then
                TRANSACTION_CODE=icmp
        fi

        # Define the subnet (IP Mask) of the sender
        echo $SENDER > /PROJECT/TEMP/IP_ADDR_SENDER
        first_digit=`head -n 1 /PROJECT/TEMP/IP_ADDR_SENDER |
        awk -F. '{print $1}'`
        second_digit=`head -n 1 /PROJECT/TEMP/IP_ADDR_SENDER
        | awk -F. '{print $2}'`
        third_digit=`head -n 1 /PROJECT/TEMP/IP_ADDR_SENDER |
        awk -F. '{print $3}'`
```

```
SENDER_SUBNET=`echo
"$first_digit.$second_digit.$third_digit"`
# echo $SENDER_SUBNET

# To define which part is outsider, whether it's the sender or
receiver
# ------------------------------------------------------------------
if [[ "$SENDER_SUBNET" = "$INTERNAL_SUBNET" ]]
then
        OUTSIDER=$RECEIVER
        INSIDER=$SENDER
else
        OUTSIDER=$SENDER
        INSIDER=$RECEIVER
fi
# wkt_skrg=`date | awk '{print $4}'`
# echo $j - $TRANSACTION_CODE - $INSIDER -
$OUTSIDER

AUTHENTICATED=`grep $OUTSIDER
/PROJECT/TABLE/zero_table`
if [ "$AUTHENTICATED" != "" ]
then
        # echo ACCESS IS PERMITTED
        wkt_skrg=`date | awk '{print $4}'`
        echo $j - $TRANSACTION_CODE - $SENDER -
        $RECEIVER $wkt_skrg PERMITTED

        ALREADY_INSERTED=`grep
        $TRANSACTION_CODE
        /PROJECT/TEMP/HISTORY_ZERO | grep
        $OUTSIDER | grep $INSIDER`

        # ALREADY_INSERTED=`grep $OUTSIDER
        /PROJECT/TEMP/HISTORY_ZERO | grep $INSIDER`
        if [ "$ALREADY_INSERTED" = "" ]
        #New access
        then
                # echo CANAL IS OPENED
                wkt_skrg=`date | awk '{print $4}'`
                                                echo
                $TRANSACTION_CODE $INSIDER
                $OUTSIDER $wkt_skrg >>
                /PROJECT/TEMP/HISTORY_ZERO
                echo CANAL IS OPENED $wkt_skrg

                # Rebuild the canals
                iptables-save > /PROJECT/TEMP/saveiptables
                > /PROJECT/TEMP/saveiptablescopy
                Canal_baris=`wc -l
                /PROJECT/TEMP/saveiptables | awk '{print
                $1}'`
                k=1
                while [[ $k -le $Canal_baris ]]
                do
                        isikanal=`head -n $k
                        /PROJECT/TEMP/saveiptables >
                        /PROJECT/TEMP/TRANSACTIONT
                        EMP | tail -n 1
```

```
                                                        /PROJECT/TEMP/TRANSACTIONT
                                                        EMP`
                                                        #echo $isikanal
        if [[ $isikanal = "-A FORWARD -i eth1 -j DROP " ]]
        then
                echo -A FORWARD -s $INSIDER -d $OUTSIDER -p $TRANSACTION_CODE -j
                ACCEPT >> /PROJECT/TEMP/saveiptablescopy
                echo -A FORWARD -s $OUTSIDER -d $INSIDER -p $TRANSACTION_CODE -j
                ACCEPT >> /PROJECT/TEMP/saveiptablescopy
                echo -A FORWARD -i eth1 -j DROP >> /PROJECT/TEMP/saveiptablescopy
        else
                echo $isikanal >> /PROJECT/TEMP/saveiptablescopy
        fi
        k=`expr ${k} + 1`
done
cat /PROJECT/TEMP/saveiptablescopy | iptables-restore
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
echo 1 > /proc/sys/net/ipv4/ip_forward

#./zero_preconfiguration
#Canal_baris=`wc -l /PROJECT/TEMP/HISTORY_ZERO | awk '{print $1}'`
#k=1
#while [[ $k -le $Canal_baris ]]
#do
#       TRANS=`head -n $k /PROJECT/TEMP/HISTORY_ZERO >
/PROJECT/TEMP/TRANSACTIONTEMP | tail -n 1 /PROJECT/TEMP/TRANSACTIONTEMP |
awk '{print $1}'`
#       IN=`head -n $k /PROJECT/TEMP/HISTORY_ZERO >
/PROJECT/TEMP/TRANSACTIONTEMP | tail -n 1 /PROJECT/TEMP/TRANSACTIONTEMP |
awk '{print $2}'`
#       OUT=`head -n $k /PROJECT/TEMP/HISTORY_ZERO >
/PROJECT/TEMP/TRANSACTIONTEMP | tail -n 1 /PROJECT/TEMP/TRANSACTIONTEMP |
awk '{print $3}'`
fi
else
echo $j - $TRANSACTION_CODE - $SENDER - $RECEIVER $wkt_skrg DENIED
fi
fi
fi
j=`expr ${j} + 1`
done

fi

done

echo "Finish"
```

```
#!/bin/sh

# PROCEDURE FOR DELETING CANAL
# RUNNING IN PARALLEL WITH FIREWALL_MAIN
# eth0 = EXTERNAL NETWORK
# eth1 = INTERNAL NETWORK

# ------------------
# 1. Initialization
# ------------------

> /PROJECT/TEMP/HISTORY_DELETE
> /PROJECT/TEMP/HISTORY_TEMP_DELETE

# ---------------------
# 2. Start processing
# ---------------------
while [ 1 ]
do
        # Identification of new canal through the addition of line
        # ------------------------------------------------------
        jumlah_baris_history=`wc -l /PROJECT/TEMP/HISTORY_ZERO | awk '{print $1}'`

        if [ $jumlah_baris_history -gt 0 ]
        then
                echo .
                echo .
                # Processing canal by processing the first line of HISTORY_ZERO file for each
                canal
                # ------------------------------------------------------------------------
                analyzed_transaction=`head -n 1 /PROJECT/TEMP/HISTORY_ZERO | awk '{print
                $1}'`
                analyzed_internal=`head -n 1 /PROJECT/TEMP/HISTORY_ZERO | awk '{print
                $2}'`
                analyzed_external=`head -n 1 /PROJECT/TEMP/HISTORY_ZERO | awk '{print
                $3}'`
                analyzed_starttime=`head -n 1 /PROJECT/TEMP/HISTORY_ZERO | awk '{print
                $4}'`

                # To obtain time duration of network transaction
                # ----------------------------------------------
                life_time=""
                life_time=`grep $analyzed_external /PROJECT/TABLE/zero_table | awk '{print
                $3}'`
                #if [[ $life_time = "" ]]
                #then
                #       life_time=60
                #fi
                echo life_time $life_time

                # To get detail on the canal hour, minute and second
                # --------------------------------------------------
                echo $analyzed_starttime > /PROJECT/TEMP/WKT_TEMP
                analyzed_starttime_jam=`awk -F: '{print $1}' /PROJECT/TEMP/WKT_TEMP`

                analyzed_starttime_jam=`expr ${analyzed_starttime_jam} \* 3600`
                analyzed_starttime_menit=`awk -F: '{print $2}' /PROJECT/TEMP/WKT_TEMP`
                analyzed_starttime_menit=`expr ${analyzed_starttime_menit} \* 60`
                analyzed_starttime_detik=`awk -F: '{print $3}' /PROJECT/TEMP/WKT_TEMP`
```

```
analyzed_starttime_integer=`expr ${analyzed_starttime_jam} +
${analyzed_starttime_menit} + ${analyzed_starttime_detik}`
echo analyzed_starttime is $analyzed_starttime_integer

# To get current time
# -------------------
wkt_skrg=`date | awk '{print $4}'`
echo $wkt_skrg > /PROJECT/TEMP/WKT_TEMP
wkt_skrg_jam=`awk -F: '{print $1}' /PROJECT/TEMP/WKT_TEMP`
wkt_skrg_jam=`expr ${wkt_skrg_jam} \* 3600`
wkt_skrg_menit=`awk -F: '{print $2}' /PROJECT/TEMP/WKT_TEMP`
wkt_skrg_menit=`expr ${wkt_skrg_menit} \* 60`
wkt_skrg_detik=`awk -F: '{print $3}' /PROJECT/TEMP/WKT_TEMP`
wkt_skrg_integer=`expr ${wkt_skrg_jam} + ${wkt_skrg_menit} +
${wkt_skrg_detik}`
echo wkt skrg is $wkt_skrg_integer

# To calculate how long an OLD CANAL has been listed in the Filter Table (the
value of b ($b) points to the rule being monitored)
# ---------------------------------------------------------------------------------------------
if [[ $analyzed_starttime_integer -gt $wkt_skrg_integer ]]
then
        lama_hidup_canal=`expr $analyzed_starttime_integer - $wkt_skrg_integer`
else
        lama_hidup_canal=`expr $wkt_skrg_integer - $analyzed_starttime_integer`
fi
echo "Lamanya rule hidup adalah" $lama_hidup_canal

# Comparing the canal life time and the time duration of living canal
# -------------------------------------------------------------------
if [ $lama_hidup_canal -gt $life_time ]
then
        # Delete data in the first line of HISTORY_ZERO file (Process each canal
        in the first line, remember ...)
        # ------------------------------------------------------------------------
        jumlah_baris_history=`wc -l /PROJECT/TEMP/HISTORY_ZERO | awk
        '{print $1}'`
        a=2
        while [[ $a -le $jumlah_baris_history ]]
        do
        head -n $a /PROJECT/TEMP/HISTORY_ZERO >
        /PROJECT/TEMP/HISTORY_TEMP_DELETE | tail -n 1
        /PROJECT/TEMP/HISTORY_TEMP_DELETE >>
        /PROJECT/TEMP/HISTORY_DELETE
        a=`expr ${a} + 1`
        echo $a
        done

        # To return the sec rule from the temporary file to the RULE_CACHE back
        # ---------------------------------------------------------------------
        jumlah_baris_history_delete=`wc -l
        /PROJECT/TEMP/HISTORY_DELETE | awk '{print $1}'`
        head -n $jumlah_baris_history_delete
        /PROJECT/TEMP/HISTORY_DELETE >
        /PROJECT/TEMP/HISTORY_ZERO

        # Determining transaction protocol
        # --------------------------------
        protocol=""
        if [[ $analyzed_transaction = udp ]]
```

```
                then
                        protocol="udp"
                fi
                if [[ $analyzed_transaction = icmp ]]
                then
                        protocol="icmp"
                fi
                if [[ $analyzed_transaction = tcp ]]
                then
                        protocol="tcp"
                fi
                # Real action for dropping canal in the Filter Table
                # -------------------------------------------------
                iptables-save > /PROJECT/TEMP/saveiptablesdrop
                > /PROJECT/TEMP/saveiptablescopydrop
                Canal_baris=`wc -l /PROJECT/TEMP/saveiptablesdrop | awk '{print $1}'`
                k=1
                while [[ $k -le $Canal_baris ]]
                do
                isikanal=`head -n $k /PROJECT/TEMP/saveiptablesdrop >
                /PROJECT/TEMP/TRANSACTIONTEMPDROP | tail -n 1
                /PROJECT/TEMP/TRANSACTIONTEMPDROP`
                echo $isikanal > /PROJECT/TEMP/ISIKANALDROP
                get_isikanal1=`grep $analyzed_external
                /PROJECT/TEMP/ISIKANALDROP | awk '{print $4}'`
                get_isikanal2=`grep $analyzed_external
                /PROJECT/TEMP/ISIKANALDROP | awk '{print $6}'`
                get_isikanal3=`grep $analyzed_external
                /PROJECT/TEMP/ISIKANALDROP | awk '{print $8}'`
                if [[ $get_isikanal1 = $analyzed_external ]] && [[ $get_isikanal2 =
                $analyzed_internal ]] && [[ $get_isikanal3 = $protocol ]]
                then
                echo DROP $protocol $analyzed_external $analyzed_internal
                else
                if [[ $get_isikanal1 = $analyzed_internal ]] && [[ $get_isikanal2 =
                $analyzed_external ]] && [[ $get_isikanal3 = $protocol ]]
                then
                echo DROP $protocol $analyzed_internal $analyzed_external
                else
                echo $isikanal >> /PROJECT/TEMP/saveiptablescopydrop
                fi
                fi
                k=`expr ${k} + 1`
                done
                cat /PROJECT/TEMP/saveiptablescopydrop | iptables-restore
                iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
                echo 1 > /proc/sys/net/ipv4/ip_forward
                #iptables -D FORWARD -i eth1 -p $protocol -s $analyzed_internal -d
                $analyzed_external -j ACCEPT
                #iptables -D FORWARD -i eth0 -p $protocol -s $analyzed_external -d
                $analyzed_internal -j ACCEPT
                > /PROJECT/TEMP/HISTORY_DELETE
                > /PROJECT/TEMP/HISTORY_TEMP_DELETE
                # echo DELETE $protocol $analyzed_internal $analyzed_external
        fi
        echo .
        echo .
    fi
done
```

APPENDIX D1
Table of External Parties for Zero-Configuration

Table D1: List of external parties

```
fsksm.utm.my 161.139.68.251 02:30.2
fsksm.utm.my 161.139.250.2 02:30.2
fsksm.utm.my 161.139.68.247 02:30.2
fsksm.utm.my 216.239.57.103 02:30.2
fsksm.utm.my 216.239.57.104 02:30.2
fsksm.utm.my 216.239.57.99 02:30.2

web.utm.my 161.139.18.91 02:40.1
www.utm.my 161.139.18.99 02:40.1

www.ukm.my 202.185.33.79 00:17.9

www.elsevier.com 129.35.76.177 01:15.5

www.gatra.com 64.156.138.148 02:56.8

www.google.com 63.150.131.40 00:30.1
www.google.com 64.233.189.104 00:30.1
www.google.com 64.233.161.147 00:30.1
www.google.com 64.233.161.104 00:30.1
www.google.com 66.94.229.254 00:30.1
www.google.com 66.102.7.104 00:30.1
www.google.com 66.102.9.104 00:30.1
www.google.com 66.102.9.99 00:30.1
www.google.com 66.102.11.104 00:30.1
www.google.com 66.102.11.99 00:30.1

www.kompas.com 64.203.71.11 02:20.6
www.kompas.com 64.203.71.51 02:20.6

www.rd.com 164.109.22.52 03:03.0
www.rd.com 164.109.22.155 03:03.0
```

APPENDIX D2
Experimental Results of Zero-Configuration

Table D2: Experimental result for opening www.fsksm.utm.my

Name of External Party	Δt_{oc}	$\Delta t_{transaction}$	**Cumulative Mean**	**Cumulative Std Dev**	**Life-time of canals**	**Access Results**
www.fsksm.utm.my	00:02.4	00:22.4	00:22.4	00:22.4	01:29.6	Success
	00:02.7	00:30.1	00:26.3	00:05.4	00:42.6	Success
	00:00.7	00:30.7	00:27.7	00:04.6	00:41.6	Success
	00:01.7	00:13.3	00:24.1	00:08.2	00:48.6	Success
	00:09.1	00:31.3	00:25.5	00:07.8	**00:48.8**	Success
	00:36.0	**01:55.8**	00:40.6	00:37.5	02:33.1	**Denied**
	00:01.5	00:09.7	00:36.2	00:36.2	02:24.7	Success
	00:09.4	00:35.3	00:36.1	00:33.5	02:16.6	Success
	00:37.5	01:34.4	00:42.6	00:36.9	02:33.2	Success
	00:03.3	00:05.9	00:38.9	00:36.7	02:28.8	Success
	00:04.7	00:23.7	00:37.5	00:35.1	02:22.7	Success
	00:18.8	01:07.1	00:40.0	00:34.5	02:23.5	Success
	01:16.1	01:56.4	00:45.9	00:39.3	02:43.7	Success
	00:02.4	00:09.9	00:43.3	00:38.9	02:40.1	Success
	00:09.1	00:36.7	00:42.9	00:37.6	02:35.5	Success
	00:36.7	01:35.0	00:46.1	00:38.5	02:41.8	Success
	00:01.7	00:09.4	00:44.0	00:38.4	02:39.1	Success
	00:09.2	00:36.8	00:43.6	00:37.3	02:35.3	Success
	00:02.3	00:10.5	00:41.8	00:37.0	02:32.8	Success
	00:08.5	00:31.4	00:41.3	00:36.1	02:29.6	Success
	00:02.4	00:24.6	00:40.5	00:35.4	02:26.6	Success
	00:02.2	00:24.6	00:39.8	00:34.7	02:23.8	Success
	00:01.8	00:10.5	00:38.5	00:34.4	02:21.8	Success
	00:08.6	00:31.3	00:38.2	00:33.7	02:19.3	Success
	00:36.8	02:00.0	00:41.5	00:36.8	02:32.0	Success
	00:34.2	01:44.7	00:43.9	00:38.2	02:38.4	Success
	00:02.6	00:24.7	00:43.2	00:37.6	02:36.0	Success
	00:01.5	00:10.4	00:42.0	00:37.4	02:34.3	Success
	00:02.4	00:12.9	00:41.0	00:37.1	02:32.4	Success
	00:17.6	00:59.0	00:41.6	00:36.6	02:31.5	Success
	00:00.3	00:06.9	00:40.5	00:36.6	02:30.2	Success
Average	00:12.4	00:40.5	-	-	-	-
Standard Deviation	00:17.2	00:36.6	-	-	-	-
Denial of Service	1 of 31 transactions					

Table D3: Experimental result for opening www.utm.my

Name of External Party	Δt_{oc}	$\Delta t_{transaction}$	**Cumulative Mean**	**Cumulative Std Dev**	**Life-time of canals**	**Access Results**
www.utm.my	00:02.2	00:18.7	00:18.7	00:18.7	01:14.8	Success
	00:02.2	00:10.5	00:14.6	00:05.8	00:32.0	Success
	00:08.9	00:28.9	00:19.4	00:09.3	**00:47.1**	Success
	00:36.3	**01:54.5**	00:43.2	00:48.2	03:07.6	**Denied**
	00:02.3	00:10.5	00:36.6	00:44.2	02:49.2	Success
	00:08.5	00:34.0	00:36.2	00:39.5	02:34.8	Success
	00:36.4	01:53.6	00:47.2	00:46.5	03:06.6	Success
	00:35.7	01:31.9	00:52.8	00:45.8	03:10.3	Success
	00:03.0	00:15.8	00:48.7	00:44.6	03:02.5	Success
	00:00.5	00:11.3	00:45.0	00:43.7	02:56.0	Success
	00:09.2	00:28.2	00:43.4	00:41.8	02:48.7	Success
	00:24.0	01:10.7	00:45.7	00:40.6	02:47.5	Success
	01:13.5	01:54.5	00:51.0	00:43.3	03:00.8	Success
	00:09.0	00:23.9	00:49.1	00:42.2	02:55.7	Success
	00:05.2	00:12.8	00:46.7	00:41.7	02:51.9	Success
	00:05.4	00:28.1	00:45.5	00:40.6	02:47.3	Success
	00:35.9	01:53.5	00:49.5	00:42.6	02:57.4	Success
	00:18.4	01:04.8	00:50.3	00:41.5	02:54.9	Success
	00:01.2	00:09.6	00:48.2	00:41.4	02:52.4	Success
	00:04.1	00:18.1	00:46.7	00:40.9	02:49.3	Success
	00:02.8	00:18.7	00:45.4	00:40.3	02:46.2	Success
	00:01.6	00:10.6	00:43.8	00:40.0	02:43.8	Success
	00:00.6	00:46.2	00:43.9	00:39.1	02:41.2	Success
	00:02.0	00:16.0	00:42.7	00:38.7	02:38.7	Success
	00:01.9	00:10.4	00:41.4	00:38.4	02:36.6	Success
	00:09.1	00:43.7	00:41.5	00:37.6	02:34.4	Success
	00:10.0	00:40.3	00:41.5	00:36.9	02:32.1	Success
	00:03.6	02:17.1	00:44.9	00:40.5	02:46.3	Success
	00:02.5	00:18.5	00:44.0	00:40.0	02:44.1	Success
	00:02.3	00:10.6	00:42.9	00:39.8	02:42.3	Success
	00:08.5	00:31.2	00:42.5	00:39.2	02:40.1	Success
Average	00:11.8	00:42.5	-	-	-	-
Std Deviation	0.000188	00:39.2	-	-	-	-
Denial of Service	1 of 31 transactions					

Table D4: Experimental result for opening www.ukm.my

Name of External Party	Δt_{oc}	$\Delta t_{transaction}$	**Cumulative Mean**	**Cumulative Std Dev**	**Life-time of canals**	**Access Results**
www.ukm.my	00:00.5	00:04.6	00:04.6	00:04.6	00:18.2	Success
	00:00.9	00:10.8	00:07.7	00:04.4	00:21.0	Success
	00:00.8	00:10.3	00:08.6	00:03.5	00:19.0	Success
	00:00.7	00:04.6	00:07.6	00:03.5	00:18.0	Success
	00:02.6	00:12.7	00:08.6	00:03.8	00:19.9	Success
	00:03.0	00:19.0	00:10.3	00:05.4	00:26.6	Success
	00:00.2	00:04.5	00:09.5	00:05.4	00:25.7	Success
	00:00.9	00:04.2	00:08.8	00:05.3	00:24.9	Success
	00:00.7	00:04.5	00:08.3	00:05.2	00:23.9	Success
	00:00.7	00:10.6	00:08.6	00:05.0	00:23.4	Success
	00:00.2	00:10.9	00:08.8	00:04.8	00:23.0	Success
	00:00.6	00:04.5	00:08.4	00:04.7	00:22.5	Success
	00:00.7	00:04.5	00:08.1	00:04.6	00:22.0	Success
	00:00.5	00:04.5	00:07.9	00:04.5	00:21.5	Success
	00:01.0	00:11.0	00:08.1	00:04.5	00:21.4	Success
	00:00.2	00:04.5	00:07.9	00:04.4	00:21.0	Success
	00:00.5	00:05.0	00:07.7	00:04.3	00:20.6	Success
	00:00.3	00:10.5	00:07.8	00:04.2	00:20.5	Success
	00:00.9	00:07.9	00:07.8	00:04.1	00:20.2	Success
	00:00.7	00:04.5	00:07.7	00:04.1	00:19.9	Success
	00:00.7	00:04.5	00:07.5	00:04.0	00:19.6	Success
	00:00.3	00:04.5	00:07.4	00:04.0	00:19.3	Success
	00:00.8	00:04.6	00:07.3	00:03.9	00:19.1	Success
	00:00.6	00:04.8	00:07.2	00:03.9	00:18.8	Success
	00:00.7	00:06.0	00:07.1	00:03.8	00:18.5	Success
	00:00.0	00:04.6	00:07.0	00:03.8	00:18.3	Success
	00:00.1	00:04.1	00:06.9	00:03.7	00:18.1	Success
	00:00.5	00:04.2	00:06.8	00:03.7	00:17.9	Success
	00:00.4	00:10.5	00:06.9	00:03.7	00:18.0	Success
	00:00.7	00:04.1	00:06.9	00:03.7	00:17.9	Success
	00:00.4	00:10.3	00:07.0	00:03.7	00:17.9	Success
Average	00:00.7	00:07.0	-	-	-	-
Std Deviation	7.13E-06	00:03.7	-	-	-	-
Denial of Service	0 of 31 transactions					

Table D5: Experimental result for opening www.elsevier.com

Name of External Party	Δt_{oc}	$\Delta t_{transaction}$	**Cumulative Mean**	**Cumulative Std Dev**	**Life-time of canals**	**Access Results**
www.elsevier.com	00:00.3	00:46.5	00:46.5	00:46.5	03:06.1	Success
	00:00.2	00:44.8	00:45.7	00:01.2	**00:49.3**	Success
	00:01.5	**00:49.8**	00:47.1	00:02.5	00:54.7	**Denied**
	00:00.2	00:46.2	00:46.8	00:02.1	**00:53.2**	Success
	00:02.6	**01:12.5**	00:52.0	00:11.6	01:26.8	**Denied**
	00:00.4	00:47.8	00:51.3	00:10.5	01:22.8	Success
	00:03.1	00:59.2	00:52.4	00:10.1	01:22.6	Success
	00:00.9	00:44.7	00:51.4	00:09.7	01:20.6	Success
	00:00.6	00:46.7	00:50.9	00:09.2	01:18.6	Success
	00:00.3	00:44.6	00:50.3	00:08.9	01:17.0	Success
	00:00.6	00:48.0	00:50.1	00:08.5	01:15.5	Success
Average	00:01.0	00:50.1	-	-	-	-
Std Deviation	1.18E-05	00:08.5	-	-	-	-
Denial of Service	2 of 11 transactions					

Table D6: Experimental result for opening www.gatra.com

Name of External Party	Δt_{oc}	$\Delta t_{transaction}$	**Cumulative Mean**	**Cumulative Std Dev**	**Life-time of canals**	**Access Results**
www.gatra.com	00:02.9	00:33.5	00:33.5	00:33.5	02:14.0	Success
	00:00.5	00:40.8	00:37.2	00:05.2	00:52.7	Success
	00:00.4	00:43.1	00:39.1	00:05.0	**00:54.2**	Success
	00:00.7	**01:32.4**	00:52.5	00:27.0	02:13.4	**Denied**
	00:02.0	00:46.7	00:51.3	00:23.5	02:01.8	Success
	00:02.9	01:52.7	01:01.5	00:32.7	02:39.7	Success
	00:02.4	01:57.3	01:09.5	00:36.6	02:59.2	Success
	00:02.7	00:33.4	01:05.0	00:36.2	02:53.6	Success
	00:00.1	01:32.6	01:08.1	00:35.1	02:53.3	Success
	00:02.5	01:57.4	01:13.0	00:36.6	03:02.7	Success
	00:02.5	00:52.5	01:11.1	00:35.2	02:56.8	Success
Average	00:01.8	01:11.1	-	-	-	-
Std Deviation	1.29E-05	00:35.2	-	-	-	-
Denial of Service	1 of 11 transactions					

Table D7: Experimental result for opening www.google.com

Name of External Party	Δt_{oc}	$\Delta t_{transaction}$	**Cumulative Mean**	**Cumulative Std Dev**	**Life-time of canals**	**Access Results**
www.google.com	00:00.1	00:12.4	00:12.4	00:12.4	00:49.7	Success
	00:02.3	00:19.3	00:15.8	00:04.8	00:30.4	Success
	00:02.4	00:25.1	00:18.9	00:06.4	00:38.0	Success
	00:02.7	00:19.3	00:19.0	00:05.2	00:34.6	Success
	00:03.1	00:19.2	00:19.1	00:04.5	00:32.6	Success
	00:02.2	00:19.2	00:19.1	00:04.0	00:31.2	Success
	00:02.7	00:19.1	00:19.1	00:03.7	00:30.1	Success
	00:02.7	00:19.2	00:19.1	00:03.4	00:29.3	Success
	00:02.7	00:19.3	00:19.1	00:03.2	00:28.7	Success
	00:02.8	00:25.6	00:19.8	00:03.6	00:30.7	Success
	00:02.7	00:19.2	00:19.7	00:03.5	00:30.1	Success
Average	00:02.4	00:12.4	-	-	-	-
Std Deviation	9.14E-06	00:19.3	-	-	-	-
Denial of Service	0 of 11 transactions					

Table D8: Experimental result for opening www.kompas.com

Name of External Party	Δt_{oc}	$\Delta t_{transaction}$	**Cumulative Mean**	**Cumulative Std Dev**	**Life-time of canals**	**Access Results**
www.kompas.com	00:06.6	01:30.0	01:30.0	01:30.0	06:00.1	Success
	00:11.1	00:54.0	01:12.0	00:25.4	02:28.4	Success
	00:01.1	00:27.1	00:57.1	00:31.6	02:31.8	Success
	00:01.0	00:34.4	00:51.4	00:28.1	02:15.8	Success
	00:01.0	00:30.1	00:47.1	00:26.2	02:05.7	Success
	00:10.9	01:42.7	00:56.4	00:32.6	02:34.2	Success
	00:02.9	01:36.3	01:02.1	00:33.4	02:42.2	Success
	00:03.0	00:33.8	00:58.6	00:32.5	02:35.9	Success
	00:01.0	00:28.8	00:55.3	00:31.9	02:31.1	Success
	00:02.4	00:33.9	00:53.1	00:30.9	02:25.7	Success
	00:02.4	00:40.3	00:52.0	00:29.5	02:20.6	Success
Average	00:03.9	00:52.0	-	-	-	-
Std Deviation	4.44E-05	00:29.5	-	-	-	-
Denial of Service	0 of 11 transactions					

Table D7: Experimental result for opening www.rd.com

Name of External Party	Δt_{oc}	$\Delta t_{transaction}$	**Cumulative Mean**	**Cumulative Std Dev**	**Life-time of canals**	**Access Results**
www.rd.com	00:20.6	01:42.1	01:42.1	01:42.1	06:48.3	Success
	00:02.4	00:41.5	01:11.8	00:42.8	03:20.2	Success
	00:00.7	00:24.8	00:56.1	00:40.6	02:58.1	Success
	00:09.3	00:44.1	00:53.1	00:33.7	02:34.3	Success
	00:09.7	00:58.5	00:54.2	00:29.3	02:22.1	Success
	00:01.2	01:25.3	00:59.4	00:29.1	02:26.8	Success
	00:02.6	01:23.9	01:02.9	00:28.2	02:27.4	Success
	00:02.5	02:02.9	01:10.4	00:33.6	02:51.3	Success
	00:02.8	02:02.2	01:16.1	00:35.9	03:03.8	Success
	00:01.2	01:57.5	01:20.3	00:36.3	03:09.1	Success
	00:02.7	01:08.0	01:19.2	00:34.6	03:03.0	Success
Average	00:05.1	01:19.2	-	-	-	-
Std Deviation	6.95E-05	00:34.6	-	-	-	-
Denial of Service	0 of 11 transactions					

APPENDIX D3
Comparison of Processing Time and Number of Packets of Zero-Configuration

Table D8: Processing time versus number of packets

No.	**Number of packets**	Δt **(min: seconds)**	No.	**Number of packets**	Δt **(min: seconds)**
1	1	00:00.0	31	137	00:09.0
2	2	00:00.0	32	166	00:11.0
3	2	00:01.0	33	176	00:09.0
4	3	00:00.0	34	178	00:09.0
5	3	00:00.0	35	194	00:13.0
6	3	00:00.0	36	205	00:12.0
7	6	00:01.0	37	232	00:13.0
8	8	00:00.0	38	249	00:18.0
9	8	00:01.0	39	293	00:17.0
10	8	00:01.0	40	308	00:20.0
11	9	00:00.0	41	313	00:20.0
12	14	00:01.0	42	321	00:23.0
13	16	00:01.0	43	329	00:18.0
14	18	00:01.0	44	335	00:18.0
15	19	00:02.0	45	349	00:18.0
16	19	00:03.0	46	418	00:30.0
17	20	00:02.0	47	455	00:27.0
18	22	00:01.0	48	484	00:29.0
19	29	00:02.0	49	487	00:29.0
20	35	00:02.0	50	524	00:28.0
21	43	00:03.0	51	561	00:38.0
22	43	00:03.0	52	569	00:40.0
23	45	00:03.0	53	618	00:39.0
24	66	00:05.0	54	651	00:38.0
25	71	00:05.0	55	730	00:35.0
26	83	00:06.0	56	887	01:03.0
27	94	00:07.0	57	942	01:10.0
28	107	00:06.0	58	1065	01:03.0
29	121	00:06.0	59	1072	01:07.0
30	123	00:07.0	60	1326	01:23.0

APPENDIX D4
Foreground Algorithm of Zero-Configuration

```
#!/bin/sh
# FIREWALL BASED ON ZERO CONFIGURATION #
# eth0 = EXTERNAL NETWORK
# eth1 = INTERNAL NETWORK
# ------------------
# 1. Initialization
# ------------------
INTERNAL_SUBNET="24.4.74"
> /PROJECT/TEMP/PROXY_BUFF_INT
> /PROJECT/TEMP/PROXY_BUFF_EXT
> /PROJECT/TEMP/IP_SENDER
> /PROJECT/TEMP/IP_ADDR_SENDER
> /PROJECT/TEMP/IP_ADDR_RECEIVER
> /PROJECT/TEMP/TABLEIP
> /PROJECT/TEMP/HISTORY_ZERO
> /PROJECT/TEMP/DELTATIME
> /PROJECT/TEMP/TIMEOUTPUT
> /PROJECT/TEMP/LOG
cd /PROJECT/FIREWALL
./firewall_preconfiguration
# To preconfigure firewall machine
echo "1" > /proc/sys/net/ipv4/ip_forward
ngrep -d eth1 -t -e > /PROJECT/TEMP/PROXY_BUFF_INT &                    # To catch any
incoming data from internal network and put it in PROXY_BUFF using background process
# ngrep -d eth0 -t > /PROJECT/TEMP/PROXY_BUFF_EXT &          # To catch any incoming data
from outside and put it in PROXY_BUFF using background process

# ---------------------
# 2. Packet monitoring
# ---------------------

while [ 1 ]
do
      #### HANDLING PACKET FROM INTERNAL NETWORK ####
      #### ------------------------------------ ####

      SIZE_BUFF_INT=`wc -c /PROJECT/TEMP/PROXY_BUFF_INT | awk '{print $1}'`
      # echo size buffer "$[SIZE_BUFF_INT]"
      if [ "$SIZE_BUFF_INT"  != "0" ]
      then
                  cat /PROJECT/TEMP/PROXY_BUFF_INT >
                  /PROJECT/TEMP/PROXY_BUFF_INT_RAW
                  > /PROJECT/TEMP/PROXY_BUFF_INT

                  # to get jml baris of buffer
                  # -------------------------
                  BUFF_baris=`wc -l /PROJECT/TEMP/PROXY_BUFF_INT_RAW | awk '{print
                  $1}'`
                  SIZE_BUFF_INT=`wc -c /PROJECT/TEMP/PROXY_BUFF_INT_RAW | awk
                  '{print $1}'`
                  #echo size buffer bytewise "$[SIZE_BUFF_INT]"
                  #echo size buffer linewise "$[BUFF_baris]"
                  j=1
                  ### 1 ###
                  #########
                  ## Process every data in buffer line-wise
                  ## -------------------------------------
                  while [[ $j -le $BUFF_baris ]]                    # &&[[ $j -le 10 ]]
                  do
```

```
TRANSACTION_CODE=""
DELTA_TIME=""
## To identify the type of network transaction
## ----------------------------------------
TRANSACTION_CODE=`head -n $j
/PROJECT/TEMP/PROXY_BUFF_INT_RAW >
/PROJECT/TEMP/TRANSACTIONTEMP | tail -n 1
/PROJECT/TEMP/TRANSACTIONTEMP | awk '{print $1}'`
SENDER=`head -n $j /PROJECT/TEMP/PROXY_BUFF_INT_RAW >
/PROJECT/TEMP/TRANSACTIONTEMP | tail -n 1
/PROJECT/TEMP/TRANSACTIONTEMP | awk '{print $4}'| awk -F:
'{print $1}'`
RECEIVER=`head -n $j /PROJECT/TEMP/PROXY_BUFF_INT_RAW >
/PROJECT/TEMP/TRANSACTIONTEMP | tail -n 1
/PROJECT/TEMP/TRANSACTIONTEMP | awk '{print $6}'| awk -F:
'{print $1}'`

#DELTA_TIME=`head -n 2
/PROJECT/TEMP/PROXY_BUFF_INT_RAW >
/PROJECT/TEMP/TRANSACTIONTEMP | tail -n 1
/PROJECT/TEMP/TRANSACTIONTEMP | awk '{print $3}'`
#ACCESS_CODE=`head -n 2
/PROJECT/TEMP/PROXY_BUFF_INT_RAW >
/PROJECT/TEMP/TRANSACTIONTEMP | tail -n 1
/PROJECT/TEMP/TRANSACTIONTEMP | awk '{print $7}'`

### 2 ###
#########

if [[ $TRANSACTION_CODE = U ]] || [[ $TRANSACTION_CODE = T
]] || [[ $TRANSACTION_CODE = I ]]
then
  if [[ $SENDER != "" ]] && [[ $RECEIVER != "" ]]
  then
        # echo $j - $TRANSACTION_CODE - $SENDER -
        $RECEIVER

        # Define the protocol
        if [[ $TRANSACTION_CODE = U ]]
        then
                TRANSACTION_CODE=udp
        fi
        if [[ $TRANSACTION_CODE = T ]]
        then
                TRANSACTION_CODE=tcp
        fi
        if [[ $TRANSACTION_CODE = I ]]
        then
                TRANSACTION_CODE=icmp
        fi

        # Define the subnet (IP Mask) of the sender
        echo $SENDER > /PROJECT/TEMP/IP_ADDR_SENDER
        first_digit=`head -n 1 /PROJECT/TEMP/IP_ADDR_SENDER |
        awk -F. '{print $1}'`
        second_digit=`head -n 1 /PROJECT/TEMP/IP_ADDR_SENDER
        | awk -F. '{print $2}'`
        third_digit=`head -n 1 /PROJECT/TEMP/IP_ADDR_SENDER |
        awk -F. '{print $3}'`
```

```
SENDER_SUBNET=`echo
"$first_digit.$second_digit.$third_digit"`
# echo $SENDER_SUBNET

# To define which part is outsider, whether it's the sender or
receiver
# ------------------------------------------------------------------
if [[ "$SENDER_SUBNET" = "$INTERNAL_SUBNET" ]]
then
        OUTSIDER=$RECEIVER
        INSIDER=$SENDER
else
        OUTSIDER=$SENDER
        INSIDER=$RECEIVER
fi
# wkt_skrg=`date | awk '{print $4}'`
# echo $j - $TRANSACTION_CODE - $INSIDER -
$OUTSIDER

AUTHENTICATED=`grep $OUTSIDER
/PROJECT/TABLE/zero_table`
if [ "$AUTHENTICATED" != "" ]
then
        # echo ACCESS IS PERMITTED
        wkt_skrg=`date | awk '{print $4}'`
        echo $j - $TRANSACTION_CODE - $SENDER -
        $RECEIVER $wkt_skrg PERMITTED

        ALREADY_INSERTED=`grep
        $TRANSACTION_CODE
        /PROJECT/TEMP/HISTORY_ZERO | grep
        $OUTSIDER | grep $INSIDER`

        # ALREADY_INSERTED=`grep $OUTSIDER
        /PROJECT/TEMP/HISTORY_ZERO | grep $INSIDER`
        if [ "$ALREADY_INSERTED" = "" ]
        #New access
        then
                # echo CANAL IS OPENED
                wkt_skrg=`date | awk '{print $4}'`
                                                echo
                $TRANSACTION_CODE $INSIDER
                $OUTSIDER $wkt_skrg >>
                /PROJECT/TEMP/HISTORY_ZERO
                echo CANAL IS OPENED $wkt_skrg

                # Rebuild the canals
                iptables-save > /PROJECT/TEMP/saveiptables
                > /PROJECT/TEMP/saveiptablescopy
                Canal_baris=`wc -l
                /PROJECT/TEMP/saveiptables | awk '{print
                $1}'`
                k=1
                while [[ $k -le $Canal_baris ]]
                do
                        isikanal=`head -n $k
                        /PROJECT/TEMP/saveiptables >
                        /PROJECT/TEMP/TRANSACTIONT
                        EMP | tail -n 1
```

```
                                                        /PROJECT/TEMP/TRANSACTIONT
                                                        EMP`
                                                        #echo $isikanal
        if [[ $isikanal = "-A FORWARD -i eth1 -j DROP " ]]
        then
                echo -A FORWARD -s $INSIDER -d $OUTSIDER -p $TRANSACTION_CODE -j
                ACCEPT >> /PROJECT/TEMP/saveiptablescopy
                echo -A FORWARD -s $OUTSIDER -d $INSIDER -p $TRANSACTION_CODE -j
                ACCEPT >> /PROJECT/TEMP/saveiptablescopy
                echo -A FORWARD -i eth1 -j DROP >> /PROJECT/TEMP/saveiptablescopy
        else
                echo $isikanal >> /PROJECT/TEMP/saveiptablescopy
        fi
        k=`expr ${k} + 1`
done
cat /PROJECT/TEMP/saveiptablescopy | iptables-restore
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
echo 1 > /proc/sys/net/ipv4/ip_forward

#./zero_preconfiguration
#Canal_baris=`wc -l /PROJECT/TEMP/HISTORY_ZERO | awk '{print $1}'`
#k=1
#while [[ $k -le $Canal_baris ]]
#do
#       TRANS=`head -n $k /PROJECT/TEMP/HISTORY_ZERO >
/PROJECT/TEMP/TRANSACTIONTEMP | tail -n 1 /PROJECT/TEMP/TRANSACTIONTEMP |
awk '{print $1}'`
#       IN=`head -n $k /PROJECT/TEMP/HISTORY_ZERO >
/PROJECT/TEMP/TRANSACTIONTEMP | tail -n 1 /PROJECT/TEMP/TRANSACTIONTEMP |
awk '{print $2}'`
#       OUT=`head -n $k /PROJECT/TEMP/HISTORY_ZERO >
/PROJECT/TEMP/TRANSACTIONTEMP | tail -n 1 /PROJECT/TEMP/TRANSACTIONTEMP |
awk '{print $3}'`
fi
else
echo $j - $TRANSACTION_CODE - $SENDER - $RECEIVER $wkt_skrg DENIED
fi
fi
fi
j=`expr ${j} + 1`
done

fi

done

echo "Finish"
```

```
#!/bin/sh

# PROCEDURE FOR DELETING CANAL
# RUNNING IN PARALLEL WITH FIREWALL_MAIN
# eth0 = EXTERNAL NETWORK
# eth1 = INTERNAL NETWORK

# ------------------
# 1. Initialization
# ------------------

> /PROJECT/TEMP/HISTORY_DELETE
> /PROJECT/TEMP/HISTORY_TEMP_DELETE

# ---------------------
# 2. Start processing
# ---------------------
while [ 1 ]
do
        # Identification of new canal through the addition of line
        # ------------------------------------------------------
        jumlah_baris_history=`wc -l /PROJECT/TEMP/HISTORY_ZERO | awk '{print $1}'`

        if [ $jumlah_baris_history -gt 0 ]
        then
                echo .
                echo .
                # Processing canal by processing the first line of HISTORY_ZERO file for each
                canal
                # ------------------------------------------------------------------------
                analyzed_transaction=`head -n 1 /PROJECT/TEMP/HISTORY_ZERO | awk '{print
                $1}'`
                analyzed_internal=`head -n 1 /PROJECT/TEMP/HISTORY_ZERO | awk '{print
                $2}'`
                analyzed_external=`head -n 1 /PROJECT/TEMP/HISTORY_ZERO | awk '{print
                $3}'`
                analyzed_starttime=`head -n 1 /PROJECT/TEMP/HISTORY_ZERO | awk '{print
                $4}'`

                # To obtain time duration of network transaction
                # ----------------------------------------------
                life_time=""
                life_time=`grep $analyzed_external /PROJECT/TABLE/zero_table | awk '{print
                $3}'`
                #if [[ $life_time = "" ]]
                #then
                #       life_time=60
                #fi
                echo life_time $life_time

                # To get detail on the canal hour, minute and second
                # --------------------------------------------------
                echo $analyzed_starttime > /PROJECT/TEMP/WKT_TEMP
                analyzed_starttime_jam=`awk -F: '{print $1}' /PROJECT/TEMP/WKT_TEMP`

                analyzed_starttime_jam=`expr ${analyzed_starttime_jam} \* 3600`
                analyzed_starttime_menit=`awk -F: '{print $2}' /PROJECT/TEMP/WKT_TEMP`
                analyzed_starttime_menit=`expr ${analyzed_starttime_menit} \* 60`
                analyzed_starttime_detik=`awk -F: '{print $3}' /PROJECT/TEMP/WKT_TEMP`
```

```
analyzed_starttime_integer=`expr ${analyzed_starttime_jam} +
${analyzed_starttime_menit} + ${analyzed_starttime_detik}`
echo analyzed_starttime is $analyzed_starttime_integer

# To get current time
# -------------------
wkt_skrg=`date | awk '{print $4}'`
echo $wkt_skrg > /PROJECT/TEMP/WKT_TEMP
wkt_skrg_jam=`awk -F: '{print $1}' /PROJECT/TEMP/WKT_TEMP`
wkt_skrg_jam=`expr ${wkt_skrg_jam} \* 3600`
wkt_skrg_menit=`awk -F: '{print $2}' /PROJECT/TEMP/WKT_TEMP`
wkt_skrg_menit=`expr ${wkt_skrg_menit} \* 60`
wkt_skrg_detik=`awk -F: '{print $3}' /PROJECT/TEMP/WKT_TEMP`
wkt_skrg_integer=`expr ${wkt_skrg_jam} + ${wkt_skrg_menit} +
${wkt_skrg_detik}`
echo wkt skrg is $wkt_skrg_integer

# To calculate how long an OLD CANAL has been listed in the Filter Table (the
value of b ($b) points to the rule being monitored)
# ---------------------------------------------------------------------------------------------
if [[ $analyzed_starttime_integer -gt $wkt_skrg_integer ]]
then
        lama_hidup_canal=`expr $analyzed_starttime_integer - $wkt_skrg_integer`
else
        lama_hidup_canal=`expr $wkt_skrg_integer - $analyzed_starttime_integer`
fi
echo "Lamanya rule hidup adalah" $lama_hidup_canal

# Comparing the canal life time and the time duration of living canal
# -------------------------------------------------------------------
if [ $lama_hidup_canal -gt $life_time ]
then
        # Delete data in the first line of HISTORY_ZERO file (Process each canal
        in the first line, remember ...)
        # ------------------------------------------------------------------------
        jumlah_baris_history=`wc -l /PROJECT/TEMP/HISTORY_ZERO | awk
        '{print $1}'`
        a=2
        while [[ $a -le $jumlah_baris_history ]]
        do
        head -n $a /PROJECT/TEMP/HISTORY_ZERO >
        /PROJECT/TEMP/HISTORY_TEMP_DELETE | tail -n 1
        /PROJECT/TEMP/HISTORY_TEMP_DELETE >>
        /PROJECT/TEMP/HISTORY_DELETE
        a=`expr ${a} + 1`
        echo $a
        done

        # To return the sec rule from the temporary file to the RULE_CACHE back
        # ---------------------------------------------------------------------
        jumlah_baris_history_delete=`wc -l
        /PROJECT/TEMP/HISTORY_DELETE | awk '{print $1}'`
        head -n $jumlah_baris_history_delete
        /PROJECT/TEMP/HISTORY_DELETE >
        /PROJECT/TEMP/HISTORY_ZERO

        # Determining transaction protocol
        # --------------------------------
        protocol=""
        if [[ $analyzed_transaction = udp ]]
```

```
                then
                        protocol="udp"
                fi
                if [[ $analyzed_transaction = icmp ]]
                then
                        protocol="icmp"
                fi
                if [[ $analyzed_transaction = tcp ]]
                then
                        protocol="tcp"
                fi
                # Real action for dropping canal in the Filter Table
                # -------------------------------------------------
                iptables-save > /PROJECT/TEMP/saveiptablesdrop
                > /PROJECT/TEMP/saveiptablescopydrop
                Canal_baris=`wc -l /PROJECT/TEMP/saveiptablesdrop | awk '{print $1}'`
                k=1
                while [[ $k -le $Canal_baris ]]
                do
                isikanal=`head -n $k /PROJECT/TEMP/saveiptablesdrop >
                /PROJECT/TEMP/TRANSACTIONTEMPDROP | tail -n 1
                /PROJECT/TEMP/TRANSACTIONTEMPDROP`
                echo $isikanal > /PROJECT/TEMP/ISIKANALDROP
                get_isikanal1=`grep $analyzed_external
                /PROJECT/TEMP/ISIKANALDROP | awk '{print $4}'`
                get_isikanal2=`grep $analyzed_external
                /PROJECT/TEMP/ISIKANALDROP | awk '{print $6}'`
                get_isikanal3=`grep $analyzed_external
                /PROJECT/TEMP/ISIKANALDROP | awk '{print $8}'`
                if [[ $get_isikanal1 = $analyzed_external ]] && [[ $get_isikanal2 =
                $analyzed_internal ]] && [[ $get_isikanal3 = $protocol ]]
                then
                echo DROP $protocol $analyzed_external $analyzed_internal
                else
                if [[ $get_isikanal1 = $analyzed_internal ]] && [[ $get_isikanal2 =
                $analyzed_external ]] && [[ $get_isikanal3 = $protocol ]]
                then
                echo DROP $protocol $analyzed_internal $analyzed_external
                else
                echo $isikanal >> /PROJECT/TEMP/saveiptablescopydrop
                fi
                fi
                k=`expr ${k} + 1`
                done
                cat /PROJECT/TEMP/saveiptablescopydrop | iptables-restore
                iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
                echo 1 > /proc/sys/net/ipv4/ip_forward
                #iptables -D FORWARD -i eth1 -p $protocol -s $analyzed_internal -d
                $analyzed_external -j ACCEPT
                #iptables -D FORWARD -i eth0 -p $protocol -s $analyzed_external -d
                $analyzed_internal -j ACCEPT
                > /PROJECT/TEMP/HISTORY_DELETE
                > /PROJECT/TEMP/HISTORY_TEMP_DELETE
                # echo DELETE $protocol $analyzed_internal $analyzed_external
        fi
        echo .
        echo .
    fi
done
```

APPENDIX E1

Calculations for Probability of Available Network Services

E1.1 Open-Conditions + Fuzzy-Based Security Rules Update

$$f_{initialization} : \forall c(n,m) = 1$$

$$f_{runtime} : \forall c(n,m) = 1 \Leftrightarrow protection(n) \geq risk(m)$$

$$\begin{aligned} P(AS-OF) &= P(c(n,m)=1 \mid f_{initialization}) \wedge P(c(n,m)=1 \mid f_{runtime}) \\ &= P(c(n,m)=1 \mid \forall c(n,m)=1) \wedge P(c(n,m)=1 \mid \forall c(n,m)=1 \Leftrightarrow protection(n) \geq risk(m)) \\ &= 1 \wedge 1 \\ &= 1 \end{aligned}$$

E1.2 Open-Conditions + Agent-Based Suspicious Program Detection

$$f_{initialization} : \forall c(n,m) = 1$$

$$f_{runtime} : \forall c(n,m) = 1 \Leftrightarrow th(n) = 0$$

$$\begin{aligned} P(AS-OA) &= P(c(n,m)=1 \mid f_{initialization}) \wedge P(c(n,m)=1 \mid f_{runtime}) \\ &= P(c(n,m)=1 \mid \forall c(n,m)=1) \wedge P(c(n,m)=1 \mid \forall c(n,m)=1 \Leftrightarrow th(n)=0) \\ &= 1 \wedge P(c(n,m)=1 \mid \forall c(n,m)=1 \Leftrightarrow th(n)=0) \\ &= P(c(n,m)=1 \mid \forall c(n,m)=1 \Leftrightarrow th(n)=0) \end{aligned}$$

Since the available services is measured in the normal condition, thus $th(n) = 0$.

$$P(AS-OA) = 1$$

E1.3 Lattice-Based + Fuzzy-Based Security Rules Update

$$f_{initialization} : \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m)$$

$$f_{runtime} : \forall c(n,m) = 1 \Leftrightarrow protection(n) \geq risk(m)$$

$$P(AS-LF) = P(c(n,m)=1 \mid f_{initialization}) \wedge P(c(n,m)=1 \mid f_{runtime})$$
$$= P(c(n,m)=1 \mid \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m))$$
$$\wedge P(c(n,m)=1 \mid \forall c(n,m) = 1 \Leftrightarrow protection(n) \geq risk(m))$$
$$= P(c(n,m)=1 \mid \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m)) \wedge 1$$
$$= P(c(n,m)=1 \mid \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m))$$

E1.4 Lattice-Based + Agent-Based Suspicious Program Detection

$$f_{initialization} : \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m)$$
$$f_{runtime} : \forall c(n,m) = 1 \Leftrightarrow th(n) = 0$$
$$P(AS-LA) = P(c(n,m)=1 \mid f_{initialization}) \wedge P(c(n,m)=1 \mid f_{runtime})$$
$$= P(c(n,m)=1 \mid \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m))$$
$$\wedge P(c(n,m)=1 \mid \forall c(n,m) = 1 \Leftrightarrow th(n) = 0)$$

Since the available services is measured in the normal condition, thus $th(n) = 0$.

$$P(AS-LA) = P(c(n,m)=1 \mid \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m)) \wedge 1$$
$$= P(c(n,m)=1 \mid \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m))$$

E1.5 Close-Conditions + Zero-Configuration

$$f_{initialization} : \forall c(n,m) = 0$$
$$f_{runtime} : \forall c(n,m) \in C = 1 \Leftrightarrow t_e \leq t < t_d$$

Since this method employs a mechanism to establish canals for the request of accessing Internet and maintain close connection at all time, thus “or” operator is used to determine the available services.

$$P(AS-CZ) = P(c(n,m)=1 \mid f_{initialization}) \vee P(c(n,m)=1 \mid f_{runtime})$$
$$= P(c(n,m)=1 \mid \forall c(n,m) = 0) \vee P(c(n,m)=1 \mid \forall c(n,m) \in C = 1 \Leftrightarrow t_e \leq t < t_d)$$
$$= 0 \vee P(c(n,m)=1 \mid \forall c(n,m) \in C = 1 \Leftrightarrow t_e \leq t < t_d)$$
$$= P(c(n,m)=1 \mid \forall c(n,m) \in C = 1 \Leftrightarrow t_e \leq t < t_d)$$

APPENDIX E2

Calculations for Probability of Exposed Line

E2.1 Open-Conditions + Fuzzy-Based Security Rules Update

$f_{initialization} : \forall c(n,m) = 1$

$f_{runtime} : \forall c(n,m) = 1 \Leftrightarrow protection(n) \geq risk(m)$

$P(AS-OF) = P(c(n,m) = 1 \mid \forall c(n,m) = 1) \wedge P(c(n,m) = 1 \mid \forall c(n,m) = 1 \Leftrightarrow protection(n) \geq risk(m))$

$= P(c(n,m) = 1 \mid \forall c(n,m) = 1 \Leftrightarrow protection(n) \geq risk(m))$

- For $t < t_r$ or $t > t_{ft}$,

  $P(EL-OF) = P(c(n,m) = 1 \mid \forall c(n,m) = 1 \Leftrightarrow protection(n) \geq risk(m)) = 1$

- For $th(n) > 0$, $P(EL-OF) = P(c(n,m) = 1 \mid \forall c(n,m) = 1 \Leftrightarrow protection(n) \geq risk(m)) = 1$

- For $protection(n) < risk(m)$, $P(EL-OF) = 0$

E2.2 Open-Conditions + Agent-Based Suspicious Program Detection

$f_{initialization} : \forall c(n,m) = 1$

$f_{runtime} : \forall c(n,m) = 1 \Leftrightarrow th(n) = 0$

$P(AS-OA) = P(c(n,m) = 1 \mid \forall c(n,m) = 1 \Leftrightarrow th(n) = 0)$

- For $t < t_r$ or $t > t_{ft}$, $P(EL-OA) = P(c(n,m) = 1 \mid \forall c(n,m) = 1 \Leftrightarrow th(n) = 0) = 1$

- For $th(n) > 0$, $P(EL-OA) = 0$

- For $protection(n) < risk(m)$, $P(EL-OA) = P(c(n,m) = 1 \mid \forall c(n,m) = 1 \Leftrightarrow th(n) = 0) = 1$

E2.3 Lattice-Based + Fuzzy-Based Security Rules Update

$f_{initialization} : \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m)$

$f_{runtime} : \forall c(n,m) = 1 \Leftrightarrow protection(n) \geq risk(m)$

$P(AS-LF) = P(c(n,m)=1 \mid \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m))$

- For $t < t_r$ or $t > t_{ft}$,

  $P(EL-LF) = P(c(n,m)=1 \mid \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m))$

- For $th(n) > 0$, $P(EL-LF) = P(c(n,m)=1 \mid \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m))$
- For $protection(n) < risk(m)$, $P(EL-LF) = 0$

E2.4 Lattice-Based + Agent-Based Suspicious Program Detection

$f_{initialization} : \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m)$

$f_{runtime} : \forall c(n,m) = 1 \Leftrightarrow th(n) = 0$

$P(AS-LA) = P(c(n,m)=1 \mid \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m)$

$\wedge P(c(n,m)=1 \mid \forall c(n,m) = 1 \Leftrightarrow th(n) = 0)$

- For $t < t_r$ or $t > t_{ft}$,

  $P(EL-LA) = P(c(n,m)=1 \mid \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m))$

- For $th(n) > 0$, $P(EL-LA) = 0$
- For $protection(n) < risk(m)$, $P(EL-LA) = 0$

E2.5 Close-Conditions + Zero-Configuration

$f_{initialization} : \forall c(n,m) = 0$

$f_{runtime} : \forall c(n,m) \in C = 1 \Leftrightarrow t_e \leq t < t_d$

$P(AS-CZ) = P(c(n,m)=1 \mid \forall c(n,m) \in C = 1 \Leftrightarrow t_e \leq t < t_d)$

- For $t < t_r$ or $t > t_{ft}$, $P(EL-CZ) = 0$
- For $th(n) > 0$, $P(EL-CZ) = P(c(n,m)=1 \mid \forall c(n,m) \in C = 1 \Leftrightarrow t_e \leq t < t_d)$
- For $protection(n) < risk(m)$, $P(EL-CZ) = P(c(n,m)=1 \mid \forall c(n,m) \in C = 1 \Leftrightarrow t_e \leq t < t_d)$

APPENDIX E3

Calculations for Probability of Denial of Service

E3.1 Open-Conditions + Fuzzy-Based Security Rules Update

$$f_{initialization} : \forall c(n,m) = 1$$

$$f_{runtime} : \forall c(n,m) = 1 \Leftrightarrow protection(n) \geq risk(m)$$

$$\begin{aligned} P(DS-OF) &= P(c(n,m)=0 \mid f_{initialization}) \vee P(c(n,m)=0 \mid f_{runtime}) \\ &= P(c(n,m)=0 \mid \forall c(n,m)=1) \vee P(c(n,m)=0 \mid \forall c(n,m)=1 \Leftrightarrow protection(n) \geq risk(m)) \\ &= 0 \vee 0 \\ &= 0 \end{aligned}$$

E3.2 Open-Conditions + Agent-Based Suspicious Program Detection

$$f_{initialization} : \forall c(n,m) = 1$$

$$f_{runtime} : \forall c(n,m) = 1 \Leftrightarrow th(n) = 0$$

$$\begin{aligned} P(DS-OA) &= P(c(n,m)=0 \mid f_{initialization}) \vee P(c(n,m)=0 \mid f_{runtime}) \\ &= P(c(n,m)=0 \mid \forall c(n,m)=1) \vee P(c(n,m)=0 \mid \forall c(n,m)=1 \Leftrightarrow th(n)=0) \\ &= 0 \vee 0 \\ &= 0 \end{aligned}$$

E3.3 Lattice-Based + Fuzzy-Based Security Rules Update

$$f_{initialization} : \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m)$$

$$f_{runtime} : \forall c(n,m) = 1 \Leftrightarrow protection(n) \geq risk(m)$$

$$\begin{aligned} P(DS-LF) &= P(c(n,m)=0 \mid f_{initialization}) \vee P(c(n,m)=0 \mid f_{runtime}) \\ &= P(c(n,m)=0 \mid \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m)) \\ &\vee P(c(n,m)=0 \mid \forall c(n,m)=1 \Leftrightarrow protection(n) \geq risk(m)) \\ &= P(c(n,m)=0 \mid \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m)) \vee 0 \\ &= P(c(n,m)=0 \mid \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m)) \end{aligned}$$

E3.4 Lattice-Based + Agent-Based Suspicious Program Detection

$$f_{initialization} : \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m)$$

$$f_{runtime} : \forall c(n,m) = 1 \Leftrightarrow th(n) = 0$$

$$P(DS - LA) = P(c(n,m) = 0 \mid f_{initialization}) \wedge P(c(n,m) = 0 \mid f_{runtime})$$

$$= P(c(n,m) = 0 \mid \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m)) \wedge P(c(n,m) = 0 \mid \forall c(n,m) = 1 \Leftrightarrow th(n) = 0)$$

$$= P(c(n,m) = 0 \mid \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m)) \wedge 0$$

$$= P(c(n,m) = 0 \mid \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m))$$

E3.5 Close-Conditions + Zero-Configuration

$$f_{initialization} : \forall c(n,m) = 0$$

$$f_{runtime} : \forall c(n,m) \in C = 1 \Leftrightarrow t_e \leq t < t_d$$

$$P(DS - CZ) = P(c(n,m) = 0 \mid f_{initialization}) \wedge P(c(n,m) = 0 \mid f_{runtime})$$

$$= P(c(n,m) = 0 \mid \forall c(n,m) = 0) \wedge P(c(n,m) = 0 \mid \forall c(n,m) \in C = 1 \Leftrightarrow t_e \leq t < t_d)$$

$$= 1 \wedge P(c(n,m) = 0 \mid \forall c(n,m) \in C = 1 \Leftrightarrow t_e \leq t < t_d)$$

$$= P(c(n,m) = 0 \mid \forall c(n,m) \in C = 1 \Leftrightarrow t_e \leq t < t_d)$$

APPENDIX E1

Calculations for Probability of Available Network Services

E1.1 Open-Conditions + Fuzzy-Based Security Rules Update

$$f_{initialization} : \forall c(n,m) = 1$$

$$f_{runtime} : \forall c(n,m) = 1 \Leftrightarrow protection(n) \geq risk(m)$$

$$\begin{aligned} P(AS-OF) &= P(c(n,m)=1 \mid f_{initialization}) \wedge P(c(n,m)=1 \mid f_{runtime}) \\ &= P(c(n,m)=1 \mid \forall c(n,m)=1) \wedge P(c(n,m)=1 \mid \forall c(n,m)=1 \Leftrightarrow protection(n) \geq risk(m)) \\ &= 1 \wedge 1 \\ &= 1 \end{aligned}$$

E1.2 Open-Conditions + Agent-Based Suspicious Program Detection

$$f_{initialization} : \forall c(n,m) = 1$$

$$f_{runtime} : \forall c(n,m) = 1 \Leftrightarrow th(n) = 0$$

$$\begin{aligned} P(AS-OA) &= P(c(n,m)=1 \mid f_{initialization}) \wedge P(c(n,m)=1 \mid f_{runtime}) \\ &= P(c(n,m)=1 \mid \forall c(n,m)=1) \wedge P(c(n,m)=1 \mid \forall c(n,m)=1 \Leftrightarrow th(n)=0) \\ &= 1 \wedge P(c(n,m)=1 \mid \forall c(n,m)=1 \Leftrightarrow th(n)=0) \\ &= P(c(n,m)=1 \mid \forall c(n,m)=1 \Leftrightarrow th(n)=0) \end{aligned}$$

Since the available services is measured in the normal condition, thus $th(n) = 0$.

$$P(AS-OA) = 1$$

E1.3 Lattice-Based + Fuzzy-Based Security Rules Update

$$f_{initialization} : \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m)$$

$$f_{runtime} : \forall c(n,m) = 1 \Leftrightarrow protection(n) \geq risk(m)$$

$$P(AS-LF) = P(c(n,m)=1 \mid f_{initialization}) \wedge P(c(n,m)=1 \mid f_{runtime})$$
$$= P(c(n,m)=1 \mid \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m))$$
$$\wedge P(c(n,m)=1 \mid \forall c(n,m) = 1 \Leftrightarrow protection(n) \geq risk(m))$$
$$= P(c(n,m)=1 \mid \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m)) \wedge 1$$
$$= P(c(n,m)=1 \mid \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m))$$

E1.4 Lattice-Based + Agent-Based Suspicious Program Detection

$$f_{initialization} : \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m)$$
$$f_{runtime} : \forall c(n,m) = 1 \Leftrightarrow th(n) = 0$$
$$P(AS-LA) = P(c(n,m)=1 \mid f_{initialization}) \wedge P(c(n,m)=1 \mid f_{runtime})$$
$$= P(c(n,m)=1 \mid \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m))$$
$$\wedge P(c(n,m)=1 \mid \forall c(n,m) = 1 \Leftrightarrow th(n) = 0)$$

Since the available services is measured in the normal condition, thus $th(n) = 0$.

$$P(AS-LA) = P(c(n,m)=1 \mid \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m)) \wedge 1$$
$$= P(c(n,m)=1 \mid \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m))$$

E1.5 Close-Conditions + Zero-Configuration

$$f_{initialization} : \forall c(n,m) = 0$$
$$f_{runtime} : \forall c(n,m) \in C = 1 \Leftrightarrow t_e \leq t < t_d$$

Since this method employs a mechanism to establish canals for the request of accessing Internet and maintain close connection at all time, thus “or” operator is used to determine the available services.

$$P(AS-CZ) = P(c(n,m)=1 \mid f_{initialization}) \vee P(c(n,m)=1 \mid f_{runtime})$$
$$= P(c(n,m)=1 \mid \forall c(n,m) = 0) \vee P(c(n,m)=1 \mid \forall c(n,m) \in C = 1 \Leftrightarrow t_e \leq t < t_d)$$
$$= 0 \vee P(c(n,m)=1 \mid \forall c(n,m) \in C = 1 \Leftrightarrow t_e \leq t < t_d)$$
$$= P(c(n,m)=1 \mid \forall c(n,m) \in C = 1 \Leftrightarrow t_e \leq t < t_d)$$

APPENDIX E2

Calculations for Probability of Exposed Line

E2.1 Open-Conditions + Fuzzy-Based Security Rules Update

$f_{initialization} : \forall c(n,m) = 1$

$f_{runtime} : \forall c(n,m) = 1 \Leftrightarrow protection(n) \geq risk(m)$

$P(AS-OF) = P(c(n,m) = 1 \mid \forall c(n,m) = 1) \wedge P(c(n,m) = 1 \mid \forall c(n,m) = 1 \Leftrightarrow protection(n) \geq risk(m))$

$= P(c(n,m) = 1 \mid \forall c(n,m) = 1 \Leftrightarrow protection(n) \geq risk(m))$

- For $t < t_r$ or $t > t_{ft}$,

  $P(EL-OF) = P(c(n,m) = 1 \mid \forall c(n,m) = 1 \Leftrightarrow protection(n) \geq risk(m)) = 1$

- For $th(n) > 0$, $P(EL-OF) = P(c(n,m) = 1 \mid \forall c(n,m) = 1 \Leftrightarrow protection(n) \geq risk(m)) = 1$

- For $protection(n) < risk(m)$, $P(EL-OF) = 0$

E2.2 Open-Conditions + Agent-Based Suspicious Program Detection

$f_{initialization} : \forall c(n,m) = 1$

$f_{runtime} : \forall c(n,m) = 1 \Leftrightarrow th(n) = 0$

$P(AS-OA) = P(c(n,m) = 1 \mid \forall c(n,m) = 1 \Leftrightarrow th(n) = 0)$

- For $t < t_r$ or $t > t_{ft}$, $P(EL-OA) = P(c(n,m) = 1 \mid \forall c(n,m) = 1 \Leftrightarrow th(n) = 0) = 1$

- For $th(n) > 0$, $P(EL-OA) = 0$

- For $protection(n) < risk(m)$, $P(EL-OA) = P(c(n,m) = 1 \mid \forall c(n,m) = 1 \Leftrightarrow th(n) = 0) = 1$

E2.3 Lattice-Based + Fuzzy-Based Security Rules Update

$f_{initialization} : \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m)$

$f_{runtime} : \forall c(n,m) = 1 \Leftrightarrow protection(n) \geq risk(m)$

$P(AS-LF) = P(c(n,m)=1 \mid \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m))$

- For $t < t_r$ or $t > t_{ft}$,

  $P(EL-LF) = P(c(n,m)=1 \mid \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m))$

- For $th(n) > 0$, $P(EL-LF) = P(c(n,m)=1 \mid \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m))$
- For $protection(n) < risk(m)$, $P(EL-LF) = 0$

E2.4 Lattice-Based + Agent-Based Suspicious Program Detection

$f_{initialization} : \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m)$

$f_{runtime} : \forall c(n,m) = 1 \Leftrightarrow th(n) = 0$

$P(AS-LA) = P(c(n,m)=1 \mid \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m)$

$\wedge P(c(n,m)=1 \mid \forall c(n,m) = 1 \Leftrightarrow th(n) = 0)$

- For $t < t_r$ or $t > t_{ft}$,

  $P(EL-LA) = P(c(n,m)=1 \mid \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m))$

- For $th(n) > 0$, $P(EL-LA) = 0$
- For $protection(n) < risk(m)$, $P(EL-LA) = 0$

E2.5 Close-Conditions + Zero-Configuration

$f_{initialization} : \forall c(n,m) = 0$

$f_{runtime} : \forall c(n,m) \in C = 1 \Leftrightarrow t_e \leq t < t_d$

$P(AS-CZ) = P(c(n,m)=1 \mid \forall c(n,m) \in C = 1 \Leftrightarrow t_e \leq t < t_d)$

- For $t < t_r$ or $t > t_{ft}$, $P(EL-CZ) = 0$
- For $th(n) > 0$, $P(EL-CZ) = P(c(n,m)=1 \mid \forall c(n,m) \in C = 1 \Leftrightarrow t_e \leq t < t_d)$
- For $protection(n) < risk(m)$, $P(EL-CZ) = P(c(n,m)=1 \mid \forall c(n,m) \in C = 1 \Leftrightarrow t_e \leq t < t_d)$

APPENDIX E3

Calculations for Probability of Denial of Service

E3.1 Open-Conditions + Fuzzy-Based Security Rules Update

$$f_{initialization} : \forall c(n,m) = 1$$

$$f_{runtime} : \forall c(n,m) = 1 \Leftrightarrow protection(n) \geq risk(m)$$

$$\begin{aligned} P(DS-OF) &= P(c(n,m)=0 \mid f_{initialization}) \vee P(c(n,m)=0 \mid f_{runtime}) \\ &= P(c(n,m)=0 \mid \forall c(n,m)=1) \vee P(c(n,m)=0 \mid \forall c(n,m)=1 \Leftrightarrow protection(n) \geq risk(m)) \\ &= 0 \vee 0 \\ &= 0 \end{aligned}$$

E3.2 Open-Conditions + Agent-Based Suspicious Program Detection

$$f_{initialization} : \forall c(n,m) = 1$$

$$f_{runtime} : \forall c(n,m) = 1 \Leftrightarrow th(n) = 0$$

$$\begin{aligned} P(DS-OA) &= P(c(n,m)=0 \mid f_{initialization}) \vee P(c(n,m)=0 \mid f_{runtime}) \\ &= P(c(n,m)=0 \mid \forall c(n,m)=1) \vee P(c(n,m)=0 \mid \forall c(n,m)=1 \Leftrightarrow th(n)=0) \\ &= 0 \vee 0 \\ &= 0 \end{aligned}$$

E3.3 Lattice-Based + Fuzzy-Based Security Rules Update

$$f_{initialization} : \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m)$$

$$f_{runtime} : \forall c(n,m) = 1 \Leftrightarrow protection(n) \geq risk(m)$$

$$\begin{aligned} P(DS-LF) &= P(c(n,m)=0 \mid f_{initialization}) \vee P(c(n,m)=0 \mid f_{runtime}) \\ &= P(c(n,m)=0 \mid \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m)) \\ &\vee P(c(n,m)=0 \mid \forall c(n,m)=1 \Leftrightarrow protection(n) \geq risk(m)) \\ &= P(c(n,m)=0 \mid \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m)) \vee 0 \\ &= P(c(n,m)=0 \mid \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m)) \end{aligned}$$

E3.4 Lattice-Based + Agent-Based Suspicious Program Detection

$$f_{initialization} : \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m)$$

$$f_{runtime} : \forall c(n,m) = 1 \Leftrightarrow th(n) = 0$$

$$P(DS - LA) = P(c(n,m) = 0 \mid f_{initialization}) \wedge P(c(n,m) = 0 \mid f_{runtime})$$

$$= P(c(n,m) = 0 \mid \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m)) \wedge P(c(n,m) = 0 \mid \forall c(n,m) = 1 \Leftrightarrow th(n) = 0)$$

$$= P(c(n,m) = 0 \mid \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m)) \wedge 0$$

$$= P(c(n,m) = 0 \mid \forall c(n,m) \in C = 1 \Leftrightarrow protection(n) \geq risk(m))$$

E3.5 Close-Conditions + Zero-Configuration

$$f_{initialization} : \forall c(n,m) = 0$$

$$f_{runtime} : \forall c(n,m) \in C = 1 \Leftrightarrow t_e \leq t < t_d$$

$$P(DS - CZ) = P(c(n,m) = 0 \mid f_{initialization}) \wedge P(c(n,m) = 0 \mid f_{runtime})$$

$$= P(c(n,m) = 0 \mid \forall c(n,m) = 0) \wedge P(c(n,m) = 0 \mid \forall c(n,m) \in C = 1 \Leftrightarrow t_e \leq t < t_d)$$

$$= 1 \wedge P(c(n,m) = 0 \mid \forall c(n,m) \in C = 1 \Leftrightarrow t_e \leq t < t_d)$$

$$= P(c(n,m) = 0 \mid \forall c(n,m) \in C = 1 \Leftrightarrow t_e \leq t < t_d)$$

APPENDIX F1

Static Firewall Script

```
#!/bin/sh
#
# firewall      Firewall startup/shutdown script
#
# Version:      @(#) /etc/rc.d/init.d/firewall.iptables  08-April-2005
#
#
# Translated to iptables format, with several additions and modifications,
# from Craig Zeller's (zeller@fatpenguin.com) ipchains-based firewall script, by
# Bob Sully (rcs@malibyte.net)
#
# Thanks to Jeff Carlson (jeff@ultimateevil.org) for his assistance re: DHCP and several other issues,
# Rohan Amin (rohan@rohanamin.com) and Erik Wasser (erik.wasser@iquer.com) for help with the port-forwarding
# routine, and Nate Waddoups for his quick PPTP hack.
#
# Latest revision: 08-Apr-2005
#
# chkconfig: 345 11 91
#
# description: IP Firewall startup/shutdown script for iptables
#
# probe: true
#
#
# CONSTANTS - Do not edit
#
ANYWHERE="0.0.0.0/0"                            # Match any IP address
BROADCAST_SRC="0.0.0.0"                                 # Broadcast Source Address
BROADCAST_DEST="255.255.255.255"                # Broadcast Destination Address
CLASS_A="10.0.0.0/8"                            # Class-A Private (RFC-1918) Networks
CLASS_B="172.16.0.0/12"                         # Class-B Private (RFC-1918) Networks
CLASS_C="192.168.0.0/16"                        # Class-C Private (RFC-1918) Networks
CLASS_D_MULTICAST="224.0.0.0/4"                 # Class-D Multicast Addresses
CLASS_E_RESERVED_NET="240.0.0.0/5"              # Class-E Reserved Addresses
PRIVPORTS="0:1023"                              # Well-Known, Privileged Port Range
UNPRIVPORTS="1024:65535"                        # Unprivileged Port Range
TRACEROUTE_SRC_PORTS="32769:65535"              # Traceroute Source Ports
TRACEROUTE_DEST_PORTS="33434:33523"             # Traceroute Destination Ports
#
# The Loopback interface defines should not be
# edited unless your Linux distribution defines
# these differently.
#
LOOPBACK_INTERFACE="lo"                                 # The loopback interface
LOOPBACK_NETWORK="127.0.0.0/8"                  # Reserved Loopback Address Range
#
# Source function library.
#
. /etc/rc.d/init.d/functions
#
# See how we were called.
#
case "$1" in
  start)
        echo "Starting Firewall services"
                echo "firewall: Configuring Firewall Rules using iptables"
                # Remove any existing rules from all chains
                iptables -F
                iptables -F -t nat
                iptables -F -t mangle
                # Set the default policy to drop
                iptables -P INPUT   DROP
                iptables -P OUTPUT  DROP
                iptables -P FORWARD DROP
                # Allow unlimited traffic on the loopback interface
                iptables -A INPUT -i $LOOPBACK_INTERFACE -j ACCEPT
                iptables -A OUTPUT -o $LOOPBACK_INTERFACE -j ACCEPT
                # A bug that showed up as of the Red Hat 7.2 release results in
                # the following 5 default policies breaking the firewall
                # initialization:
                #     fgrep -q '7.2' /etc/redhat-release
                #     if [ $? -ne 0 ] ; then
                #       iptables -t nat    -P PREROUTING  DROP
```

```
            #       iptables -t nat    -P OUTPUT      DROP
            #       iptables -t nat    -P POSTROUTING DROP
            #       iptables -t mangle -P PREROUTING  DROP
            #       iptables -t mangle -P OUTPUT      DROP
            #     fi
            # Remove any pre-existing user-defined chains
            iptables -X
            iptables -X -t nat
            iptables -X -t mangle
            # Zero counts
            iptables -Z
            # Open the configuration file
            if [ -f /etc/firewall/firewall.conf.iptables ]; then
               . /etc/firewall/firewall.conf.iptables
            else
               # Turn off IP Forwarding & Masquerading
               echo 0 >/proc/sys/net/ipv4/ip_forward
               # Turn off dynamic IP hacking
   echo "0" > /proc/sys/net/ipv4/ip_dynaddr

               if [ $MASQUERADING -gt 0 ]; then
                  # Allow unlimited local traffic on the internal interface
                  iptables -A INPUT -i $INTERNAL_INTERFACE -j ACCEPT
                  iptables -A OUTPUT -o $INTERNAL_INTERFACE -j ACCEPT
               fi
               echo "firewall: No configuration file found at /etc/firewall/firewall.conf.iptables; "
               echo "firewall: default policies set to DROP on INPUT/OUTPUT/FORWARD chains."
               exit 1
            fi
#
# If your IP address is dynamically assigned by a DHCP server,
# your DHCP server's IP address and this machine's IP address are
# obtained from /etc/dhcpc/hostinfo-$EXTERNAL_INTERFACE or
# /etc/dhcpc/dhcpcd-$EXTERNAL_INTERFACE.info.
#
if [ $DHCP -gt 0 ]; then
               # Grab external IP address if already assigned
  EXTERNAL_IP=$( ifconfig $EXTERNAL_INTERFACE | grep 'inet[^6]' | sed 's/[a-zA-Z:]//g' | awk '{print $1}' )
               if [ -n $EXTERNAL_IP ]; then
    EXT_NETMASK=$( ifconfig $EXTERNAL_INTERFACE | grep 'inet[^6]' | sed 's/[a-zA-Z:]//g' | awk '{print $3}' )
    EXTERNAL_NETWORK=$( ipcalc -n $EXTERNAL_IP $EXT_NETMASK | cut -d\= -f2 )
    BROADCAST_NET=$( ipcalc -b $EXTERNAL_IP $EXT_NETMASK | cut -d\= -f2 )
               fi
  # Turn on dynamic IP hacking
  echo "1" > /proc/sys/net/ipv4/ip_dynaddr
  # Incoming DHCPOFFER from available DHCP servers
  iptables -A INPUT -i $EXTERNAL_INTERFACE -p udp \
          -s 0.0.0.0        --sport 67 \
          -d 255.255.255.255 --dport 68 -j ACCEPT
  # Initialization of rebinding: No lease or Lease time expired.
  iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p udp   \
          -s 0.0.0.0        --sport 68 \
          -d 255.255.255.255 --dport 67 -j ACCEPT
  # Fall back to initialization
  # The client knows its server, but has either lost its
  # lease, or else needs to reconfirm the IP address after
  # rebooting.
  iptables -A INPUT -i $EXTERNAL_INTERFACE -p udp \
          -s $DHCP_SERVER_IP --sport 67     \
          -d 255.255.255.255 --dport 68 -j ACCEPT
  iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p udp \
          -s 255.255.255.255 --sport 68 \
          -d $DHCP_SERVER_IP --dport 67 -j ACCEPT
  # As a result of the above, we're supposed to change our IP
  # address with this message, which is addressed to our new
  # address before the dhcp client has received the update.
  # Depending on the server implementation, the destination
  # address can be the new IP address, the subnet address, or
  # the limited broadcast address.
  # If the network subnet address is used as the destination,
  # the next rule must allow incoming packets destined to the
  # subnet address, and the rule must preceed any general rules
  # that block such incoming broadcast packets.
  iptables -A INPUT -i $EXTERNAL_INTERFACE -p udp \
          -s $DHCP_SERVER_IP --sport 67     \
```

```
        --dport 68 -j ACCEPT
    # Lease renewal
    iptables -A INPUT -i $EXTERNAL_INTERFACE -p udp \
        -s $DHCP_SERVER_IP --sport 67    \
        -d $EXTERNAL_IP --dport 68 -j ACCEPT
    iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p udp \
        -s $EXTERNAL_IP --sport 68      \
        -d $DHCP_SERVER_IP --dport 67 -j ACCEPT
    echo "firewall: DHCP Client configured"
          else
            # External IP assigned without DHCP (i.e. static); get some more info
      EXT_NETMASK=$( ifconfig $EXTERNAL_INTERFACE | grep 'inet[^6]' | sed 's/[a-zA-Z:]//g' | awk '{print $3}' )
      EXTERNAL_NETWORK=$( ipcalc -n $EXTERNAL_IP $EXT_NETMASK | cut -d\= -f2 )
      BROADCAST_NET=$( ipcalc -b $EXTERNAL_IP $EXT_NETMASK | cut -d\= -f2 )
  fi
        #
        # Refuse directed broadcasts; you may choose not to log these, as they can fill up your logs quickly
        #
#       iptables -A INPUT -i $EXTERNAL_INTERFACE -d $EXTERNAL_NETWORK \
#     -m limit --limit 1/s      \
#     -j LOG --log-prefix "[Directed Broadcast] "
        iptables -A INPUT -i $EXTERNAL_INTERFACE -d $EXTERNAL_NETWORK -j DROP
#       iptables -A INPUT -i $EXTERNAL_INTERFACE -d $BROADCAST_NET \
#     -m limit --limit 1/s      \
#     -j LOG --log-prefix "[Directed Broadcast] "
        iptables -A INPUT  -i $EXTERNAL_INTERFACE -d $BROADCAST_NET -j DROP
        # Refuse limited broadcasts
#       iptables -A INPUT  -i $EXTERNAL_INTERFACE -d 255.255.255.255 \
#     -m limit --limit 1/s      \
#     -j LOG --log-prefix "[Limited Broadcast] "
        iptables -A INPUT  -i $EXTERNAL_INTERFACE -d 255.255.255.255 -j DROP
        #
        # Edit these to match the number of servers or connections
        # you support.
        #
        # X Window port allocation begins at 6000 and increments
        # for each additional server running from 6000 to 6063.
        XWINDOW_PORTS="6000:6063"                       # (TCP) X Windows
        # SSH starts at 1023 and works down to 513 for each additional
        # simultaneous incoming connection.
        SSH_HI_PORTS="513:1023"                                 # SSH Simultaneous Connections
        #
        # Iptables allows creation of customized chains.  The -l (log) flag no longer
        # exists.  This is a custom chain which allows logging of DROPped packets.
        #
        iptables -N LnD                                 # Define custom DROP chain
        iptables -A LnD -p tcp -m limit --limit 1/s -j LOG --log-prefix "[TCP drop] " --log-level=info
        iptables -A LnD -p udp -m limit --limit 1/s -j LOG --log-prefix "[UDP drop] " --log-level=info
        iptables -A LnD -p icmp -m limit --limit 1/s -j LOG --log-prefix "[ICMP drop] " --log-level=info
        iptables -A LnD -f -m limit --limit 1/s -j LOG --log-prefix "[FRAG drop] " --log-level=info
        iptables -A LnD -j DROP
        #
        # This custom chain logs, then REJECTs packets.
        #
        iptables -N LnR                                 # Define custom REJECT chain
        iptables -A LnR -p tcp -m limit --limit 1/s -j LOG --log-prefix "[TCP reject] " --log-level=info
        iptables -A LnR -p udp -m limit --limit 1/s -j LOG --log-prefix "[UDP reject] " --log-level=info
        iptables -A LnR -p icmp -m limit --limit 1/s -j LOG --log-prefix "[ICMP reject] " --log-level=info
        iptables -A LnR -f -m limit --limit 1/s -j LOG --log-prefix "[FRAG reject] " --log-level=info
        iptables -A LnR -j REJECT
        #
        # This chain logs, then DROPs "Xmas" and Null packets which might indicate a port-scan attempt
        #
        iptables -N ScanD                       # Define custom chain for possible port-scans
        iptables -A ScanD -p tcp -m limit --limit 1/s -j LOG --log-prefix "[TCP Scan?] "
        iptables -A ScanD -p udp -m limit --limit 1/s -j LOG --log-prefix "[UDP Scan?] "
        iptables -A ScanD -p icmp -m limit --limit 1/s -j LOG --log-prefix "[ICMP Scan?] "
        iptables -A ScanD -f -m limit --limit 1/s -j LOG --log-prefix "[FRAG Scan?] "
        iptables -A ScanD -j DROP
        #
  # This chain limits the number of new incoming connections to preventing DDoS attacks
  #
  iptables -N DDoS              # Define custom chain for possible DDoS attacks
  iptables -A DDoS -m limit --limit 12/s --limit-burst 24 -j RETURN
        iptables -A DDoS -j LOG --log-prefix "[DDos Attack?] "
        iptables -A DDoS -j DROP
```

```
        #
# This chain drops connections from IANA reserved IP blocks
#
        iptables -N IANA
        iptables -A IANA -p tcp -m limit --limit 1/s -j LOG --log-prefix "[IANA Reserved - TCP] " --log-level=info
        iptables -A IANA -p udp -m limit --limit 1/s -j LOG --log-prefix "[IANA Reserved - UDP] " --log-level=info
        iptables -A IANA -p icmp -m limit --limit 1/s -j LOG --log-prefix "[IANA Reserved - ICMP] " --log-level=info
        iptables -A IANA -f -m limit --limit 1/s -j LOG --log-prefix "[IANA Reserved - FRAG] " --log-level=info
        iptables -A IANA -j DROP
        #
# This chain drops connections from IPs in the firewall.banned file
#
        iptables -N Banned
        iptables -A Banned -p tcp -m limit --limit 1/s -j LOG --log-prefix "[TCP Banned] " --log-level=info
        iptables -A Banned -p udp -m limit --limit 1/s -j LOG --log-prefix "[UDP Banned] " --log-level=info
        iptables -A Banned -p icmp -m limit --limit 1/s -j LOG --log-prefix "[ICMP Banned] " --log-level=info
        iptables -A Banned -f -m limit --limit 1/s -j LOG --log-prefix "[FRAG Banned] " --log-level=info
        iptables -A Banned -j DROP
#
# Disallow packets frequently used by port-scanners
#
        # All of the bits are cleared
        iptables -A INPUT -p tcp --tcp-flags ALL NONE -j ScanD
        # SYN and FIN are both set
        iptables -A INPUT -p tcp --tcp-flags SYN,FIN SYN,FIN -j ScanD
        # SYN and RST are both set
        iptables -A INPUT -p tcp --tcp-flags SYN,RST SYN,RST -j ScanD
        # FIN and RST are both set
        iptables -A INPUT -p tcp --tcp-flags FIN,RST FIN,RST -j ScanD
        # FIN is the only bit set, without the expected accompanying ACK
        iptables -A INPUT -p tcp --tcp-flags ACK,FIN FIN -j ScanD
        # PSH is the only bit set, without the expected accompanying ACK
        iptables -A INPUT -p tcp --tcp-flags ACK,PSH PSH -j ScanD
        # URG is the only bit set, without the expected accompanying ACK
        iptables -A INPUT -p tcp --tcp-flags ACK,URG URG -j ScanD
        # SYN-Flood
        # (Request for new connection; large number indicate possible DDoS-type attack;
        #  same as --syn)
        iptables -A INPUT -p tcp --tcp-flags SYN,RST,ACK SYN -j DDoS
        # Enable broadcast echo Protection
        echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
        # Disable Source Routed Packets
        for f in /proc/sys/net/ipv4/conf/*/accept_source_route; do
          echo 0 > $f
        done
        # Enable TCP SYN Cookie Protection
        echo 1 > /proc/sys/net/ipv4/tcp_syncookies
        # Disable ICMP Redirect Acceptance
        for f in /proc/sys/net/ipv4/conf/*/accept_redirects; do
          echo 0 > $f
        done
        # Don't send Redirect Messages
        for f in /proc/sys/net/ipv4/conf/*/send_redirects; do
          echo 0 > $f
        done
        # Disable ICMP Redirect Acceptance
        for f in /proc/sys/net/ipv4/conf/*/accept_redirects; do
           echo 0 > $f
        done
        # Drop Spoofed Packets coming in on an interface, which if replied to,
        # would result in the reply going out a different interface.
        for f in /proc/sys/net/ipv4/conf/*/rp_filter; do
          echo 1 > $f
        done
        # Log packets with impossible addresses.
        for f in /proc/sys/net/ipv4/conf/*/log_martians; do
          echo 1 > $f
        done
        # Disallow fragmented packets.  This may not be as necessary as it once was.
        # Comment it out with # if desired.
#       iptables -A INPUT -f -i $EXTERNAL_INTERFACE -j LnD
#       iptables -A INPUT -f -i $INTERNAL_INTERFACE -j LnD
        #
        # Loopback
        #
        # Unlimited traffic on the loopback interface (lo)
```

```
    iptables -A INPUT -i $LOOPBACK_INTERFACE -j ACCEPT
    iptables -A OUTPUT -o $LOOPBACK_INTERFACE -j ACCEPT
    #
    # Refuse any connections to/from problem sites.
    #
    # /etc/firewall/firewall.banned contains a list of IPs
    # to block all access, both inbound and outbound.
    # The file should contain IP addresses with CIDR
    # netmask, one per line:
    #
    # NOTE: No comments are allowed in the file.
    #
    # 111.222.333.444/32              - To block a single IP address
    # 111.222.333.444/8               - To block a Class-A network
    # 111.222.333.444/16              - To block a Class-B network
    # 111.222.333.444/24              - To block a Class-C network
    #
    # The CIDR netmask number describes the number of bits
    # in the network portion of the address, and may be on
    # any boundary.
    #
    if [ -f /etc/firewall/firewall.banned ]; then
       while read BANNED; do
                iptables -A INPUT -i $EXTERNAL_INTERFACE -s $BANNED -j Banned
                iptables -A INPUT -i $EXTERNAL_INTERFACE -d $BANNED -j Banned
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -s $BANNED -j Banned
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -d $BANNED -j Banned
                iptables -A FORWARD -d $BANNED -j Banned
iptables -A FORWARD -s $BANNED -j Banned
       done < /etc/firewall/firewall.banned
       echo "firewall: Banned addresses added to rule set"
    else
       echo "firewall: Banned address/network file not found."
    fi
    #
    # Refuse connections from IANA-reserved blocks
    #
    if [ -f /etc/firewall/firewall.iana-reserved ]; then
       while read RESERVED; do
                iptables -A INPUT -i $EXTERNAL_INTERFACE -s $RESERVED -j IANA
                iptables -A INPUT -i $EXTERNAL_INTERFACE -d $RESERVED -j IANA
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -s $RESERVED -j IANA
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -d $RESERVED -j IANA
       done < /etc/firewall/firewall.iana-reserved
       echo "firewall: Connections from IANA-reserved addresses blocked"
    else
       echo "firewall: IANA-reserved address/network file not found."
    fi
    #
    # Localizations
    #
    # The /etc/firewall/firewall.local file should contain rules in
    # standard 'iptables' format.
    #
    if [ -f /etc/firewall/firewall.local.iptables ]; then
       . /etc/firewall/firewall.local.iptables
       echo "firewall: Local rules added"
    else
       echo "firewall: Local rules file not found."
    fi
    #
    # ICMP
    #
    # (4) Source Quench.
    # Incoming & outgoing requests to slow down (flow control)
    iptables -A INPUT -i $EXTERNAL_INTERFACE -p ICMP --icmp-type 4 \
       -s $ANYWHERE  -d $EXTERNAL_IP -j ACCEPT
    iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p ICMP --icmp-type 4 \
       -s $EXTERNAL_IP  -d $ANYWHERE -j ACCEPT
    if [ $MASQUERADING -gt 0 ]; then
       iptables -A FORWARD -p ICMP --icmp-type 4 -j ACCEPT
    fi
    # (12) Parameter Problem.
    # Incoming & outgoing error messages
    iptables -A INPUT -i $EXTERNAL_INTERFACE -p ICMP --icmp-type 12 \
       -s $ANYWHERE  -d $EXTERNAL_IP -j ACCEPT
```

```
iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p ICMP --icmp-type 12 \
   -s $EXTERNAL_IP  -d $ANYWHERE -j ACCEPT
if [ $MASQUERADING -gt 0 ]; then
   iptables -A FORWARD -p ICMP --icmp-type 12 -j ACCEPT
fi
# (3) Destination Unreachable, Service Unavailable.
# Incoming & outgoing size negotiation, service or
# destination unavailability, final traceroute response
iptables -A INPUT -i $EXTERNAL_INTERFACE -p ICMP --icmp-type 3 \
   -s $ANYWHERE  -d $EXTERNAL_IP -j ACCEPT
iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p ICMP --icmp-type 3 \
   -s $EXTERNAL_IP  -d $ANYWHERE -j ACCEPT
iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p ICMP --icmp-type \
   fragmentation-needed -s $EXTERNAL_IP  -d $ANYWHERE -j ACCEPT
if [ $MASQUERADING -gt 0 ]; then
   iptables -A FORWARD -p ICMP --icmp-type 3 -j ACCEPT
   iptables -A FORWARD -p ICMP --icmp-type fragmentation-needed -j ACCEPT
fi
# (11) Time Exceeded.
# Incoming & outgoing timeout conditions,
# also intermediate TTL response to traceroutes
iptables -A INPUT -i $EXTERNAL_INTERFACE -p ICMP --icmp-type 11 \
   -s $ANYWHERE  -d $EXTERNAL_IP -j ACCEPT
iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p ICMP --icmp-type 11 \
   -s $EXTERNAL_IP  -d $ANYWHERE -j ACCEPT
if [ $MASQUERADING -gt 0 ]; then
   iptables -A FORWARD -p ICMP --icmp-type 11 -j ACCEPT
fi
# (0 | 8) Allow OUTPUT pings to anywhere.
if [ $OUTBOUND_PING -gt 0 ]; then
   iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p ICMP --icmp-type 8 \
               -s $EXTERNAL_IP  -d $ANYWHERE -j ACCEPT
   iptables -A INPUT -i $EXTERNAL_INTERFACE -p ICMP --icmp-type 0 \
               -s $ANYWHERE  -d $EXTERNAL_IP -j ACCEPT
   if [ $MASQUERADING -gt 0 ]; then
     iptables -A FORWARD -p ICMP --icmp-type 8 -s $INTERNAL_NETWORK -j ACCEPT
     iptables -A FORWARD -p ICMP --icmp-type 0 -d $INTERNAL_NETWORK -j ACCEPT
   fi
   if [ $VERBOSE -gt 0 ]; then
               echo "firewall: Outbound ping enabled"
   fi
fi
# (0 | 8) Allow incoming pings from anywhere
#       (stops at firewall).
if [ $INBOUND_PING -gt 0 ]; then
   iptables -A INPUT -i $EXTERNAL_INTERFACE -p ICMP --icmp-type 8 \
               -s $ANYWHERE  -d $EXTERNAL_IP -j ACCEPT
   iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p ICMP --icmp-type 0 \
               -s $EXTERNAL_IP  -d $ANYWHERE -j ACCEPT
   if [ $VERBOSE -gt 0 ]; then
               echo "firewall: Inbound ping enabled"
   fi
fi
#
# Unprivileged Ports
# Avoid ports subject to protocol and system administration problems.
#

NFS_PORT="2049"                                          # (TCP/UDP) NFS
OPENWINDOWS_PORT="2000"                                  # (TCP) Openwindows
SOCKS_PORT="1080"                                # (TCP) Socks
# Openwindows: establishing a connection
iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP -m state --state NEW \
--dport $OPENWINDOWS_PORT -s $EXTERNAL_IP -d $ANYWHERE -j LnR
# Openwindows: incoming connection
iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state NEW \
--dport $OPENWINDOWS_PORT -d $EXTERNAL_IP -j LnD
# X Window: establishing a remote connection
iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP -m state --state NEW \
--dport $XWINDOW_PORTS -s $EXTERNAL_IP -d $ANYWHERE -j LnR
# X Window: incoming connection attempt
iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state NEW \
--dport $XWINDOW_PORTS -d $EXTERNAL_IP -j LnD
# SOCKS: establishing a connection
iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP -m state --state NEW \
--dport $SOCKS_PORT -s $EXTERNAL_IP -d $ANYWHERE  -j LnR
```

```
                # SOCKS: incoming connection
                iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state NEW \
                --dport         $SOCKS_PORT -d $EXTERNAL_IP -j LnD
                # NFS: TCP connections
                iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state NEW \
                --dport $NFS_PORT -d $EXTERNAL_IP -j LnD
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP -m state --state NEW \
                --dport         $NFS_PORT -d $ANYWHERE -j LnR
                # NFS: UDP connections
                iptables -A INPUT -i $EXTERNAL_INTERFACE -p UDP \
                --dport $NFS_PORT -d $EXTERNAL_IP -j LnD
                # NFS: incoming request (normal UDP mode)
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p UDP \
                --dport $NFS_PORT -d $ANYWHERE -j LnR
        #
        # DNAT/SNAT Port Forwarding
        #
                if [ $PORT_FORWARD -gt 0 ]; then
          if [ -f /etc/firewall/firewall.nat ]; then
            while read IP_PORT; do
              # extract the protocols, IPs and ports
                        NAT_TYPE=$(echo "$IP_PORT" | awk '{print $1}')
              NAT_EXT_PORT=$(echo "$IP_PORT" | awk '{print $2}')
              NAT_INT_IP=$(echo "$IP_PORT" | awk '{print $3}')
              NAT_INT_PORT=$(echo "$IP_PORT" | awk '{print $4}')
              # write the rules!
              # this is the prerouting dnat
              iptables -A PREROUTING -t nat -p $NAT_TYPE -d $EXTERNAL_IP --dport $NAT_EXT_PORT -j DNAT \
               --to-destination $NAT_INT_IP:$NAT_INT_PORT
              # This allows packets from external->internal
              iptables -A FORWARD -i $EXTERNAL_INTERFACE -o $INTERNAL_INTERFACE -p $NAT_TYPE  \
               -d $NAT_INT_IP --dport $NAT_INT_PORT -m state \
               --state NEW,ESTABLISHED,RELATED -j ACCEPT
              # This allows packets from internal->external
              iptables -A FORWARD -i $INTERNAL_INTERFACE -o $EXTERNAL_INTERFACE -p $NAT_TYPE  \
               -s $NAT_INT_IP --sport $NAT_INT_PORT -m state \
               --state NEW,ESTABLISHED,RELATED -j ACCEPT
              # This enables access to the 'public' server from the internal network
              iptables -t nat -A POSTROUTING -d $NAT_INT_IP -s $INTERNAL_NETWORK \
               -p $NAT_TYPE --dport $NAT_INT_PORT -j SNAT --to $INTERNAL_IP
              echo firewall: dnat: $NAT_TYPE:$EXTERNAL_IP:$NAT_EXT_PORT - $NAT_INT_IP:$NAT_INT_PORT
            done < /etc/firewall/firewall.nat
           # unset some variables
            unset IP_PORT
                      unset NAT_TYPE
            unset NAT_EXT_PORT
            unset NAT_INT_IP
            unset NAT_INT_PORT
                    else
            echo "firewall.nat (port-forwarding table) not found!  Port-forwarding not enabled."
                      fi
        fi
                #
                # NOTE:
                #     The symbolic names used in /etc/services for the port numbers
                #     vary by supplier.
                #
                # Required Services
                #
                # DNS client modes (53)
                #
                if [ $DNS_CLIENT -gt 0 ]; then
                   iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p UDP \
                                --sport $UNPRIVPORTS --dport 53 -s $EXTERNAL_IP \
                                -d $ANYWHERE -j ACCEPT
                   iptables -A INPUT -i $EXTERNAL_INTERFACE -m state --state ESTABLISHED,RELATED -p UDP --
sport 53 \
                                --dport $UNPRIVPORTS -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT
                   if [ $MASQUERADING -gt 0 ]; then
                     iptables -A FORWARD -p UDP -s $INTERNAL_NETWORK --sport $UNPRIVPORTS --dport 53 -j
ACCEPT
                     iptables -A FORWARD -p UDP -d $INTERNAL_NETWORK --sport 53 --dport $UNPRIVPORTS -j
ACCEPT
                   fi
                   # TCP client-to-server requests are allowed by the protocol
                   # if UDP requests fail. This is rarely seen. Usually, clients
```

```
                # use TCP as a secondary name server for zone transfers from
                # their primary name servers, and as hackers.
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP --sport \
                        $UNPRIVPORTS --dport 53 -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT
                iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
                        --sport 53 --dport $UNPRIVPORTS -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT
                if [ $MASQUERADING -gt 0 ]; then
                  iptables -A FORWARD -p TCP -s $INTERNAL_NETWORK --sport $UNPRIVPORTS --dport 53 -j
ACCEPT
                  iptables -A FORWARD -m state --state ESTABLISHED,RELATED -p TCP -d $INTERNAL_NETWORK \
                        --sport 53 --dport $UNPRIVPORTS -j ACCEPT
                fi
                if [ $VERBOSE -gt 0 ]; then
                        echo "firewall: DNS client enabled"
                fi
              fi
              #
              # DNS server modes (53)
              #
              #
              # DNS caching & forwarding name server
              #
              if [ $DNS_CACHING_SERVER -gt 0 ]; then
                # Server-to-server query or response
                # Caching only name server uses UDP, not TCP
                iptables -A INPUT -i $EXTERNAL_INTERFACE -p UDP \
              --sport 53 --dport 53 -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p UDP \
              --sport 53 --dport 53 -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT
                if [ $VERBOSE -gt 0 ]; then
                        echo "firewall: DNS Caching server enabled"
                fi
              fi
              #
              # DNS full name server
              #
              if [ $DNS_FULL_SERVER -gt 0 ]; then
                # Client-to-server DNS transaction.
                iptables -A INPUT -i $EXTERNAL_INTERFACE -p UDP \
              --sport $UNPRIVPORTS --dport 53 -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p UDP \
              --sport 53 --dport $UNPRIVPORTS -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT
                # Zone Transfers.
                # Due to the potential danger of zone transfers,
                # allow TCP traffic to only specific secondaries.
          # /etc/firewall/firewall.dns contains a list of
          # secondary, tertiary, etc. domain name servers with which
          # zone transfers are allowed.  The file should contain IP
          # addresses with CIDR netmask, one per line:
        if [ -f /etc/firewall/firewall.dns ]; then
            while read DNS_SECONDARY; do
          iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP \
            --sport $UNPRIVPORTS --dport 53 -s $DNS_SECONDARY -d $EXTERNAL_IP -j ACCEPT
          iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
            --sport 53 --dport $UNPRIVPORTS -s $EXTERNAL_IP -d $DNS_SECONDARY -j ACCEPT
          done < /etc/firewall/firewall.dns
        else
          echo "firewall: ** No secondary DNS configured **"
        fi
          if [ $VERBOSE -gt 0 ]; then
            echo "firewall: DNS Full server enabled"
          fi
        fi
              #
              # AUTH (113) - Allowing your outgoing AUTH requests as a client
              #
              if [ $AUTH_CLIENT -gt 0 ]; then
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP \
              --sport $UNPRIVPORTS --dport 113 -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT
                iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
              --sport 113 --dport $UNPRIVPORTS -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT
                if [ $MASQUERADING -gt 0 ]; then
                  iptables -A FORWARD -p TCP -s $INTERNAL_NETWORK --sport $UNPRIVPORTS --dport 113 -j
ACCEPT
                  iptables -A FORWARD -m state --state ESTABLISHED,RELATED -p TCP -d $INTERNAL_NETWORK \
                        --sport 113 --dport $UNPRIVPORTS -j ACCEPT
```

```
                fi
                if [ $VERBOSE -gt 0 ]; then
                        echo "firewall: Auth client enabled"
                fi
              fi
              # AUTH server (113)
              if [ $AUTH_SERVER -gt 0 ]; then
                # Accepting incoming AUTH requests
                iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP \
              --sport $UNPRIVPORTS --dport 113 -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
              --sport 113 --dport $UNPRIVPORTS -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT
                if [ $VERBOSE -gt 0 ]; then
                        echo "firewall: Auth server enabled"
                fi
              else
                # Rejecting incoming AUTH requests
                iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP \
                        --dport 113 -d $EXTERNAL_IP -j LnR
                if [ $VERBOSE -gt 0 ]; then
                        echo "firewall: Auth server requests will be rejected"
                fi
              fi
              #
              # TCP Services on selected ports.
              #
              #
              # Sending Mail through a remote SMTP server (25)
              #
              if [ $SMTP_REMOTE_SERVER -gt 0 ]; then
                # SMTP client to an ISP account without a local server
                for SMTP_SRVR in ${SMTP_SERVER}; do
                  iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP \
                --sport $UNPRIVPORTS --dport 25 -s $EXTERNAL_IP -d $SMTP_SRVR -j ACCEPT
                  iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
                --sport 25 --dport $UNPRIVPORTS -s $SMTP_SRVR -d $EXTERNAL_IP -j ACCEPT
                  if [ $MASQUERADING -gt 0 ]; then
                    iptables -A FORWARD -p TCP -s $INTERNAL_NETWORK -d $SMTP_SRVR --sport $UNPRIVPORTS
--dport 25 -j ACCEPT
                    iptables -A FORWARD -m state --state ESTABLISHED,RELATED -p TCP -s $SMTP_SRVR -d
$INTERNAL_NETWORK \
                        --sport 25 --dport $UNPRIVPORTS -j ACCEPT
                  fi
                  if [ $VERBOSE -gt 0 ]; then
                        echo "firewall: Clients may access remote SMTP server: ${SMTP_SRVR}"
                  fi
                done
              fi
              #
              # Sending Mail through a local SMTP server (25)
              #
              if [ $SMTP_LOCAL_SERVER -gt 0 ]; then
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP \
              --sport $UNPRIVPORTS --dport 25 -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT
                iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
              --sport 25 --dport $UNPRIVPORTS -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT
                # Receiving Mail as a Local SMTP server (25)
                iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP \
              --sport $UNPRIVPORTS --dport 25 -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
              --sport 25 --dport $UNPRIVPORTS -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT
                if [ $VERBOSE -gt 0 ]; then
                        echo "firewall: SMTP Local server enabled"
                fi
              fi
              #
              # POP3 (110) - Retrieving Mail as a POP3 client
              #
              if [ $POP3_CLIENT -gt 0 ]; then
                for POP_SRVR in ${POP_SERVER}; do
                  iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP \
                --sport $UNPRIVPORTS --dport 110 -s $EXTERNAL_IP -d $POP_SRVR -j ACCEPT
                  iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
                --sport 110 --dport $UNPRIVPORTS -s $POP_SRVR -d $EXTERNAL_IP -j ACCEPT
                  if [ $MASQUERADING -gt 0 ]; then
```

```
                    iptables -A FORWARD -p TCP -s $INTERNAL_NETWORK -d $POP_SRVR --sport $UNPRIVPORTS -
-dport 110 -j ACCEPT
                    iptables -A FORWARD -m state --state ESTABLISHED,RELATED -p TCP -s $POP_SRVR -d
$INTERNAL_NETWORK \
                        --sport 110 --dport $UNPRIVPORTS -j ACCEPT
                  fi
                  if [ $VERBOSE -gt 0 ]; then
                        echo "firewall: Clients may access remote POP-3 server: ${POP_SRVR}"
                  fi
                done
              fi
              #
              # POP3 (110) - Hosting a POP3 server for remote clients
              #
              if [ $POP3_SERVER -gt 0 ]; then
                for MY_POP3_CLIENT in ${MY_POP3_CLIENTS}; do
                  iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP \
                --sport $UNPRIVPORTS --dport 110 -s $MY_POP3_CLIENT -d $EXTERNAL_IP -j ACCEPT
                  iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
                --sport 110 --dport $UNPRIVPORTS -s $EXTERNAL_IP -d $MY_POP3_CLIENT -j ACCEPT
                  if [ $VERBOSE -gt 0 ]; then
                    echo "firewall: Remote site ${MY_POP3_CLIENT} may access local POP-3 server"
                  fi
                done
              fi
              #
              # IMAP (143) - Retrieving Mail as an IMAP client
              #
              if [ $IMAP_CLIENT -gt 0 ]; then
                for IMAP_SRVR in ${MY_IMAP_SERVER}; do
                  iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP \
                --sport $UNPRIVPORTS --dport 143 -s $EXTERNAL_IP -d $IMAP_SRVR -j ACCEPT
                  iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
                --sport 143 --dport $UNPRIVPORTS -s $IMAP_SRVR -d $EXTERNAL_IP -j ACCEPT
                  if [ $MASQUERADING -gt 0 ]; then
                    iptables -A FORWARD -p TCP -s $INTERNAL_NETWORK -d $IMAP_SRVR --sport $UNPRIVPORTS
--dport 143 -j ACCEPT
                    iptables -A FORWARD -m state --state ESTABLISHED,RELATED -p TCP -s $IMAP_SRVR -d
$INTERNAL_NETWORK \
                        --sport 143 --dport $UNPRIVPORTS -j ACCEPT
                  fi
                  if [ $VERBOSE -gt 0 ]; then
                      echo "firewall: Clients may access remote IMAP server: ${IMAP_SRVR}"
                  fi
                done
              fi
              #
              # IMAP (143) - Hosting an IMAP server for remote clients
              #
              if [ $IMAP_SERVER -gt 0 ]; then
                for MY_IMAP_CLIENT in ${MY_IMAP_CLIENTS}; do
                  iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP \
                --sport $UNPRIVPORTS --dport 143 -s $MY_IMAP_CLIENT -d $EXTERNAL_IP -j ACCEPT
                  iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
                --sport 143 --dport $UNPRIVPORTS -s $EXTERNAL_IP -d $MY_IMAP_CLIENTS -j ACCEPT
                  if [ $VERBOSE -gt 0 ]; then
                      echo "firewall: Remote site ${MY_IMAP_CLIENT} may access local IMAP server"
                  fi
                done
              fi
              #
              # IMAPS (993) - Retrieving Mail as an Secure IMAP client
              #
              if [ $IMAPS_CLIENT -gt 0 ]; then
                for IMAPS_SRVR in ${MY_IMAPS_SERVER}; do
                  iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP \
                --sport $UNPRIVPORTS --dport 993 -s $EXTERNAL_IP -d $IMAPS_SRVR -j ACCEPT
                  iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
                --sport 993 --dport $UNPRIVPORTS -s $IMAPS_SRVR -d $EXTERNAL_IP -j ACCEPT
                  if [ $MASQUERADING -gt 0 ]; then
                    iptables -A FORWARD -p TCP -s $INTERNAL_NETWORK -d $IMAP_SRVR --sport $UNPRIVPORTS
--dport 993 -j ACCEPT
                    iptables -A FORWARD -m state --state ESTABLISHED,RELATED -p TCP -s $IMAPS_SRVR -d
$INTERNAL_NETWORK \
                        --sport 993 --dport $UNPRIVPORTS -j ACCEPT
                  fi
```

```
                    if [ $VERBOSE -gt 0 ]; then
                            echo "firewall: Clients may access remote Secure IMAP server: ${IMAPS_SRVR}"
                    fi
                  done
                fi
                #
                # IMAPS (993) - Hosting a Secure IMAP server for remote clients
                #
                if [ $IMAPS_SERVER -gt 0 ]; then
                  for MY_IMAPS_CLIENT in ${MY_IMAPS_CLIENTS}; do
                   iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP \
                --sport $UNPRIVPORTS --dport 993 -s $MY_IMAP_CLIENT -d $EXTERNAL_IP -j ACCEPT
                   iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
                --sport 993 --dport $UNPRIVPORTS -s $EXTERNAL_IP -d $MY_IMAP_CLIENT -j ACCEPT
                    if [ $VERBOSE -gt 0 ]; then
                            echo "firewall: Remote site ${MY_IMAPS_CLIENT} may access local Secure IMAP server"
                    fi
                  done
                fi
                #
                # NNTP (119) - Reading and posting news as a Usenet client
                #
                if [ $NNTP_CLIENT -gt 0 ]; then
                  for NEWS_SRVR in ${NEWS_SERVER}; do
                    iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP \
                  --sport $UNPRIVPORTS --dport 119 -s $EXTERNAL_IP -d $NEWS_SRVR -j ACCEPT
                    iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
                  --sport 119 --dport $UNPRIVPORTS -s $NEWS_SRVR -d $EXTERNAL_IP -j ACCEPT
                    if [ $MASQUERADING -gt 0 ]; then
                      iptables -A FORWARD -p TCP -s $INTERNAL_NETWORK -d $NEWS_SRVR --sport
$UNPRIVPORTS --dport 119 -j ACCEPT
                      iptables -A FORWARD -m state --state ESTABLISHED,RELATED -p TCP -s $NEWS_SRVR -d
$INTERNAL_NETWORK \
                              --sport 119 --dport $UNPRIVPORTS -j ACCEPT
                    fi
                    if [ $VERBOSE -gt 0 ]; then
                            echo "firewall: Clients may access remote NNTP server: ${NEWS_SRVR}"
                    fi
                  done
                fi
                #
                # NNTP (119) - Hosting a Usenet news server for remote clients
                #
                if [ $NNTP_SERVER -gt 0 ]; then
                  for NNTP_CLIENT in ${MY_NNTP_CLIENTS}; do
                    iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP \
                  --sport $UNPRIVPORTS --dport 119 -s $NNTP_CLIENT -d $EXTERNAL_IP -j ACCEPT

                    iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
                  --sport 119 --dport $UNPRIVPORTS -s $EXTERNAL_IP -d $NNTP_CLIENT -j ACCEPT

                    if [ $VERBOSE -gt 0 ]; then
                            echo "firewall: Remote client ${NNTP_CLIENT} may access local NNTP server"
                    fi
                  done
                fi
                #
                # NNTP (119) - Allowing peer news feeds for a local Usenet server
                #
                if [ $NNTP_NEWS_FEED -gt 0 ]; then
                   iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP \
                --sport $UNPRIVPORTS --dport 119 -s $EXTERNAL_IP -d $MY_NEWS_FEED -j ACCEPT
                   iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
                --sport 119 --dport $UNPRIVPORTS -s $MY_NEWS_FEED -d $EXTERNAL_IP -j ACCEPT
                   if [ $VERBOSE -gt 0 ]; then
                           echo "firewall: External NNTP News feed access enabled"
                   fi
                fi
                #
        # Secure NNTP (563) - Reading and posting news as a Usenet client over SSL
        # Submitted by Renaud Colinet
                #
        if [ $NNTPS_CLIENT -gt 0 ]; then
                  for SNEWS_SRVR in ${SNEWS_SERVER}; do
            iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP \
          --sport $UNPRIVPORTS --dport 563 -s $EXTERNAL_IP -d $SNEWS_SRVR -j ACCEPT
```

```
        iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
      --sport 563 --dport $UNPRIVPORTS -s $SNEWS_SERVER -d $EXTERNAL_IP -j ACCEPT

        if [ $MASQUERADING -gt 0 ]; then
          iptables -A FORWARD -p TCP -s $INTERNAL_NETWORK -d $SNEWS_SRVR --sport $UNPRIVPORTS --dport
563 -j ACCEPT
          iptables -A FORWARD -m state --state ESTABLISHED,RELATED -p TCP -s $SNEWS_SRVR -d
$INTERNAL_NETWORK \
           --sport 563 --dport $UNPRIVPORTS -j ACCEPT
        fi
        if [ $VERBOSE -gt 0 ]; then
          echo "firewall: Clients may access remote secure NNTP server: ${SNEWS_SRVR}"
        fi
              done
    fi
            #
            # TELNET (23) - Allowing outgoing client access to remote sites
            #
            if [ $TELNET_CLIENT -gt 0 ]; then
              iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP \
            --sport $UNPRIVPORTS --dport 23 -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT
              iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
            --sport 23 --dport $UNPRIVPORTS -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT
              if [ $MASQUERADING -gt 0 ]; then
                iptables -A FORWARD -p TCP -s $INTERNAL_NETWORK --sport $UNPRIVPORTS --dport 23 -j
ACCEPT
                iptables -A FORWARD -m state --state ESTABLISHED,RELATED -p TCP -d $INTERNAL_NETWORK \
                        --sport 23 --dport $UNPRIVPORTS -j ACCEPT
              fi
              if [ $VERBOSE -gt 0 ]; then
                        echo "firewall: Clients may access remote TELNET servers"
              fi
            fi
            #
            # TELNET (23) - Allowing incoming access to your local server
            # Note:  Not recommended! Suggest SSH instead!
            #
            if [ $TELNET_SERVER -gt 0 ]; then
              for MY_TELNET_CLIENT in ${MY_TELNET_CLIENTS}; do
                iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP \
              --sport $UNPRIVPORTS --dport 23 -s $MY_TELNET_CLIENTS -d $EXTERNAL_IP -j ACCEPT
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
              --sport 23 --dport $UNPRIVPORTS -s $EXTERNAL_IP -d $MY_TELNET_CLIENTS -j ACCEPT
                 if [ $VERBOSE -gt 0 ]; then
                          echo "firewall: Remote site ${MY_TELNET_CLIENT} may access local TELNET server"
                 fi
              done
            fi
            #
            # SSH Client (22) - Allowing client access to remote SSH servers
            #
            if [ $SSH_CLIENT -gt 0 ]; then
              iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP \
            --sport $UNPRIVPORTS --dport 22 -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT
              iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
            --sport 22 --dport $UNPRIVPORTS -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT
              iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP \
            --sport $SSH_HI_PORTS --dport 22 -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT
              iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
            --sport 22 --dport $SSH_HI_PORTS -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT
              if [ $MASQUERADING -gt 0 ]; then
                iptables -A FORWARD -p TCP -s $INTERNAL_NETWORK --sport $UNPRIVPORTS --dport 22 -j
ACCEPT
                iptables -A FORWARD -m state --state ESTABLISHED,RELATED -p TCP -d $INTERNAL_NETWORK \
            --sport 22 --dport $UNPRIVPORTS -j ACCEPT
                iptables -A FORWARD -p TCP -s $INTERNAL_NETWORK --sport $SSH_HI_PORTS --dport 22 -j
ACCEPT
                iptables -A FORWARD -m state --state ESTABLISHED,RELATED -p TCP -d $INTERNAL_NETWORK \
                        --sport 22 --dport $SSH_HI_PORTS -j ACCEPT
              fi
              if [ $VERBOSE -gt 0 ]; then
                        echo "firewall: Clients may access remote SSH servers"
              fi
            fi
            #
```

```
                # SSH (see config) - Allowing remote client access to your local SSH server
                #
                if [ $SSH_SERVER -gt 0 ]; then
                  for MY_SSH_CLIENT in ${MY_SSH_CLIENTS}; do
                     iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP \
                    --sport $UNPRIVPORTS --dport $SSH_PORT -s $MY_SSH_CLIENT -d $EXTERNAL_IP -j ACCEPT

                     iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
                    --sport $SSH_PORT --dport $UNPRIVPORTS -s $EXTERNAL_IP -d $MY_SSH_CLIENT -j ACCEPT

                     iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP \
                    --sport $SSH_HI_PORTS --dport $SSH_PORT -s $MY_SSH_CLIENT -d $EXTERNAL_IP -j ACCEPT

                     iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
                     --sport $SSH_PORT --dport $SSH_HI_PORTS -s $EXTERNAL_IP -d $MY_SSH_CLIENT -j ACCEPT

                    if [ $VERBOSE -gt 0 ]; then
                                echo "firewall: Remote site ${MY_SSH_CLIENT} may access local SSH server"
                    fi
                done
                fi
                #
                # FTP (20, 21) - Allowing outgoing client access to remote FTP servers
                #
                if [ $FTP_CLIENT -gt 0 ]; then
                   # Outgoing request
                   iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
                --sport 21 --dport $UNPRIVPORTS -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT
                   iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP -m state --state NEW,ESTABLISHED \
                --sport $UNPRIVPORTS --dport 21 -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT
                   # Normal Port mode FTP data channels
                   iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state NEW \
                --sport 20 --dport $UNPRIVPORTS -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT
                   iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
                --sport $UNPRIVPORTS --dport 20 -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT
                   # Passive mode FTP data channels
                   iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
                --sport $UNPRIVPORTS --dport $UNPRIVPORTS -s $ANYWHERE -d $EXTERNAL_IP -j ACCEP
                   iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP -m state --state NEW,ESTABLISHED \
                --sport $UNPRIVPORTS --dport $UNPRIVPORTS -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT
                   if [ $MASQUERADING -gt 0 ]; then
                      iptables -A FORWARD -p TCP -s $INTERNAL_NETWORK --sport $UNPRIVPORTS --dport 20:21 -j
ACCEPT
                      iptables -A FORWARD -p TCP -d $INTERNAL_NETWORK --sport 20:21 --dport $UNPRIVPORTS -j
ACCEPT
                      iptables -A FORWARD -p TCP -s $INTERNAL_NETWORK --sport $UNPRIVPORTS --dport
$UNPRIVPORTS -j ACCEPT
                      iptables -A FORWARD -p TCP -d $INTERNAL_NETWORK --sport $UNPRIVPORTS --dport
$UNPRIVPORTS -j ACCEPT
                   fi
                   if [ $VERBOSE -gt 0 ]; then
                                echo "firewall: Clients may access remote FTP servers"
                   fi
                fi
                #
                # FTP (20, 21) - Allowing incoming access to your local FTP server
                #
    if [ $FTP_SERVER -gt 0 ]; then
      for MY_FTP_CLIENT in ${MY_FTP_CLIENTS}; do
        # Incoming request
          iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state NEW,ESTABLISHED \
       --sport $UNPRIVPORTS --dport 21 -s $MY_FTP_CLIENT -d $EXTERNAL_IP -j ACCEPT
         iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
       --sport 21 --dport $UNPRIVPORTS -s $EXTERNAL_IP -d $MY_FTP_CLIENT -j ACCEPT
        # Normal Port mode FTP data channel responses
          iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP \
       --sport 20 --dport $UNPRIVPORTS -s $EXTERNAL_IP -d $MY_FTP_CLIENT -j ACCEPT
          iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
       --sport $UNPRIVPORTS --dport 20 -s $MY_FTP_CLIENT -d $EXTERNAL_IP -j ACCEPT
        # Passive mode FTP data channel responses
          iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state NEW,ESTABLISHED \
       --sport $UNPRIVPORTS --dport $UNPRIVPORTS -s $MY_FTP_CLIENT -d $EXTERNAL_IP -j ACCEPT
          iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
       --sport $UNPRIVPORTS --dport $UNPRIVPORTS -s $EXTERNAL_IP -d $MY_FTP_CLIENT -j ACCEPT
         if [ $VERBOSE -gt 0 ]; then
           echo "firewall: Remote site ${MY_FTP_CLIENT} may access local FTP server"
```

```
        fi
            done
          fi
          #
          # HTTP (80) - Accessing remote web sites as a client
          #
          if [ $HTTP_CLIENT -gt 0 ]; then
             iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP \
          --sport $UNPRIVPORTS --dport 80 -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT
             iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
          --sport 80 --dport $UNPRIVPORTS -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT
             if [ $MASQUERADING -gt 0 ]; then
                iptables -A FORWARD -p TCP -s $INTERNAL_NETWORK --sport $UNPRIVPORTS --dport 80 -j
ACCEPT
                iptables -A FORWARD -p TCP -d $INTERNAL_NETWORK --sport 80 --dport $UNPRIVPORTS -j
ACCEPT
             fi
             if [ $VERBOSE -gt 0 ]; then
                        echo "firewall: Clients may access remote HTTP servers"
             fi
          fi
          #
          # HTTP (80) - Allowing remote access to a local web server
          #
          if [ $HTTP_SERVER -gt 0 ]; then
            for HTTP_CLIENT in ${MY_HTTP_CLIENTS}; do
              iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP \
            --sport $UNPRIVPORTS --dport 80 -s $HTTP_CLIENT -d $EXTERNAL_IP -j ACCEPT
              iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
            --sport 80 --dport $UNPRIVPORTS -s $EXTERNAL_IP -d $HTTP_CLIENT -j ACCEPT
              iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP \
            --sport $UNPRIVPORTS --dport 8080 -s $HTTP_CLIENT -d $EXTERNAL_IP -j ACCEPT
              iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
            --sport 8080 --dport $UNPRIVPORTS -s $EXTERNAL_IP -d $HTTP_CLIENT -j ACCEPT

              if [ $VERBOSE -gt 0 ]; then
                         echo "firewall: Remote client ${HTTP_CLIENT} may access local HTTP server"
              fi
            done
          fi
          #
          # HTTPS (443) - Accessing remote web sites over SSL as a client
          #
          if [ $HTTPS_CLIENT -gt 0 ]; then
             iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP \
          --sport $UNPRIVPORTS --dport 443 -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT

             iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
          --sport 443 --dport $UNPRIVPORTS -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT

             if [ $MASQUERADING -gt 0 ]; then
                iptables -A FORWARD -p TCP -s $INTERNAL_NETWORK --sport $UNPRIVPORTS --dport 443 -j
ACCEPT
                iptables -A FORWARD -m state --state ESTABLISHED,RELATED -p TCP -d $INTERNAL_NETWORK \
                        --sport 443 --dport $UNPRIVPORTS -j ACCEPT
             fi
             if [ $VERBOSE -gt 0 ]; then
                        echo "firewall: Clients may access remote HTTPS servers"
             fi
          fi
          #
          # HTTPS (443) - Allowing remote access to a local SSL web server
          #
          if [ $HTTPS_SERVER -gt 0 ]; then
            for HTTPS_CLIENT in ${MY_HTTPS_CLIENTS}; do
              iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP \
            --sport $UNPRIVPORTS --dport 443 -s $HTTPS_CLIENT -d $EXTERNAL_IP -j ACCEPT
              iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
            --sport 443 --dport $UNPRIVPORTS -s $EXTERNAL_IP -d $HTTPS_CLIENT -j ACCEPT
              if [ $VERBOSE -gt 0 ]; then
                         echo "firewall: Remote client ${HTTPS_CLIENT} may access local HTTPS server"
              fi
            done
          fi
          #
          # HTTP Proxy Client (8008/8080)
```

```
#
if [ $HTTP_PROXY -gt 0 ]; then
   iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP \
--sport $UNPRIVPORTS --dport $WEB_PROXY_PORT -s $EXTERNAL_IP -d $WEB_PROXY_SERVER -j
ACCEPT
   iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
--sport $WEB_PROXY_PORT --dport $UNPRIVPORTS -s $WEB_PROXY_SERVER -d $EXTERNAL_IP -j
ACCEPT
   if [ $MASQUERADING -gt 0 ]; then
     iptables -A FORWARD -p TCP -s $INTERNAL_NETWORK --sport $UNPRIVPORTS --dport
$WEB_PROXY_PORT -j ACCEPT
     iptables -A FORWARD -m state --state ESTABLISHED,RELATED -p TCP -d $INTERNAL_NETWORK \
          --sport $WEB_PROXY_PORT --dport $UNPRIVPORTS -j ACCEPT
   fi
   if [ $VERBOSE -gt 0 ]; then
          echo "firewall: Clients may access remote sites via HTTP Proxy Server"
   fi
fi
#
# FINGER (79) - Accessing remote finger servers as a client
#
if [ $FINGER_CLIENT -gt 0 ]; then
   iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP \
--sport $UNPRIVPORTS --dport 79 -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT
   iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
--sport 79 --dport $UNPRIVPORTS -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT
   if [ $MASQUERADING -gt 0 ]; then
     iptables -A FORWARD -p TCP -s $INTERNAL_NETWORK --sport $UNPRIVPORTS --dport 79 -j
ACCEPT
     iptables -A FORWARD -m state --state ESTABLISHED,RELATED -p TCP -d $INTERNAL_NETWORK \
          --sport 79 --dport $UNPRIVPORTS -j ACCEPT
   fi
   if [ $VERBOSE -gt 0 ]; then
          echo "firewall: Clients may access remote FINGER servers"
   fi
fi
#
# FINGER (79) - Allowing remote client access to a local finger server (dangerous!)
#
if [ $FINGER_SERVER -gt 0 ]; then
  for FINGER_CLIENT in $MY_FINGER_CLIENTS}; do
    iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP \
  --sport $UNPRIVPORTS --dport 79 -s $FINGER_CLIENT -d $EXTERNAL_IP -j ACCEPT
    iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
  --sport 79 --dport $UNPRIVPORTS -s $EXTERNAL_IP -d $FINGER_CLIENT -j ACCEPT
    if [ $VERBOSE -gt 0 ]; then
          echo "firewall: Remote client ${FINGER_CLIENT} may access local FINGER server"
    fi
  done
fi
#
# WHOIS (43) - Accessing a remote WHOIS server as a client
#
if [ $WHOIS_CLIENT -gt 0 ]; then
   iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP \
--sport $UNPRIVPORTS --dport 43 -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT
   iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
--sport 43 --dport $UNPRIVPORTS -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT
   if [ $MASQUERADING -gt 0 ]; then
     iptables -A FORWARD -p TCP -s $INTERNAL_NETWORK --sport $UNPRIVPORTS --dport 43 -j
ACCEPT
     iptables -A FORWARD -m state --state ESTABLISHED,RELATED -p TCP -d $INTERNAL_NETWORK \
          --sport 43 --dport $UNPRIVPORTS -j ACCEPT
   fi
   if [ $VERBOSE -gt 0 ]; then
          echo "firewall: Clients may access remote WHOIS servers"
   fi
fi
#
# GOPHER (70) - Accessing a remote GOPHER server as a client
#
if [ $GOPHER_CLIENT -gt 0 ]; then
   iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP \
--sport $UNPRIVPORTS --dport 70 -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT

   iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
```

```
            --sport 70 --dport $UNPRIVPORTS -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT

              if [ $MASQUERADING -gt 0 ]; then
                iptables -A FORWARD -p TCP -s $INTERNAL_NETWORK --sport $UNPRIVPORTS --dport 70 -j
ACCEPT
                iptables -A FORWARD -m state --state ESTABLISHED,RELATED -p TCP -d $INTERNAL_NETWORK \
                        --sport 70 --dport $UNPRIVPORTS -j ACCEPT
              fi
              if [ $VERBOSE -gt 0 ]; then
                        echo "firewall: Clients may access remote GOPHER servers"
              fi
            fi
            #
            # WAIS (210) - Accessing a remote WAIS server as a client
            #
            if [ $WAIS_CLIENT -gt 0 ]; then
              iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP \
            --sport $UNPRIVPORTS --dport 210 -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT
              iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
            --sport 210 --dport $UNPRIVPORTS -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT
              if [ $MASQUERADING -gt 0 ]; then
                iptables -A FORWARD -p TCP -s $INTERNAL_NETWORK --sport $UNPRIVPORTS --dport 210 -j
ACCEPT
                iptables -A FORWARD -m state --state ESTABLISHED,RELATED -p TCP -d $INTERNAL_NETWORK \
                        --sport 210 --dport $UNPRIVPORTS -j ACCEPT
              fi
              if [ $VERBOSE -gt 0 ]; then
                        echo "firewall: Clients may access remote WAIS servers"
              fi
            fi
            #
    # Real Video (554) - Real Video Client
    #
    if [ $RV_CLIENT -gt 0 ]; then
      iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP \
        --sport $UNPRIVPORTS --dport 554 -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT
      iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP \
        --sport 554 -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT
              if [ $MASQUERADING -gt 0 ]; then
                iptables -A FORWARD -p TCP -s $INTERNAL_NETWORK --sport $UNPRIVPORTS --dport 554 -j
ACCEPT
                iptables -A FORWARD -m state --state ESTABLISHED,RELATED -p TCP -d $INTERNAL_NETWORK \
                        --sport 554 --dport $UNPRIVPORTS -j ACCEPT
              fi
      if [ $VERBOSE -gt 0 ]; then
        echo "firewall: Real Video client enabled"
      fi
    fi
    #
    # PPTP (1723) - Accessing PPTP servers as a client
    #
    if [ $PPTP_CLIENT -gt 0 ]; then
      iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP \
        --sport $UNPRIVPORTS --dport 1723 \
        -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT
      iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP  \
        --sport 1723 --dport $UNPRIVPORTS \
        -s $ANYWHERE -d $EXTERNAL_IP \
        -m state --state ESTABLISHED,RELATED -j ACCEPT
      iptables -A INPUT -i $EXTERNAL_INTERFACE -p 47 -j ACCEPT
      iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p 47 -j ACCEPT
      if [ $MASQUERADING -gt 0 ]; then
        iptables -A INPUT -i $INTERNAL_INTERFACE -p 47 -j ACCEPT
        iptables -A OUTPUT -o $INTERNAL_INTERFACE -p 47 -j ACCEPT
        iptables -A FORWARD -p TCP -s $INTERNAL_NETWORK \
          --sport $UNPRIVPORTS --dport 1723 -j ACCEPT
        iptables -A FORWARD -p TCP -d $INTERNAL_NETWORK \
          -m state --state ESTABLISHED,RELATED \
          --sport 1723 --dport $UNPRIVPORTS -j ACCEPT
        iptables -A FORWARD -p 47 -s $INTERNAL_NETWORK -j ACCEPT
        iptables -A FORWARD -p 47 -d $INTERNAL_NETWORK -j ACCEPT
      fi
      if [ $VERBOSE -gt 0 ]; then
        echo "firewall: Clients may access remote PPTP servers"
      fi
    fi
```

```
                #
                # UDP - Accept only on selected ports
                #
                #
                # TRACEROUTE
                #
                # Traceroute usually uses -s 32769:65535 -d 33434:33523
                #
                if [ $OUTBOUND_TRACEROUTE -gt 0 ]; then
                   # Enable outgoing TRACEROUTE requests
                   iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p UDP \
                --sport $TRACEROUTE_SRC_PORTS --dport $TRACEROUTE_DEST_PORTS \
                -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT

                   if [ $MASQUERADING -gt 0 ]; then
                     iptables -A FORWARD -p UDP -s $INTERNAL_NETWORK --sport $TRACEROUTE_SRC_PORTS \
                              --dport $TRACEROUTE_DEST_PORTS -j ACCEPT
                     iptables -A FORWARD -p UDP -d $INTERNAL_NETWORK --sport $TRACEROUTE_DEST_PORTS \
                              --dport $TRACEROUTE_SRC_PORTS -j ACCEPT
                   fi
                   if [ $VERBOSE -gt 0 ]; then
                              echo "firewall: Outbound TRACEROUTE enabled"
                   fi
                fi
                if [ $INBOUND_TRACEROUTE -gt 0 ]; then
                   # Enable incoming TRACEROUTE query
                   iptables -A INPUT -i $EXTERNAL_INTERFACE -p UDP \
                --sport $TRACEROUTE_SRC_PORTS --dport $TRACEROUTE_DEST_PORTS \
                -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT

                   if [ $MASQUERADING -gt 0 ]; then
                     iptables -A FORWARD -p UDP -d $INTERNAL_NETWORK --sport $TRACEROUTE_SRC_PORTS \
                              --dport $TRACEROUTE_DEST_PORTS -j ACCEPT
                   fi
                   if [ $VERBOSE -gt 0 ]; then
                              echo "firewall: Inbound TRACEROUTE enabled"
                   fi
                fi
        #
        # DHCP Server
                #
        # This assumes that you're running a DHCP server on your firewall to
        # supply IP addresses to your internal network using dhcpd.  See any
        # of several DHCP HowTo sites for the actual server setup.
        #
        if [ $DHCP_SERVER -gt 0 ]; then
           iptables -A INPUT -i $INTERNAL_INTERFACE -p udp -s $BROADCAST_SRC \
        -d $BROADCAST_DEST --sport 67:68 --dport 67:68 -j ACCEPT
           iptables -A OUTPUT -o $INTERNAL_INTERFACE -p udp -s $INTERNAL_IP \
        --sport 67:68 --dport 67:68 -j ACCEPT
           iptables -A FORWARD -p udp -s $INTERNAL_NETWORK --sport 67:68 --dport 67:68 -j ACCEPT
                   iptables -A FORWARD -p udp -d $INTERNAL_NETWORK --sport 67:68 --dport 67:68 -j ACCEPT

          if [ $VERBOSE -gt 0 ]; then
              echo "firewall: DHCP Server enabled"
          fi
        fi
        #
        # NTP (123) - Accessing remote Network Time Servers
        #
        if [ $NTP_CLIENT -gt 0 ]; then
           iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p UDP \
        --sport $UNPRIVPORTS --dport 123 -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT
           iptables -A INPUT -i $EXTERNAL_INTERFACE -p UDP \
        --sport 123 --dport $UNPRIVPORTS -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT
           iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p UDP \
        --sport 123 --dport 123 -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT
           iptables -A INPUT -i $EXTERNAL_INTERFACE -p UDP \
        --sport 123 --dport 123 -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT

                   if [ $MASQUERADING -gt 0 ]; then
                     iptables -A FORWARD -p UDP -s $INTERNAL_NETWORK --sport $UNPRIVPORTS --dport 123 -j
ACCEPT
                     iptables -A FORWARD -p UDP -d $INTERNAL_NETWORK --sport 123 --dport $UNPRIVPORTS -j
ACCEPT
                     iptables -A FORWARD -p UDP -s $INTERNAL_NETWORK --sport 123 --dport 123 -j ACCEPT
```

```
                iptables -A FORWARD -p UDP -d $INTERNAL_NETWORK --sport 123 --dport 123 -j ACCEPT
              fi
        if [ $VERBOSE -gt 0 ]; then
          echo "firewall: NTP Client enabled"
        fi
      fi
              #
              # ICQ (4000) - The Miribilis ICQ Client
              #
              if [ $ICQ_CLIENT -gt 0 ]; then
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p UDP \
              --sport $UNPRIVPORTS --dport 4000 -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT
                iptables -A INPUT -i $EXTERNAL_INTERFACE -p UDP \
              --sport 4000 --dport $UNPRIVPORTS -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT
                if [ $MASQUERADING -gt 0 ]; then
                  iptables -A FORWARD -p UDP -s $INTERNAL_NETWORK --sport $UNPRIVPORTS --dport 4000 -j
ACCEPT
                  iptables -A FORWARD -p UDP -d $INTERNAL_NETWORK --sport 4000 --dport $UNPRIVPORTS -j
ACCEPT
                fi
                if [ $VERBOSE -gt 0 ]; then
                  echo "firewall: ICQ Client enabled"
                fi
              fi
              #
              # GAMES
              # Half-Life/CounterStrike
              #
              if [ $HALF_LIFE -gt 0 ]; then
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p UDP \
              --sport 27000:27050 --dport $UNPRIVPORTS -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT
                iptables -A INPUT -i $EXTERNAL_INTERFACE -p UDP \
              -m state --state RELATED,ESTABLISHED,NEW --dport 27000:27050 -s $ANYWHERE \
              -d $EXTERNAL_IP -j ACCEPT
                if [ $MASQUERADING -gt 0 ]; then
                  iptables -A FORWARD -p UDP -s $INTERNAL_NETWORK --sport 27000:27050 --dport
$UNPRIVPORTS -j ACCEPT
                  iptables -A FORWARD -p UDP -d $INTERNAL_NETWORK -m state --state
RELATED,ESTABLISHED,NEW --dport 27000:27050 -j ACCEPT
                fi
                if [ $VERBOSE -gt 0 ]; then
                  echo "firewall: Half-Life/CounterStrike game ports enabled"
                fi
              fi
              #
              # Return to Castle Wolfenstein
              #
              if [ $WOLF_CLIENT -gt 0 ]; then
                iptables  -A OUTPUT -o $EXTERNAL_INTERFACE -p UDP \
              --sport $UNPRIVPORTS --dport 27950:27965 -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT
                iptables  -A INPUT -i $EXTERNAL_INTERFACE -p UDP \
              --sport 27950:27965 --dport $UNPRIVPORTS -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT
                if [ $MASQUERADING -gt 0 ]; then
                  iptables -A FORWARD -p UDP -s $INTERNAL_NETWORK --sport $UNPRIVPORTS --dport
27950:27965 -j ACCEPT
                  iptables -A FORWARD -p UDP -d $INTERNAL_NETWORK --sport 27950:27965 --dport
$UNPRIVPORTS -j ACCEPT
                fi
                if [ $VERBOSE -gt 0 ]; then
                  echo "firewall: Castle Wolfenstein game ports enabled"
                fi
              fi
              # --------------------------------------------------------------
              #
              # Spoofing and Bad Addresses
              #
              # Refuse spoofed packets.
              # Ignore blatantly illegal source addresses.
              # Protect yourself from sending to bad addresses.
              # Refuse spoofed packets pretending to be from
              # the external interface's IP address.
              iptables -A INPUT -i $EXTERNAL_INTERFACE -s $EXTERNAL_IP -j LnD
              # Refuse packets claiming to be to or from a Class-A private network.
              iptables -A INPUT -i $EXTERNAL_INTERFACE -s $CLASS_A -j LnD
              iptables -A INPUT -i $EXTERNAL_INTERFACE -d $CLASS_A -j LnD
              iptables -A OUTPUT -o $EXTERNAL_INTERFACE -s $CLASS_A -j LnD
```

```
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -d $CLASS_A -j LnD
                # Refuse packets claiming to be to or from a Class-B private network.
                iptables -A INPUT -i $EXTERNAL_INTERFACE -s $CLASS_B -j LnD
                iptables -A INPUT -i $EXTERNAL_INTERFACE -d $CLASS_B -j LnD
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -s $CLASS_B -j LnD
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -d $CLASS_B -j LnD
                # Refuse packets claiming to be to or from a Class-C private network.
                iptables -A INPUT -i $EXTERNAL_INTERFACE -s $CLASS_C -j LnD
                iptables -A INPUT -i $EXTERNAL_INTERFACE -d $CLASS_C -j LnD
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -s $CLASS_C -j LnD
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -d $CLASS_C -j LnD
                # Refuse packets claiming to be from the loopback.
                iptables -A INPUT -i $EXTERNAL_INTERFACE -s $LOOPBACK_NETWORK -j LnD
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -s $LOOPBACK_NETWORK -j LnD
                # Refuse malformed broadcast packets.
                iptables -A INPUT -i $EXTERNAL_INTERFACE -s $BROADCAST_DEST -j LnD
                iptables -A INPUT -i $EXTERNAL_INTERFACE -d $BROADCAST_SRC  -j LnD
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -s $BROADCAST_DEST -j LnD
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -d $BROADCAST_SRC  -j LnD
                # Refuse Class-D Multicast addresses.
                # Multicast is only illegal as a source address.
                # Multicast uses UDP.
                iptables -A INPUT -i $EXTERNAL_INTERFACE -s $CLASS_D_MULTICAST -j LnD
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -s $CLASS_D_MULTICAST -j LnR
                # Refuse Class-E reserved IP addresses.
                iptables -A INPUT -i $EXTERNAL_INTERFACE -s $CLASS_E_RESERVED_NET -j LnD
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -d $CLASS_E_RESERVED_NET -j LnR
                # ----------------------------------------------------------------
                #
                # DROP (on input), REJECT (output) and LOG anything else on the external (red) interface
                #
                iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP \
                   -s $ANYWHERE -j LnD
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP \
                   -s $ANYWHERE -j LnR
                iptables -A INPUT -i $EXTERNAL_INTERFACE -p UDP \
                   -s $ANYWHERE -j LnD
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p UDP \
                   -s $ANYWHERE -j LnR
                iptables -A INPUT -i $EXTERNAL_INTERFACE -p ICMP \
                   -s $ANYWHERE -j LnD
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p ICMP \
                   -s $ANYWHERE -j LnR
                #
                # Masquerade internal traffic
                #
                if [ $MASQUERADING -gt 0 ]; then
                   # All internal traffic is masqueraded externally
                   iptables -t nat -A POSTROUTING -s $INTERNAL_NETWORK -o $EXTERNAL_INTERFACE -j SNAT \
                   --to $EXTERNAL_IP
        # Note: some may find this works better on machines with non-static
                   # external IP addresses:
        # iptables -t nat -A POSTROUTING -o ethX -j MASQUERADE
                   # Enable IP Forwarding
                   echo 1 >/proc/sys/net/ipv4/ip_forward
                   #
                   # Unlimited traffic within the local network
                   #
                   # All internal machines have access to the firewall machine
                   iptables -A INPUT -i $INTERNAL_INTERFACE -s $INTERNAL_NETWORK -j ACCEPT
                   iptables -A OUTPUT -o $INTERNAL_INTERFACE -d $INTERNAL_NETWORK -j ACCEPT
        if [ $VERBOSE -gt 0 ]; then
                                echo "firewall: Masquerading internal network"
                   fi
                fi
                # ----------------------------------------------------------------
                # Zero counts
                iptables -Z
                # ----------------------------------------------------------------
                echo "done"
                touch /var/lock/subsys/firewall
                echo
                ;;
status)
                if [ -f /var/lock/subsys/firewall ]; then
                   echo "Firewall started and configured"
```

```
                else
                   echo "Firewall stopped"
                fi
                exit 0
                ;;
restart|reload)
                $0 stop
                $0 start
                ;;
stop)
                echo "Shutting down Firewall services"
                # Turn off IP Forwarding
                echo 0 >/proc/sys/net/ipv4/ip_forward
                # Turn off dynamic IP hacking
                echo 0 > /proc/sys/net/ipv4/ip_dynaddr
                # Flush the rule chains
                iptables -F
                # Delete custom chains
                iptables -X
                # Zero counts
                iptables -Z
                # Set the default policy to DROP
                iptables -P INPUT DROP
                iptables -P OUTPUT DROP
                iptables -P FORWARD DROP
                # Allow unlimited traffic on the loopback interface
                iptables -A INPUT -i $LOOPBACK_INTERFACE -j ACCEPT
                iptables -A OUTPUT -o $LOOPBACK_INTERFACE -j ACCEPT
                # Open the configuration file
                if [ -f /etc/firewall/firewall.conf.iptables ]; then
                   . /etc/firewall/firewall.conf.iptables
        if [ $MASQUERADING -gt 0 ]; then
           # Allow unlimited local traffic on the internal interface
           iptables -A INPUT -i $INTERNAL_INTERFACE -j ACCEPT
           iptables -A OUTPUT -o $INTERNAL_INTERFACE -j ACCEPT
                   fi
                else
                   echo "firewall: No configuration file found at /etc/firewall/firewall.conf.iptables"
                   exit 1
                fi
                rm -f /var/lock/subsys/firewall
                echo
                ;;
*)
                echo "Usage: /etc/rc.d/init.d/firewall.iptables {start|stop|status|restart|reload}"
                exit 1
esac
exit 0
```

APPENDIX F2

Dynamic Firewall Script

```
#!/bin/bash

source /usr/local/share/dynfw.sh

args 2 $# "${0} IPADDR {on/off}" "Drops packets to/from IPADDR. Good for obnoxious
                        networks/hosts/DoS"

if [ "$2" == "on" ]
then
  #rules will be appended or inserted as normal
 APPEND="-A"
  INSERT="-I"
  rec_check ipdrop $1 "$1 already blocked" on
  record ipdrop $1
elif [ "$2" == "off" ]
then
  #rules will be deleted instead
 APPEND="-D"
  INSERT="-D"
  rec_check ipdrop $1 "$1 not currently blocked" off
  unrecord ipdrop $1
else
  echo "Error: \"off\" or \"on\" expected as second argument"
  exit 1
fi

#block outside IP address that's causing problems
#attacker's incoming TCP connections will take a minute or so to time out,
#reducing DoS effectiveness.

iptables $INSERT INPUT   -s $1 -j DROP
iptables $INSERT OUTPUT  -d $1 -j DROP
iptables $INSERT FORWARD -d $1 -j DROP
iptables $INSERT FORWARD -s $1 -j DROP

echo "IP ${1} drop ${2}."
```

APPENDIX F3

Calculating Parameters for No Firewall Configuration

$f_{initialization} : \forall c(n,m) = 1$

$f_{runtime} : \forall c(n,m) = 1$

F3.1 Probability of Available Services

$$\begin{aligned} P(AS-NF) &= P(c(n,m)=1 \mid f_{initialization}) \wedge P(c(n,m)=1 \mid f_{runtime}) \\ &= P(c(n,m)=1 \mid \forall c(n,m)=1) \wedge P(c(n,m)=1 \mid \forall c(n,m)=1) \\ &= 1 \wedge 1 \\ &= 1 \end{aligned}$$

F3.2 Probability of Expose Line

$P(AS-NF) = 1$

- For $t < t_r$ or $t > t_{ft}$, $P(EL-NF) = 1$
- For $th(n) > 0$, $P(EL-NF) = 1$
- For $protection(n) < risk(m)$, $P(EL-NF) = 1$

F3.3 Probability of Denial of Service

$$\begin{aligned} P(DS-NF) &= P(c(n,m)=0 \mid f_{initialization}) \vee P(c(n,m)=0 \mid f_{runtime}) \\ &= P(c(n,m)=1 \mid \forall c(n,m)=1) \vee P(c(n,m)=1 \mid \forall c(n,m)=1) \\ &= 0 \vee 0 \\ &= 0 \end{aligned}$$

APPENDIX F4

Calculating Parameters for Static Firewall Configuration

$f_{initialization} : \exists c(n,m) = 1 + \exists c(n,m) = 0$

$f_{runtime} : \exists c(n,m) = 1 + \exists c(n,m) = 0$

F4.1 Probability of Available Services

$$P(AS-SF) = P(c(n,m) = 1 \mid f_{initialization}) \wedge P(c(n,m) = 1 \mid f_{runtime})$$
$$= P(c(n,m) = 1 \mid \exists c(n,m) = 1 + \exists c(n,m) = 0) \wedge P(c(n,m) = 1 \mid \exists c(n,m) = 1 + \exists c(n,m) = 0)$$
$$= P(c(n,m) = 1 \mid \exists c(n,m) = 1 + \exists c(n,m) = 0)$$

F4.2 Probability of Expose Line

$$P(AS-SF) = P(c(n,m) = 1 \mid \exists c(n,m) = 1 + \exists c(n,m) = 0)$$

- For $t < t_r$ or $t > t_{ft}$, $P(EL-SF) = P(c(n,m) = 1 \mid \exists c(n,m) = 1 + \exists c(n,m) = 0)$
- For $th(n) > 0$, $P(EL-SF) = P(c(n,m) = 1 \mid \exists c(n,m) = 1 + \exists c(n,m) = 0)$
- For $protection(n) < risk(m)$,
  $P(EL-SF) = P(c(n,m) = 1 \mid \exists c(n,m) = 1 + \exists c(n,m) = 0)$

F4.3 Probability of Denial of Service

$$P(DS-SF) = P(c(n,m) = 0 \mid f_{initialization}) \vee P(c(n,m) = 0 \mid f_{runtime})$$
$$= P(c(n,m) = 0 \mid \exists c(n,m) = 1 + \exists c(n,m) = 0) \vee P(c(n,m) = 0 \mid \exists c(n,m) = 1 + \exists c(n,m) = 0)$$
$$= P(c(n,m) = 0 \mid \exists c(n,m) = 1 + \exists c(n,m) = 0)$$

APPENDIX F5

Calculating Parameters for Dynamic Firewall Configuration

$f_{initialization} : \exists c(n,m) = 1 + \exists c(n,m) = 0$

$f_{runtime} : \exists c(n,m) = 0 \Leftrightarrow th(n) > 0$

F5.1 Probability of Available Services

$$P(AS-DF) = P(c(n,m) = 1 \mid f_{initialization}) \wedge P(c(n,m) = 1 \mid f_{runtime})$$
$$= P(c(n,m) = 1 \mid \exists c(n,m) = 1 + \exists c(n,m) = 0) \wedge P(c(n,m) = 1 \mid \exists c(n,m) = 0 \Leftrightarrow th(n) > 0)$$
$$= P(c(n,m) = 1 \mid \exists c(n,m) = 1 + \exists c(n,m) = 0) \wedge 1$$
$$= P(c(n,m) = 1 \mid \exists c(n,m) = 1 + \exists c(n,m) = 0)$$

F5.2 Probability of Expose Line

$$P(AS-DF) = P(c(n,m) = 1 \mid \exists c(n,m) = 1 + \exists c(n,m) = 0) \wedge P(c(n,m) = 1 \mid \exists c(n,m) = 0 \Leftrightarrow th(n) > 0)$$

- For $t < t_r$ or $t > t_{ft}$, $P(EL-DF) = P(c(n,m) = 1 \mid \exists c(n,m) = 1 + \exists c(n,m) = 0)$
- For $th(n) > 0$, $P(EL-DF) = 0$
- For $protection(n) < risk(m)$,
  $P(EL-DF) = P(c(n,m) = 1 \mid \exists c(n,m) = 1 + \exists c(n,m) = 0)$

F5.3 Probability of Denial of Service

$$P(DS-DF) = P(c(n,m) = 0 \mid f_{initialization}) \vee P(c(n,m) = 0 \mid f_{runtime})$$
$$= P(c(n,m) = 0 \mid \exists c(n,m) = 1 + \exists c(n,m) = 0) \vee P(c(n,m) = 0 \mid \exists c(n,m) = 0 \Leftrightarrow th(n) > 0)$$
$$= P(c(n,m) = 0 \mid \exists c(n,m) = 1 + \exists c(n,m) = 0) \vee 0$$
$$= P(c(n,m) = 0 \mid \exists c(n,m) = 1 + \exists c(n,m) = 0)$$

APPENDIX F1

Static Firewall Script

```
#!/bin/sh
#
# firewall      Firewall startup/shutdown script
#
# Version:      @(#) /etc/rc.d/init.d/firewall.iptables  08-April-2005
#
#
# Translated to iptables format, with several additions and modifications,
# from Craig Zeller's (zeller@fatpenguin.com) ipchains-based firewall script, by
# Bob Sully (rcs@malibyte.net)
#
# Thanks to Jeff Carlson (jeff@ultimateevil.org) for his assistance re: DHCP and several other issues,
# Rohan Amin (rohan@rohanamin.com) and Erik Wasser (erik.wasser@iquer.com) for help with the port-forwarding
# routine, and Nate Waddoups for his quick PPTP hack.
#
# Latest revision: 08-Apr-2005
#
# chkconfig: 345 11 91
#
# description: IP Firewall startup/shutdown script for iptables
#
# probe: true
#
#
# CONSTANTS - Do not edit
#
ANYWHERE="0.0.0.0/0"                            # Match any IP address
BROADCAST_SRC="0.0.0.0"                                 # Broadcast Source Address
BROADCAST_DEST="255.255.255.255"                # Broadcast Destination Address
CLASS_A="10.0.0.0/8"                            # Class-A Private (RFC-1918) Networks
CLASS_B="172.16.0.0/12"                         # Class-B Private (RFC-1918) Networks
CLASS_C="192.168.0.0/16"                        # Class-C Private (RFC-1918) Networks
CLASS_D_MULTICAST="224.0.0.0/4"                 # Class-D Multicast Addresses
CLASS_E_RESERVED_NET="240.0.0.0/5"              # Class-E Reserved Addresses
PRIVPORTS="0:1023"                              # Well-Known, Privileged Port Range
UNPRIVPORTS="1024:65535"                        # Unprivileged Port Range
TRACEROUTE_SRC_PORTS="32769:65535"              # Traceroute Source Ports
TRACEROUTE_DEST_PORTS="33434:33523"             # Traceroute Destination Ports
#
# The Loopback interface defines should not be
# edited unless your Linux distribution defines
# these differently.
#
LOOPBACK_INTERFACE="lo"                                 # The loopback interface
LOOPBACK_NETWORK="127.0.0.0/8"                  # Reserved Loopback Address Range
#
# Source function library.
#
. /etc/rc.d/init.d/functions
#
# See how we were called.
#
case "$1" in
  start)
        echo "Starting Firewall services"
                echo "firewall: Configuring Firewall Rules using iptables"
                # Remove any existing rules from all chains
                iptables -F
                iptables -F -t nat
                iptables -F -t mangle
                # Set the default policy to drop
                iptables -P INPUT   DROP
                iptables -P OUTPUT  DROP
                iptables -P FORWARD DROP
                # Allow unlimited traffic on the loopback interface
                iptables -A INPUT -i $LOOPBACK_INTERFACE -j ACCEPT
                iptables -A OUTPUT -o $LOOPBACK_INTERFACE -j ACCEPT
                # A bug that showed up as of the Red Hat 7.2 release results in
                # the following 5 default policies breaking the firewall
                # initialization:
                #     fgrep -q '7.2' /etc/redhat-release
                #     if [ $? -ne 0 ] ; then
                #       iptables -t nat    -P PREROUTING  DROP
```

```
            #       iptables -t nat    -P OUTPUT      DROP
            #       iptables -t nat    -P POSTROUTING DROP
            #       iptables -t mangle -P PREROUTING  DROP
            #       iptables -t mangle -P OUTPUT      DROP
            #     fi
            # Remove any pre-existing user-defined chains
            iptables -X
            iptables -X -t nat
            iptables -X -t mangle
            # Zero counts
            iptables -Z
            # Open the configuration file
            if [ -f /etc/firewall/firewall.conf.iptables ]; then
               . /etc/firewall/firewall.conf.iptables
            else
               # Turn off IP Forwarding & Masquerading
               echo 0 >/proc/sys/net/ipv4/ip_forward
               # Turn off dynamic IP hacking
   echo "0" > /proc/sys/net/ipv4/ip_dynaddr

               if [ $MASQUERADING -gt 0 ]; then
                  # Allow unlimited local traffic on the internal interface
                  iptables -A INPUT -i $INTERNAL_INTERFACE -j ACCEPT
                  iptables -A OUTPUT -o $INTERNAL_INTERFACE -j ACCEPT
               fi
               echo "firewall: No configuration file found at /etc/firewall/firewall.conf.iptables; "
               echo "firewall: default policies set to DROP on INPUT/OUTPUT/FORWARD chains."
               exit 1
            fi
#
# If your IP address is dynamically assigned by a DHCP server,
# your DHCP server's IP address and this machine's IP address are
# obtained from /etc/dhcpc/hostinfo-$EXTERNAL_INTERFACE or
# /etc/dhcpc/dhcpcd-$EXTERNAL_INTERFACE.info.
#
if [ $DHCP -gt 0 ]; then
               # Grab external IP address if already assigned
  EXTERNAL_IP=$( ifconfig $EXTERNAL_INTERFACE | grep 'inet[^6]' | sed 's/[a-zA-Z:]//g' | awk '{print $1}' )
               if [ -n $EXTERNAL_IP ]; then
    EXT_NETMASK=$( ifconfig $EXTERNAL_INTERFACE | grep 'inet[^6]' | sed 's/[a-zA-Z:]//g' | awk '{print $3}' )
    EXTERNAL_NETWORK=$( ipcalc -n $EXTERNAL_IP $EXT_NETMASK | cut -d\= -f2 )
    BROADCAST_NET=$( ipcalc -b $EXTERNAL_IP $EXT_NETMASK | cut -d\= -f2 )
               fi
  # Turn on dynamic IP hacking
  echo "1" > /proc/sys/net/ipv4/ip_dynaddr
  # Incoming DHCPOFFER from available DHCP servers
  iptables -A INPUT -i $EXTERNAL_INTERFACE -p udp \
          -s 0.0.0.0        --sport 67 \
          -d 255.255.255.255 --dport 68 -j ACCEPT
  # Initialization of rebinding: No lease or Lease time expired.
  iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p udp   \
          -s 0.0.0.0        --sport 68 \
          -d 255.255.255.255 --dport 67 -j ACCEPT
  # Fall back to initialization
  # The client knows its server, but has either lost its
  # lease, or else needs to reconfirm the IP address after
  # rebooting.
  iptables -A INPUT -i $EXTERNAL_INTERFACE -p udp \
          -s $DHCP_SERVER_IP --sport 67     \
          -d 255.255.255.255 --dport 68 -j ACCEPT
  iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p udp \
          -s 255.255.255.255 --sport 68 \
          -d $DHCP_SERVER_IP --dport 67 -j ACCEPT
  # As a result of the above, we're supposed to change our IP
  # address with this message, which is addressed to our new
  # address before the dhcp client has received the update.
  # Depending on the server implementation, the destination
  # address can be the new IP address, the subnet address, or
  # the limited broadcast address.
  # If the network subnet address is used as the destination,
  # the next rule must allow incoming packets destined to the
  # subnet address, and the rule must preceed any general rules
  # that block such incoming broadcast packets.
  iptables -A INPUT -i $EXTERNAL_INTERFACE -p udp \
          -s $DHCP_SERVER_IP --sport 67     \
```

```
        --dport 68 -j ACCEPT
    # Lease renewal
    iptables -A INPUT -i $EXTERNAL_INTERFACE -p udp \
        -s $DHCP_SERVER_IP --sport 67    \
        -d $EXTERNAL_IP --dport 68 -j ACCEPT
    iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p udp \
        -s $EXTERNAL_IP --sport 68      \
        -d $DHCP_SERVER_IP --dport 67 -j ACCEPT
    echo "firewall: DHCP Client configured"
          else
            # External IP assigned without DHCP (i.e. static); get some more info
      EXT_NETMASK=$( ifconfig $EXTERNAL_INTERFACE | grep 'inet[^6]' | sed 's/[a-zA-Z:]//g' | awk '{print $3}' )
      EXTERNAL_NETWORK=$( ipcalc -n $EXTERNAL_IP $EXT_NETMASK | cut -d\= -f2 )
      BROADCAST_NET=$( ipcalc -b $EXTERNAL_IP $EXT_NETMASK | cut -d\= -f2 )
  fi
        #
        # Refuse directed broadcasts; you may choose not to log these, as they can fill up your logs quickly
        #
#       iptables -A INPUT -i $EXTERNAL_INTERFACE -d $EXTERNAL_NETWORK \
#     -m limit --limit 1/s      \
#     -j LOG --log-prefix "[Directed Broadcast] "
        iptables -A INPUT -i $EXTERNAL_INTERFACE -d $EXTERNAL_NETWORK -j DROP
#       iptables -A INPUT -i $EXTERNAL_INTERFACE -d $BROADCAST_NET \
#     -m limit --limit 1/s      \
#     -j LOG --log-prefix "[Directed Broadcast] "
        iptables -A INPUT  -i $EXTERNAL_INTERFACE -d $BROADCAST_NET -j DROP
        # Refuse limited broadcasts
#       iptables -A INPUT  -i $EXTERNAL_INTERFACE -d 255.255.255.255 \
#     -m limit --limit 1/s      \
#     -j LOG --log-prefix "[Limited Broadcast] "
        iptables -A INPUT  -i $EXTERNAL_INTERFACE -d 255.255.255.255 -j DROP
        #
        # Edit these to match the number of servers or connections
        # you support.
        #
        # X Window port allocation begins at 6000 and increments
        # for each additional server running from 6000 to 6063.
        XWINDOW_PORTS="6000:6063"                       # (TCP) X Windows
        # SSH starts at 1023 and works down to 513 for each additional
        # simultaneous incoming connection.
        SSH_HI_PORTS="513:1023"                                 # SSH Simultaneous Connections
        #
        # Iptables allows creation of customized chains.  The -l (log) flag no longer
        # exists.  This is a custom chain which allows logging of DROPped packets.
        #
        iptables -N LnD                                 # Define custom DROP chain
        iptables -A LnD -p tcp -m limit --limit 1/s -j LOG --log-prefix "[TCP drop] " --log-level=info
        iptables -A LnD -p udp -m limit --limit 1/s -j LOG --log-prefix "[UDP drop] " --log-level=info
        iptables -A LnD -p icmp -m limit --limit 1/s -j LOG --log-prefix "[ICMP drop] " --log-level=info
        iptables -A LnD -f -m limit --limit 1/s -j LOG --log-prefix "[FRAG drop] " --log-level=info
        iptables -A LnD -j DROP
        #
        # This custom chain logs, then REJECTs packets.
        #
        iptables -N LnR                                 # Define custom REJECT chain
        iptables -A LnR -p tcp -m limit --limit 1/s -j LOG --log-prefix "[TCP reject] " --log-level=info
        iptables -A LnR -p udp -m limit --limit 1/s -j LOG --log-prefix "[UDP reject] " --log-level=info
        iptables -A LnR -p icmp -m limit --limit 1/s -j LOG --log-prefix "[ICMP reject] " --log-level=info
        iptables -A LnR -f -m limit --limit 1/s -j LOG --log-prefix "[FRAG reject] " --log-level=info
        iptables -A LnR -j REJECT
        #
        # This chain logs, then DROPs "Xmas" and Null packets which might indicate a port-scan attempt
        #
        iptables -N ScanD                       # Define custom chain for possible port-scans
        iptables -A ScanD -p tcp -m limit --limit 1/s -j LOG --log-prefix "[TCP Scan?] "
        iptables -A ScanD -p udp -m limit --limit 1/s -j LOG --log-prefix "[UDP Scan?] "
        iptables -A ScanD -p icmp -m limit --limit 1/s -j LOG --log-prefix "[ICMP Scan?] "
        iptables -A ScanD -f -m limit --limit 1/s -j LOG --log-prefix "[FRAG Scan?] "
        iptables -A ScanD -j DROP
        #
  # This chain limits the number of new incoming connections to preventing DDoS attacks
  #
  iptables -N DDoS              # Define custom chain for possible DDoS attacks
  iptables -A DDoS -m limit --limit 12/s --limit-burst 24 -j RETURN
        iptables -A DDoS -j LOG --log-prefix "[DDos Attack?] "
        iptables -A DDoS -j DROP
```

```
        #
# This chain drops connections from IANA reserved IP blocks
#
        iptables -N IANA
        iptables -A IANA -p tcp -m limit --limit 1/s -j LOG --log-prefix "[IANA Reserved - TCP] " --log-level=info
        iptables -A IANA -p udp -m limit --limit 1/s -j LOG --log-prefix "[IANA Reserved - UDP] " --log-level=info
        iptables -A IANA -p icmp -m limit --limit 1/s -j LOG --log-prefix "[IANA Reserved - ICMP] " --log-level=info
        iptables -A IANA -f -m limit --limit 1/s -j LOG --log-prefix "[IANA Reserved - FRAG] " --log-level=info
        iptables -A IANA -j DROP
        #
# This chain drops connections from IPs in the firewall.banned file
#
        iptables -N Banned
        iptables -A Banned -p tcp -m limit --limit 1/s -j LOG --log-prefix "[TCP Banned] " --log-level=info
        iptables -A Banned -p udp -m limit --limit 1/s -j LOG --log-prefix "[UDP Banned] " --log-level=info
        iptables -A Banned -p icmp -m limit --limit 1/s -j LOG --log-prefix "[ICMP Banned] " --log-level=info
        iptables -A Banned -f -m limit --limit 1/s -j LOG --log-prefix "[FRAG Banned] " --log-level=info
        iptables -A Banned -j DROP
#
# Disallow packets frequently used by port-scanners
#
        # All of the bits are cleared
        iptables -A INPUT -p tcp --tcp-flags ALL NONE -j ScanD
        # SYN and FIN are both set
        iptables -A INPUT -p tcp --tcp-flags SYN,FIN SYN,FIN -j ScanD
        # SYN and RST are both set
        iptables -A INPUT -p tcp --tcp-flags SYN,RST SYN,RST -j ScanD
        # FIN and RST are both set
        iptables -A INPUT -p tcp --tcp-flags FIN,RST FIN,RST -j ScanD
        # FIN is the only bit set, without the expected accompanying ACK
        iptables -A INPUT -p tcp --tcp-flags ACK,FIN FIN -j ScanD
        # PSH is the only bit set, without the expected accompanying ACK
        iptables -A INPUT -p tcp --tcp-flags ACK,PSH PSH -j ScanD
        # URG is the only bit set, without the expected accompanying ACK
        iptables -A INPUT -p tcp --tcp-flags ACK,URG URG -j ScanD
        # SYN-Flood
        # (Request for new connection; large number indicate possible DDoS-type attack;
        #  same as --syn)
        iptables -A INPUT -p tcp --tcp-flags SYN,RST,ACK SYN -j DDoS
        # Enable broadcast echo Protection
        echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
        # Disable Source Routed Packets
        for f in /proc/sys/net/ipv4/conf/*/accept_source_route; do
          echo 0 > $f
        done
        # Enable TCP SYN Cookie Protection
        echo 1 > /proc/sys/net/ipv4/tcp_syncookies
        # Disable ICMP Redirect Acceptance
        for f in /proc/sys/net/ipv4/conf/*/accept_redirects; do
          echo 0 > $f
        done
        # Don't send Redirect Messages
        for f in /proc/sys/net/ipv4/conf/*/send_redirects; do
          echo 0 > $f
        done
        # Disable ICMP Redirect Acceptance
        for f in /proc/sys/net/ipv4/conf/*/accept_redirects; do
           echo 0 > $f
        done
        # Drop Spoofed Packets coming in on an interface, which if replied to,
        # would result in the reply going out a different interface.
        for f in /proc/sys/net/ipv4/conf/*/rp_filter; do
          echo 1 > $f
        done
        # Log packets with impossible addresses.
        for f in /proc/sys/net/ipv4/conf/*/log_martians; do
          echo 1 > $f
        done
        # Disallow fragmented packets.  This may not be as necessary as it once was.
        # Comment it out with # if desired.
#       iptables -A INPUT -f -i $EXTERNAL_INTERFACE -j LnD
#       iptables -A INPUT -f -i $INTERNAL_INTERFACE -j LnD
        #
        # Loopback
        #
        # Unlimited traffic on the loopback interface (lo)
```

```
    iptables -A INPUT -i $LOOPBACK_INTERFACE -j ACCEPT
    iptables -A OUTPUT -o $LOOPBACK_INTERFACE -j ACCEPT
    #
    # Refuse any connections to/from problem sites.
    #
    # /etc/firewall/firewall.banned contains a list of IPs
    # to block all access, both inbound and outbound.
    # The file should contain IP addresses with CIDR
    # netmask, one per line:
    #
    # NOTE: No comments are allowed in the file.
    #
    # 111.222.333.444/32              - To block a single IP address
    # 111.222.333.444/8               - To block a Class-A network
    # 111.222.333.444/16              - To block a Class-B network
    # 111.222.333.444/24              - To block a Class-C network
    #
    # The CIDR netmask number describes the number of bits
    # in the network portion of the address, and may be on
    # any boundary.
    #
    if [ -f /etc/firewall/firewall.banned ]; then
       while read BANNED; do
                iptables -A INPUT -i $EXTERNAL_INTERFACE -s $BANNED -j Banned
                iptables -A INPUT -i $EXTERNAL_INTERFACE -d $BANNED -j Banned
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -s $BANNED -j Banned
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -d $BANNED -j Banned
                iptables -A FORWARD -d $BANNED -j Banned
iptables -A FORWARD -s $BANNED -j Banned
       done < /etc/firewall/firewall.banned
       echo "firewall: Banned addresses added to rule set"
    else
       echo "firewall: Banned address/network file not found."
    fi
    #
    # Refuse connections from IANA-reserved blocks
    #
    if [ -f /etc/firewall/firewall.iana-reserved ]; then
       while read RESERVED; do
                iptables -A INPUT -i $EXTERNAL_INTERFACE -s $RESERVED -j IANA
                iptables -A INPUT -i $EXTERNAL_INTERFACE -d $RESERVED -j IANA
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -s $RESERVED -j IANA
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -d $RESERVED -j IANA
       done < /etc/firewall/firewall.iana-reserved
       echo "firewall: Connections from IANA-reserved addresses blocked"
    else
       echo "firewall: IANA-reserved address/network file not found."
    fi
    #
    # Localizations
    #
    # The /etc/firewall/firewall.local file should contain rules in
    # standard 'iptables' format.
    #
    if [ -f /etc/firewall/firewall.local.iptables ]; then
       . /etc/firewall/firewall.local.iptables
       echo "firewall: Local rules added"
    else
       echo "firewall: Local rules file not found."
    fi
    #
    # ICMP
    #
    # (4) Source Quench.
    # Incoming & outgoing requests to slow down (flow control)
    iptables -A INPUT -i $EXTERNAL_INTERFACE -p ICMP --icmp-type 4 \
       -s $ANYWHERE  -d $EXTERNAL_IP -j ACCEPT
    iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p ICMP --icmp-type 4 \
       -s $EXTERNAL_IP  -d $ANYWHERE -j ACCEPT
    if [ $MASQUERADING -gt 0 ]; then
       iptables -A FORWARD -p ICMP --icmp-type 4 -j ACCEPT
    fi
    # (12) Parameter Problem.
    # Incoming & outgoing error messages
    iptables -A INPUT -i $EXTERNAL_INTERFACE -p ICMP --icmp-type 12 \
       -s $ANYWHERE  -d $EXTERNAL_IP -j ACCEPT
```

```
iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p ICMP --icmp-type 12 \
   -s $EXTERNAL_IP  -d $ANYWHERE -j ACCEPT
if [ $MASQUERADING -gt 0 ]; then
   iptables -A FORWARD -p ICMP --icmp-type 12 -j ACCEPT
fi
# (3) Destination Unreachable, Service Unavailable.
# Incoming & outgoing size negotiation, service or
# destination unavailability, final traceroute response
iptables -A INPUT -i $EXTERNAL_INTERFACE -p ICMP --icmp-type 3 \
   -s $ANYWHERE  -d $EXTERNAL_IP -j ACCEPT
iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p ICMP --icmp-type 3 \
   -s $EXTERNAL_IP  -d $ANYWHERE -j ACCEPT
iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p ICMP --icmp-type \
   fragmentation-needed -s $EXTERNAL_IP  -d $ANYWHERE -j ACCEPT
if [ $MASQUERADING -gt 0 ]; then
   iptables -A FORWARD -p ICMP --icmp-type 3 -j ACCEPT
   iptables -A FORWARD -p ICMP --icmp-type fragmentation-needed -j ACCEPT
fi
# (11) Time Exceeded.
# Incoming & outgoing timeout conditions,
# also intermediate TTL response to traceroutes
iptables -A INPUT -i $EXTERNAL_INTERFACE -p ICMP --icmp-type 11 \
   -s $ANYWHERE  -d $EXTERNAL_IP -j ACCEPT
iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p ICMP --icmp-type 11 \
   -s $EXTERNAL_IP  -d $ANYWHERE -j ACCEPT
if [ $MASQUERADING -gt 0 ]; then
   iptables -A FORWARD -p ICMP --icmp-type 11 -j ACCEPT
fi
# (0 | 8) Allow OUTPUT pings to anywhere.
if [ $OUTBOUND_PING -gt 0 ]; then
   iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p ICMP --icmp-type 8 \
               -s $EXTERNAL_IP  -d $ANYWHERE -j ACCEPT
   iptables -A INPUT -i $EXTERNAL_INTERFACE -p ICMP --icmp-type 0 \
               -s $ANYWHERE  -d $EXTERNAL_IP -j ACCEPT
   if [ $MASQUERADING -gt 0 ]; then
     iptables -A FORWARD -p ICMP --icmp-type 8 -s $INTERNAL_NETWORK -j ACCEPT
     iptables -A FORWARD -p ICMP --icmp-type 0 -d $INTERNAL_NETWORK -j ACCEPT
   fi
   if [ $VERBOSE -gt 0 ]; then
               echo "firewall: Outbound ping enabled"
   fi
fi
# (0 | 8) Allow incoming pings from anywhere
#       (stops at firewall).
if [ $INBOUND_PING -gt 0 ]; then
   iptables -A INPUT -i $EXTERNAL_INTERFACE -p ICMP --icmp-type 8 \
               -s $ANYWHERE  -d $EXTERNAL_IP -j ACCEPT
   iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p ICMP --icmp-type 0 \
               -s $EXTERNAL_IP  -d $ANYWHERE -j ACCEPT
   if [ $VERBOSE -gt 0 ]; then
               echo "firewall: Inbound ping enabled"
   fi
fi
#
# Unprivileged Ports
# Avoid ports subject to protocol and system administration problems.
#

NFS_PORT="2049"                                          # (TCP/UDP) NFS
OPENWINDOWS_PORT="2000"                                  # (TCP) Openwindows
SOCKS_PORT="1080"                                # (TCP) Socks
# Openwindows: establishing a connection
iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP -m state --state NEW \
--dport $OPENWINDOWS_PORT -s $EXTERNAL_IP -d $ANYWHERE -j LnR
# Openwindows: incoming connection
iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state NEW \
--dport $OPENWINDOWS_PORT -d $EXTERNAL_IP -j LnD
# X Window: establishing a remote connection
iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP -m state --state NEW \
--dport $XWINDOW_PORTS -s $EXTERNAL_IP -d $ANYWHERE -j LnR
# X Window: incoming connection attempt
iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state NEW \
--dport $XWINDOW_PORTS -d $EXTERNAL_IP -j LnD
# SOCKS: establishing a connection
iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP -m state --state NEW \
--dport $SOCKS_PORT -s $EXTERNAL_IP -d $ANYWHERE  -j LnR
```

```
                # SOCKS: incoming connection
                iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state NEW \
                --dport         $SOCKS_PORT -d $EXTERNAL_IP -j LnD
                # NFS: TCP connections
                iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state NEW \
                --dport $NFS_PORT -d $EXTERNAL_IP -j LnD
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP -m state --state NEW \
                --dport         $NFS_PORT -d $ANYWHERE -j LnR
                # NFS: UDP connections
                iptables -A INPUT -i $EXTERNAL_INTERFACE -p UDP \
                --dport $NFS_PORT -d $EXTERNAL_IP -j LnD
                # NFS: incoming request (normal UDP mode)
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p UDP \
                --dport $NFS_PORT -d $ANYWHERE -j LnR
        #
        # DNAT/SNAT Port Forwarding
        #
                if [ $PORT_FORWARD -gt 0 ]; then
          if [ -f /etc/firewall/firewall.nat ]; then
            while read IP_PORT; do
              # extract the protocols, IPs and ports
                        NAT_TYPE=$(echo "$IP_PORT" | awk '{print $1}')
              NAT_EXT_PORT=$(echo "$IP_PORT" | awk '{print $2}')
              NAT_INT_IP=$(echo "$IP_PORT" | awk '{print $3}')
              NAT_INT_PORT=$(echo "$IP_PORT" | awk '{print $4}')
              # write the rules!
              # this is the prerouting dnat
              iptables -A PREROUTING -t nat -p $NAT_TYPE -d $EXTERNAL_IP --dport $NAT_EXT_PORT -j DNAT \
               --to-destination $NAT_INT_IP:$NAT_INT_PORT
              # This allows packets from external->internal
              iptables -A FORWARD -i $EXTERNAL_INTERFACE -o $INTERNAL_INTERFACE -p $NAT_TYPE  \
               -d $NAT_INT_IP --dport $NAT_INT_PORT -m state \
               --state NEW,ESTABLISHED,RELATED -j ACCEPT
              # This allows packets from internal->external
              iptables -A FORWARD -i $INTERNAL_INTERFACE -o $EXTERNAL_INTERFACE -p $NAT_TYPE  \
               -s $NAT_INT_IP --sport $NAT_INT_PORT -m state \
               --state NEW,ESTABLISHED,RELATED -j ACCEPT
              # This enables access to the 'public' server from the internal network
              iptables -t nat -A POSTROUTING -d $NAT_INT_IP -s $INTERNAL_NETWORK \
               -p $NAT_TYPE --dport $NAT_INT_PORT -j SNAT --to $INTERNAL_IP
              echo firewall: dnat: $NAT_TYPE:$EXTERNAL_IP:$NAT_EXT_PORT - $NAT_INT_IP:$NAT_INT_PORT
            done < /etc/firewall/firewall.nat
           # unset some variables
            unset IP_PORT
                      unset NAT_TYPE
            unset NAT_EXT_PORT
            unset NAT_INT_IP
            unset NAT_INT_PORT
                    else
            echo "firewall.nat (port-forwarding table) not found!  Port-forwarding not enabled."
                      fi
        fi
                #
                # NOTE:
                #     The symbolic names used in /etc/services for the port numbers
                #     vary by supplier.
                #
                # Required Services
                #
                # DNS client modes (53)
                #
                if [ $DNS_CLIENT -gt 0 ]; then
                   iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p UDP \
                                --sport $UNPRIVPORTS --dport 53 -s $EXTERNAL_IP \
                                -d $ANYWHERE -j ACCEPT
                   iptables -A INPUT -i $EXTERNAL_INTERFACE -m state --state ESTABLISHED,RELATED -p UDP --
sport 53 \
                                --dport $UNPRIVPORTS -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT
                   if [ $MASQUERADING -gt 0 ]; then
                     iptables -A FORWARD -p UDP -s $INTERNAL_NETWORK --sport $UNPRIVPORTS --dport 53 -j
ACCEPT
                     iptables -A FORWARD -p UDP -d $INTERNAL_NETWORK --sport 53 --dport $UNPRIVPORTS -j
ACCEPT
                   fi
                   # TCP client-to-server requests are allowed by the protocol
                   # if UDP requests fail. This is rarely seen. Usually, clients
```

```
                # use TCP as a secondary name server for zone transfers from
                # their primary name servers, and as hackers.
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP --sport \
                        $UNPRIVPORTS --dport 53 -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT
                iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
                        --sport 53 --dport $UNPRIVPORTS -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT
                if [ $MASQUERADING -gt 0 ]; then
                  iptables -A FORWARD -p TCP -s $INTERNAL_NETWORK --sport $UNPRIVPORTS --dport 53 -j
ACCEPT
                  iptables -A FORWARD -m state --state ESTABLISHED,RELATED -p TCP -d $INTERNAL_NETWORK \
                        --sport 53 --dport $UNPRIVPORTS -j ACCEPT
                fi
                if [ $VERBOSE -gt 0 ]; then
                        echo "firewall: DNS client enabled"
                fi
              fi
              #
              # DNS server modes (53)
              #
              #
              # DNS caching & forwarding name server
              #
              if [ $DNS_CACHING_SERVER -gt 0 ]; then
                # Server-to-server query or response
                # Caching only name server uses UDP, not TCP
                iptables -A INPUT -i $EXTERNAL_INTERFACE -p UDP \
              --sport 53 --dport 53 -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p UDP \
              --sport 53 --dport 53 -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT
                if [ $VERBOSE -gt 0 ]; then
                        echo "firewall: DNS Caching server enabled"
                fi
              fi
              #
              # DNS full name server
              #
              if [ $DNS_FULL_SERVER -gt 0 ]; then
                # Client-to-server DNS transaction.
                iptables -A INPUT -i $EXTERNAL_INTERFACE -p UDP \
              --sport $UNPRIVPORTS --dport 53 -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p UDP \
              --sport 53 --dport $UNPRIVPORTS -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT
                # Zone Transfers.
                # Due to the potential danger of zone transfers,
                # allow TCP traffic to only specific secondaries.
          # /etc/firewall/firewall.dns contains a list of
          # secondary, tertiary, etc. domain name servers with which
          # zone transfers are allowed.  The file should contain IP
          # addresses with CIDR netmask, one per line:
        if [ -f /etc/firewall/firewall.dns ]; then
            while read DNS_SECONDARY; do
          iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP \
            --sport $UNPRIVPORTS --dport 53 -s $DNS_SECONDARY -d $EXTERNAL_IP -j ACCEPT
          iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
            --sport 53 --dport $UNPRIVPORTS -s $EXTERNAL_IP -d $DNS_SECONDARY -j ACCEPT
          done < /etc/firewall/firewall.dns
        else
          echo "firewall: ** No secondary DNS configured **"
        fi
          if [ $VERBOSE -gt 0 ]; then
            echo "firewall: DNS Full server enabled"
          fi
        fi
              #
              # AUTH (113) - Allowing your outgoing AUTH requests as a client
              #
              if [ $AUTH_CLIENT -gt 0 ]; then
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP \
              --sport $UNPRIVPORTS --dport 113 -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT
                iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
              --sport 113 --dport $UNPRIVPORTS -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT
                if [ $MASQUERADING -gt 0 ]; then
                  iptables -A FORWARD -p TCP -s $INTERNAL_NETWORK --sport $UNPRIVPORTS --dport 113 -j
ACCEPT
                  iptables -A FORWARD -m state --state ESTABLISHED,RELATED -p TCP -d $INTERNAL_NETWORK \
                        --sport 113 --dport $UNPRIVPORTS -j ACCEPT
```

```
                fi
                if [ $VERBOSE -gt 0 ]; then
                        echo "firewall: Auth client enabled"
                fi
              fi
              # AUTH server (113)
              if [ $AUTH_SERVER -gt 0 ]; then
                # Accepting incoming AUTH requests
                iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP \
              --sport $UNPRIVPORTS --dport 113 -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
              --sport 113 --dport $UNPRIVPORTS -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT
                if [ $VERBOSE -gt 0 ]; then
                        echo "firewall: Auth server enabled"
                fi
              else
                # Rejecting incoming AUTH requests
                iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP \
                        --dport 113 -d $EXTERNAL_IP -j LnR
                if [ $VERBOSE -gt 0 ]; then
                        echo "firewall: Auth server requests will be rejected"
                fi
              fi
              #
              # TCP Services on selected ports.
              #
              #
              # Sending Mail through a remote SMTP server (25)
              #
              if [ $SMTP_REMOTE_SERVER -gt 0 ]; then
                # SMTP client to an ISP account without a local server
                for SMTP_SRVR in ${SMTP_SERVER}; do
                  iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP \
                --sport $UNPRIVPORTS --dport 25 -s $EXTERNAL_IP -d $SMTP_SRVR -j ACCEPT
                  iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
                --sport 25 --dport $UNPRIVPORTS -s $SMTP_SRVR -d $EXTERNAL_IP -j ACCEPT
                  if [ $MASQUERADING -gt 0 ]; then
                    iptables -A FORWARD -p TCP -s $INTERNAL_NETWORK -d $SMTP_SRVR --sport $UNPRIVPORTS
--dport 25 -j ACCEPT
                    iptables -A FORWARD -m state --state ESTABLISHED,RELATED -p TCP -s $SMTP_SRVR -d
$INTERNAL_NETWORK \
                        --sport 25 --dport $UNPRIVPORTS -j ACCEPT
                  fi
                  if [ $VERBOSE -gt 0 ]; then
                        echo "firewall: Clients may access remote SMTP server: ${SMTP_SRVR}"
                  fi
                done
              fi
              #
              # Sending Mail through a local SMTP server (25)
              #
              if [ $SMTP_LOCAL_SERVER -gt 0 ]; then
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP \
              --sport $UNPRIVPORTS --dport 25 -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT
                iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
              --sport 25 --dport $UNPRIVPORTS -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT
                # Receiving Mail as a Local SMTP server (25)
                iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP \
              --sport $UNPRIVPORTS --dport 25 -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
              --sport 25 --dport $UNPRIVPORTS -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT
                if [ $VERBOSE -gt 0 ]; then
                        echo "firewall: SMTP Local server enabled"
                fi
              fi
              #
              # POP3 (110) - Retrieving Mail as a POP3 client
              #
              if [ $POP3_CLIENT -gt 0 ]; then
                for POP_SRVR in ${POP_SERVER}; do
                  iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP \
                --sport $UNPRIVPORTS --dport 110 -s $EXTERNAL_IP -d $POP_SRVR -j ACCEPT
                  iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
                --sport 110 --dport $UNPRIVPORTS -s $POP_SRVR -d $EXTERNAL_IP -j ACCEPT
                  if [ $MASQUERADING -gt 0 ]; then
```

```
                    iptables -A FORWARD -p TCP -s $INTERNAL_NETWORK -d $POP_SRVR --sport $UNPRIVPORTS -
-dport 110 -j ACCEPT
                    iptables -A FORWARD -m state --state ESTABLISHED,RELATED -p TCP -s $POP_SRVR -d
$INTERNAL_NETWORK \
                        --sport 110 --dport $UNPRIVPORTS -j ACCEPT
                  fi
                  if [ $VERBOSE -gt 0 ]; then
                        echo "firewall: Clients may access remote POP-3 server: ${POP_SRVR}"
                  fi
                done
              fi
              #
              # POP3 (110) - Hosting a POP3 server for remote clients
              #
              if [ $POP3_SERVER -gt 0 ]; then
                for MY_POP3_CLIENT in ${MY_POP3_CLIENTS}; do
                  iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP \
                --sport $UNPRIVPORTS --dport 110 -s $MY_POP3_CLIENT -d $EXTERNAL_IP -j ACCEPT
                  iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
                --sport 110 --dport $UNPRIVPORTS -s $EXTERNAL_IP -d $MY_POP3_CLIENT -j ACCEPT
                  if [ $VERBOSE -gt 0 ]; then
                    echo "firewall: Remote site ${MY_POP3_CLIENT} may access local POP-3 server"
                  fi
                done
              fi
              #
              # IMAP (143) - Retrieving Mail as an IMAP client
              #
              if [ $IMAP_CLIENT -gt 0 ]; then
                for IMAP_SRVR in ${MY_IMAP_SERVER}; do
                  iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP \
                --sport $UNPRIVPORTS --dport 143 -s $EXTERNAL_IP -d $IMAP_SRVR -j ACCEPT
                  iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
                --sport 143 --dport $UNPRIVPORTS -s $IMAP_SRVR -d $EXTERNAL_IP -j ACCEPT
                  if [ $MASQUERADING -gt 0 ]; then
                    iptables -A FORWARD -p TCP -s $INTERNAL_NETWORK -d $IMAP_SRVR --sport $UNPRIVPORTS
--dport 143 -j ACCEPT
                    iptables -A FORWARD -m state --state ESTABLISHED,RELATED -p TCP -s $IMAP_SRVR -d
$INTERNAL_NETWORK \
                        --sport 143 --dport $UNPRIVPORTS -j ACCEPT
                  fi
                  if [ $VERBOSE -gt 0 ]; then
                      echo "firewall: Clients may access remote IMAP server: ${IMAP_SRVR}"
                  fi
                done
              fi
              #
              # IMAP (143) - Hosting an IMAP server for remote clients
              #
              if [ $IMAP_SERVER -gt 0 ]; then
                for MY_IMAP_CLIENT in ${MY_IMAP_CLIENTS}; do
                  iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP \
                --sport $UNPRIVPORTS --dport 143 -s $MY_IMAP_CLIENT -d $EXTERNAL_IP -j ACCEPT
                  iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
                --sport 143 --dport $UNPRIVPORTS -s $EXTERNAL_IP -d $MY_IMAP_CLIENTS -j ACCEPT
                  if [ $VERBOSE -gt 0 ]; then
                      echo "firewall: Remote site ${MY_IMAP_CLIENT} may access local IMAP server"
                  fi
                done
              fi
              #
              # IMAPS (993) - Retrieving Mail as an Secure IMAP client
              #
              if [ $IMAPS_CLIENT -gt 0 ]; then
                for IMAPS_SRVR in ${MY_IMAPS_SERVER}; do
                  iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP \
                --sport $UNPRIVPORTS --dport 993 -s $EXTERNAL_IP -d $IMAPS_SRVR -j ACCEPT
                  iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
                --sport 993 --dport $UNPRIVPORTS -s $IMAPS_SRVR -d $EXTERNAL_IP -j ACCEPT
                  if [ $MASQUERADING -gt 0 ]; then
                    iptables -A FORWARD -p TCP -s $INTERNAL_NETWORK -d $IMAP_SRVR --sport $UNPRIVPORTS
--dport 993 -j ACCEPT
                    iptables -A FORWARD -m state --state ESTABLISHED,RELATED -p TCP -s $IMAPS_SRVR -d
$INTERNAL_NETWORK \
                        --sport 993 --dport $UNPRIVPORTS -j ACCEPT
                  fi
```

```
                    if [ $VERBOSE -gt 0 ]; then
                            echo "firewall: Clients may access remote Secure IMAP server: ${IMAPS_SRVR}"
                    fi
                  done
                fi
                #
                # IMAPS (993) - Hosting a Secure IMAP server for remote clients
                #
                if [ $IMAPS_SERVER -gt 0 ]; then
                  for MY_IMAPS_CLIENT in ${MY_IMAPS_CLIENTS}; do
                   iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP \
                --sport $UNPRIVPORTS --dport 993 -s $MY_IMAP_CLIENT -d $EXTERNAL_IP -j ACCEPT
                   iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
                --sport 993 --dport $UNPRIVPORTS -s $EXTERNAL_IP -d $MY_IMAP_CLIENT -j ACCEPT
                    if [ $VERBOSE -gt 0 ]; then
                            echo "firewall: Remote site ${MY_IMAPS_CLIENT} may access local Secure IMAP server"
                    fi
                  done
                fi
                #
                # NNTP (119) - Reading and posting news as a Usenet client
                #
                if [ $NNTP_CLIENT -gt 0 ]; then
                  for NEWS_SRVR in ${NEWS_SERVER}; do
                    iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP \
                  --sport $UNPRIVPORTS --dport 119 -s $EXTERNAL_IP -d $NEWS_SRVR -j ACCEPT
                    iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
                  --sport 119 --dport $UNPRIVPORTS -s $NEWS_SRVR -d $EXTERNAL_IP -j ACCEPT
                    if [ $MASQUERADING -gt 0 ]; then
                      iptables -A FORWARD -p TCP -s $INTERNAL_NETWORK -d $NEWS_SRVR --sport
$UNPRIVPORTS --dport 119 -j ACCEPT
                      iptables -A FORWARD -m state --state ESTABLISHED,RELATED -p TCP -s $NEWS_SRVR -d
$INTERNAL_NETWORK \
                              --sport 119 --dport $UNPRIVPORTS -j ACCEPT
                    fi
                    if [ $VERBOSE -gt 0 ]; then
                            echo "firewall: Clients may access remote NNTP server: ${NEWS_SRVR}"
                    fi
                  done
                fi
                #
                # NNTP (119) - Hosting a Usenet news server for remote clients
                #
                if [ $NNTP_SERVER -gt 0 ]; then
                  for NNTP_CLIENT in ${MY_NNTP_CLIENTS}; do
                    iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP \
                  --sport $UNPRIVPORTS --dport 119 -s $NNTP_CLIENT -d $EXTERNAL_IP -j ACCEPT

                    iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
                  --sport 119 --dport $UNPRIVPORTS -s $EXTERNAL_IP -d $NNTP_CLIENT -j ACCEPT

                    if [ $VERBOSE -gt 0 ]; then
                            echo "firewall: Remote client ${NNTP_CLIENT} may access local NNTP server"
                    fi
                  done
                fi
                #
                # NNTP (119) - Allowing peer news feeds for a local Usenet server
                #
                if [ $NNTP_NEWS_FEED -gt 0 ]; then
                   iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP \
                --sport $UNPRIVPORTS --dport 119 -s $EXTERNAL_IP -d $MY_NEWS_FEED -j ACCEPT
                   iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
                --sport 119 --dport $UNPRIVPORTS -s $MY_NEWS_FEED -d $EXTERNAL_IP -j ACCEPT
                   if [ $VERBOSE -gt 0 ]; then
                           echo "firewall: External NNTP News feed access enabled"
                   fi
                fi
                #
        # Secure NNTP (563) - Reading and posting news as a Usenet client over SSL
        # Submitted by Renaud Colinet
                #
        if [ $NNTPS_CLIENT -gt 0 ]; then
                  for SNEWS_SRVR in ${SNEWS_SERVER}; do
            iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP \
          --sport $UNPRIVPORTS --dport 563 -s $EXTERNAL_IP -d $SNEWS_SRVR -j ACCEPT
```

```
        iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
      --sport 563 --dport $UNPRIVPORTS -s $SNEWS_SERVER -d $EXTERNAL_IP -j ACCEPT

        if [ $MASQUERADING -gt 0 ]; then
          iptables -A FORWARD -p TCP -s $INTERNAL_NETWORK -d $SNEWS_SRVR --sport $UNPRIVPORTS --dport
563 -j ACCEPT
          iptables -A FORWARD -m state --state ESTABLISHED,RELATED -p TCP -s $SNEWS_SRVR -d
$INTERNAL_NETWORK \
           --sport 563 --dport $UNPRIVPORTS -j ACCEPT
        fi
        if [ $VERBOSE -gt 0 ]; then
          echo "firewall: Clients may access remote secure NNTP server: ${SNEWS_SRVR}"
        fi
              done
    fi
            #
            # TELNET (23) - Allowing outgoing client access to remote sites
            #
            if [ $TELNET_CLIENT -gt 0 ]; then
              iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP \
            --sport $UNPRIVPORTS --dport 23 -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT
              iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
            --sport 23 --dport $UNPRIVPORTS -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT
              if [ $MASQUERADING -gt 0 ]; then
                iptables -A FORWARD -p TCP -s $INTERNAL_NETWORK --sport $UNPRIVPORTS --dport 23 -j
ACCEPT
                iptables -A FORWARD -m state --state ESTABLISHED,RELATED -p TCP -d $INTERNAL_NETWORK \
                        --sport 23 --dport $UNPRIVPORTS -j ACCEPT
              fi
              if [ $VERBOSE -gt 0 ]; then
                        echo "firewall: Clients may access remote TELNET servers"
              fi
            fi
            #
            # TELNET (23) - Allowing incoming access to your local server
            # Note:  Not recommended! Suggest SSH instead!
            #
            if [ $TELNET_SERVER -gt 0 ]; then
              for MY_TELNET_CLIENT in ${MY_TELNET_CLIENTS}; do
                iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP \
              --sport $UNPRIVPORTS --dport 23 -s $MY_TELNET_CLIENTS -d $EXTERNAL_IP -j ACCEPT
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
              --sport 23 --dport $UNPRIVPORTS -s $EXTERNAL_IP -d $MY_TELNET_CLIENTS -j ACCEPT
                 if [ $VERBOSE -gt 0 ]; then
                          echo "firewall: Remote site ${MY_TELNET_CLIENT} may access local TELNET server"
                 fi
              done
            fi
            #
            # SSH Client (22) - Allowing client access to remote SSH servers
            #
            if [ $SSH_CLIENT -gt 0 ]; then
              iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP \
            --sport $UNPRIVPORTS --dport 22 -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT
              iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
            --sport 22 --dport $UNPRIVPORTS -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT
              iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP \
            --sport $SSH_HI_PORTS --dport 22 -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT
              iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
            --sport 22 --dport $SSH_HI_PORTS -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT
              if [ $MASQUERADING -gt 0 ]; then
                iptables -A FORWARD -p TCP -s $INTERNAL_NETWORK --sport $UNPRIVPORTS --dport 22 -j
ACCEPT
                iptables -A FORWARD -m state --state ESTABLISHED,RELATED -p TCP -d $INTERNAL_NETWORK \
            --sport 22 --dport $UNPRIVPORTS -j ACCEPT
                iptables -A FORWARD -p TCP -s $INTERNAL_NETWORK --sport $SSH_HI_PORTS --dport 22 -j
ACCEPT
                iptables -A FORWARD -m state --state ESTABLISHED,RELATED -p TCP -d $INTERNAL_NETWORK \
                        --sport 22 --dport $SSH_HI_PORTS -j ACCEPT
              fi
              if [ $VERBOSE -gt 0 ]; then
                        echo "firewall: Clients may access remote SSH servers"
              fi
            fi
            #
```

```
                # SSH (see config) - Allowing remote client access to your local SSH server
                #
                if [ $SSH_SERVER -gt 0 ]; then
                  for MY_SSH_CLIENT in ${MY_SSH_CLIENTS}; do
                     iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP \
                    --sport $UNPRIVPORTS --dport $SSH_PORT -s $MY_SSH_CLIENT -d $EXTERNAL_IP -j ACCEPT

                     iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
                    --sport $SSH_PORT --dport $UNPRIVPORTS -s $EXTERNAL_IP -d $MY_SSH_CLIENT -j ACCEPT

                     iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP \
                    --sport $SSH_HI_PORTS --dport $SSH_PORT -s $MY_SSH_CLIENT -d $EXTERNAL_IP -j ACCEPT

                     iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
                     --sport $SSH_PORT --dport $SSH_HI_PORTS -s $EXTERNAL_IP -d $MY_SSH_CLIENT -j ACCEPT

                    if [ $VERBOSE -gt 0 ]; then
                                echo "firewall: Remote site ${MY_SSH_CLIENT} may access local SSH server"
                    fi
                done
                fi
                #
                # FTP (20, 21) - Allowing outgoing client access to remote FTP servers
                #
                if [ $FTP_CLIENT -gt 0 ]; then
                   # Outgoing request
                   iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
                --sport 21 --dport $UNPRIVPORTS -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT
                   iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP -m state --state NEW,ESTABLISHED \
                --sport $UNPRIVPORTS --dport 21 -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT
                   # Normal Port mode FTP data channels
                   iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state NEW \
                --sport 20 --dport $UNPRIVPORTS -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT
                   iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
                --sport $UNPRIVPORTS --dport 20 -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT
                   # Passive mode FTP data channels
                   iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
                --sport $UNPRIVPORTS --dport $UNPRIVPORTS -s $ANYWHERE -d $EXTERNAL_IP -j ACCEP
                   iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP -m state --state NEW,ESTABLISHED \
                --sport $UNPRIVPORTS --dport $UNPRIVPORTS -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT
                   if [ $MASQUERADING -gt 0 ]; then
                      iptables -A FORWARD -p TCP -s $INTERNAL_NETWORK --sport $UNPRIVPORTS --dport 20:21 -j
ACCEPT
                      iptables -A FORWARD -p TCP -d $INTERNAL_NETWORK --sport 20:21 --dport $UNPRIVPORTS -j
ACCEPT
                      iptables -A FORWARD -p TCP -s $INTERNAL_NETWORK --sport $UNPRIVPORTS --dport
$UNPRIVPORTS -j ACCEPT
                      iptables -A FORWARD -p TCP -d $INTERNAL_NETWORK --sport $UNPRIVPORTS --dport
$UNPRIVPORTS -j ACCEPT
                   fi
                   if [ $VERBOSE -gt 0 ]; then
                                echo "firewall: Clients may access remote FTP servers"
                   fi
                fi
                #
                # FTP (20, 21) - Allowing incoming access to your local FTP server
                #
    if [ $FTP_SERVER -gt 0 ]; then
      for MY_FTP_CLIENT in ${MY_FTP_CLIENTS}; do
        # Incoming request
          iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state NEW,ESTABLISHED \
       --sport $UNPRIVPORTS --dport 21 -s $MY_FTP_CLIENT -d $EXTERNAL_IP -j ACCEPT
         iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
       --sport 21 --dport $UNPRIVPORTS -s $EXTERNAL_IP -d $MY_FTP_CLIENT -j ACCEPT
        # Normal Port mode FTP data channel responses
          iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP \
       --sport 20 --dport $UNPRIVPORTS -s $EXTERNAL_IP -d $MY_FTP_CLIENT -j ACCEPT
          iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
       --sport $UNPRIVPORTS --dport 20 -s $MY_FTP_CLIENT -d $EXTERNAL_IP -j ACCEPT
        # Passive mode FTP data channel responses
          iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state NEW,ESTABLISHED \
       --sport $UNPRIVPORTS --dport $UNPRIVPORTS -s $MY_FTP_CLIENT -d $EXTERNAL_IP -j ACCEPT
          iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
       --sport $UNPRIVPORTS --dport $UNPRIVPORTS -s $EXTERNAL_IP -d $MY_FTP_CLIENT -j ACCEPT
         if [ $VERBOSE -gt 0 ]; then
           echo "firewall: Remote site ${MY_FTP_CLIENT} may access local FTP server"
```

```
        fi
            done
          fi
          #
          # HTTP (80) - Accessing remote web sites as a client
          #
          if [ $HTTP_CLIENT -gt 0 ]; then
             iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP \
          --sport $UNPRIVPORTS --dport 80 -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT
             iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
          --sport 80 --dport $UNPRIVPORTS -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT
             if [ $MASQUERADING -gt 0 ]; then
                iptables -A FORWARD -p TCP -s $INTERNAL_NETWORK --sport $UNPRIVPORTS --dport 80 -j
ACCEPT
                iptables -A FORWARD -p TCP -d $INTERNAL_NETWORK --sport 80 --dport $UNPRIVPORTS -j
ACCEPT
             fi
             if [ $VERBOSE -gt 0 ]; then
                        echo "firewall: Clients may access remote HTTP servers"
             fi
          fi
          #
          # HTTP (80) - Allowing remote access to a local web server
          #
          if [ $HTTP_SERVER -gt 0 ]; then
            for HTTP_CLIENT in ${MY_HTTP_CLIENTS}; do
              iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP \
            --sport $UNPRIVPORTS --dport 80 -s $HTTP_CLIENT -d $EXTERNAL_IP -j ACCEPT
              iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
            --sport 80 --dport $UNPRIVPORTS -s $EXTERNAL_IP -d $HTTP_CLIENT -j ACCEPT
              iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP \
            --sport $UNPRIVPORTS --dport 8080 -s $HTTP_CLIENT -d $EXTERNAL_IP -j ACCEPT
              iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
            --sport 8080 --dport $UNPRIVPORTS -s $EXTERNAL_IP -d $HTTP_CLIENT -j ACCEPT

              if [ $VERBOSE -gt 0 ]; then
                         echo "firewall: Remote client ${HTTP_CLIENT} may access local HTTP server"
              fi
            done
          fi
          #
          # HTTPS (443) - Accessing remote web sites over SSL as a client
          #
          if [ $HTTPS_CLIENT -gt 0 ]; then
             iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP \
          --sport $UNPRIVPORTS --dport 443 -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT

             iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
          --sport 443 --dport $UNPRIVPORTS -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT

             if [ $MASQUERADING -gt 0 ]; then
                iptables -A FORWARD -p TCP -s $INTERNAL_NETWORK --sport $UNPRIVPORTS --dport 443 -j
ACCEPT
                iptables -A FORWARD -m state --state ESTABLISHED,RELATED -p TCP -d $INTERNAL_NETWORK \
                        --sport 443 --dport $UNPRIVPORTS -j ACCEPT
             fi
             if [ $VERBOSE -gt 0 ]; then
                        echo "firewall: Clients may access remote HTTPS servers"
             fi
          fi
          #
          # HTTPS (443) - Allowing remote access to a local SSL web server
          #
          if [ $HTTPS_SERVER -gt 0 ]; then
            for HTTPS_CLIENT in ${MY_HTTPS_CLIENTS}; do
              iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP \
            --sport $UNPRIVPORTS --dport 443 -s $HTTPS_CLIENT -d $EXTERNAL_IP -j ACCEPT
              iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
            --sport 443 --dport $UNPRIVPORTS -s $EXTERNAL_IP -d $HTTPS_CLIENT -j ACCEPT
              if [ $VERBOSE -gt 0 ]; then
                         echo "firewall: Remote client ${HTTPS_CLIENT} may access local HTTPS server"
              fi
            done
          fi
          #
          # HTTP Proxy Client (8008/8080)
```

```
#
if [ $HTTP_PROXY -gt 0 ]; then
   iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP \
--sport $UNPRIVPORTS --dport $WEB_PROXY_PORT -s $EXTERNAL_IP -d $WEB_PROXY_SERVER -j
ACCEPT
   iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
--sport $WEB_PROXY_PORT --dport $UNPRIVPORTS -s $WEB_PROXY_SERVER -d $EXTERNAL_IP -j
ACCEPT
   if [ $MASQUERADING -gt 0 ]; then
     iptables -A FORWARD -p TCP -s $INTERNAL_NETWORK --sport $UNPRIVPORTS --dport
$WEB_PROXY_PORT -j ACCEPT
     iptables -A FORWARD -m state --state ESTABLISHED,RELATED -p TCP -d $INTERNAL_NETWORK \
          --sport $WEB_PROXY_PORT --dport $UNPRIVPORTS -j ACCEPT
   fi
   if [ $VERBOSE -gt 0 ]; then
          echo "firewall: Clients may access remote sites via HTTP Proxy Server"
   fi
fi
#
# FINGER (79) - Accessing remote finger servers as a client
#
if [ $FINGER_CLIENT -gt 0 ]; then
   iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP \
--sport $UNPRIVPORTS --dport 79 -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT
   iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
--sport 79 --dport $UNPRIVPORTS -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT
   if [ $MASQUERADING -gt 0 ]; then
     iptables -A FORWARD -p TCP -s $INTERNAL_NETWORK --sport $UNPRIVPORTS --dport 79 -j
ACCEPT
     iptables -A FORWARD -m state --state ESTABLISHED,RELATED -p TCP -d $INTERNAL_NETWORK \
          --sport 79 --dport $UNPRIVPORTS -j ACCEPT
   fi
   if [ $VERBOSE -gt 0 ]; then
          echo "firewall: Clients may access remote FINGER servers"
   fi
fi
#
# FINGER (79) - Allowing remote client access to a local finger server (dangerous!)
#
if [ $FINGER_SERVER -gt 0 ]; then
  for FINGER_CLIENT in $MY_FINGER_CLIENTS}; do
    iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP \
  --sport $UNPRIVPORTS --dport 79 -s $FINGER_CLIENT -d $EXTERNAL_IP -j ACCEPT
    iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
  --sport 79 --dport $UNPRIVPORTS -s $EXTERNAL_IP -d $FINGER_CLIENT -j ACCEPT
    if [ $VERBOSE -gt 0 ]; then
          echo "firewall: Remote client ${FINGER_CLIENT} may access local FINGER server"
    fi
  done
fi
#
# WHOIS (43) - Accessing a remote WHOIS server as a client
#
if [ $WHOIS_CLIENT -gt 0 ]; then
   iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP \
--sport $UNPRIVPORTS --dport 43 -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT
   iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
--sport 43 --dport $UNPRIVPORTS -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT
   if [ $MASQUERADING -gt 0 ]; then
     iptables -A FORWARD -p TCP -s $INTERNAL_NETWORK --sport $UNPRIVPORTS --dport 43 -j
ACCEPT
     iptables -A FORWARD -m state --state ESTABLISHED,RELATED -p TCP -d $INTERNAL_NETWORK \
          --sport 43 --dport $UNPRIVPORTS -j ACCEPT
   fi
   if [ $VERBOSE -gt 0 ]; then
          echo "firewall: Clients may access remote WHOIS servers"
   fi
fi
#
# GOPHER (70) - Accessing a remote GOPHER server as a client
#
if [ $GOPHER_CLIENT -gt 0 ]; then
   iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP \
--sport $UNPRIVPORTS --dport 70 -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT

   iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
```

```
            --sport 70 --dport $UNPRIVPORTS -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT

              if [ $MASQUERADING -gt 0 ]; then
                iptables -A FORWARD -p TCP -s $INTERNAL_NETWORK --sport $UNPRIVPORTS --dport 70 -j
ACCEPT
                iptables -A FORWARD -m state --state ESTABLISHED,RELATED -p TCP -d $INTERNAL_NETWORK \
                        --sport 70 --dport $UNPRIVPORTS -j ACCEPT
              fi
              if [ $VERBOSE -gt 0 ]; then
                        echo "firewall: Clients may access remote GOPHER servers"
              fi
            fi
            #
            # WAIS (210) - Accessing a remote WAIS server as a client
            #
            if [ $WAIS_CLIENT -gt 0 ]; then
              iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP \
            --sport $UNPRIVPORTS --dport 210 -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT
              iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP -m state --state ESTABLISHED,RELATED \
            --sport 210 --dport $UNPRIVPORTS -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT
              if [ $MASQUERADING -gt 0 ]; then
                iptables -A FORWARD -p TCP -s $INTERNAL_NETWORK --sport $UNPRIVPORTS --dport 210 -j
ACCEPT
                iptables -A FORWARD -m state --state ESTABLISHED,RELATED -p TCP -d $INTERNAL_NETWORK \
                        --sport 210 --dport $UNPRIVPORTS -j ACCEPT
              fi
              if [ $VERBOSE -gt 0 ]; then
                        echo "firewall: Clients may access remote WAIS servers"
              fi
            fi
            #
    # Real Video (554) - Real Video Client
    #
    if [ $RV_CLIENT -gt 0 ]; then
      iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP \
        --sport $UNPRIVPORTS --dport 554 -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT
      iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP \
        --sport 554 -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT
              if [ $MASQUERADING -gt 0 ]; then
                iptables -A FORWARD -p TCP -s $INTERNAL_NETWORK --sport $UNPRIVPORTS --dport 554 -j
ACCEPT
                iptables -A FORWARD -m state --state ESTABLISHED,RELATED -p TCP -d $INTERNAL_NETWORK \
                        --sport 554 --dport $UNPRIVPORTS -j ACCEPT
              fi
      if [ $VERBOSE -gt 0 ]; then
        echo "firewall: Real Video client enabled"
      fi
    fi
    #
    # PPTP (1723) - Accessing PPTP servers as a client
    #
    if [ $PPTP_CLIENT -gt 0 ]; then
      iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP \
        --sport $UNPRIVPORTS --dport 1723 \
        -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT
      iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP  \
        --sport 1723 --dport $UNPRIVPORTS \
        -s $ANYWHERE -d $EXTERNAL_IP \
        -m state --state ESTABLISHED,RELATED -j ACCEPT
      iptables -A INPUT -i $EXTERNAL_INTERFACE -p 47 -j ACCEPT
      iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p 47 -j ACCEPT
      if [ $MASQUERADING -gt 0 ]; then
        iptables -A INPUT -i $INTERNAL_INTERFACE -p 47 -j ACCEPT
        iptables -A OUTPUT -o $INTERNAL_INTERFACE -p 47 -j ACCEPT
        iptables -A FORWARD -p TCP -s $INTERNAL_NETWORK \
          --sport $UNPRIVPORTS --dport 1723 -j ACCEPT
        iptables -A FORWARD -p TCP -d $INTERNAL_NETWORK \
          -m state --state ESTABLISHED,RELATED \
          --sport 1723 --dport $UNPRIVPORTS -j ACCEPT
        iptables -A FORWARD -p 47 -s $INTERNAL_NETWORK -j ACCEPT
        iptables -A FORWARD -p 47 -d $INTERNAL_NETWORK -j ACCEPT
      fi
      if [ $VERBOSE -gt 0 ]; then
        echo "firewall: Clients may access remote PPTP servers"
      fi
    fi
```

```
                #
                # UDP - Accept only on selected ports
                #
                #
                # TRACEROUTE
                #
                # Traceroute usually uses -s 32769:65535 -d 33434:33523
                #
                if [ $OUTBOUND_TRACEROUTE -gt 0 ]; then
                   # Enable outgoing TRACEROUTE requests
                   iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p UDP \
                --sport $TRACEROUTE_SRC_PORTS --dport $TRACEROUTE_DEST_PORTS \
                -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT

                   if [ $MASQUERADING -gt 0 ]; then
                     iptables -A FORWARD -p UDP -s $INTERNAL_NETWORK --sport $TRACEROUTE_SRC_PORTS \
                              --dport $TRACEROUTE_DEST_PORTS -j ACCEPT
                     iptables -A FORWARD -p UDP -d $INTERNAL_NETWORK --sport $TRACEROUTE_DEST_PORTS \
                              --dport $TRACEROUTE_SRC_PORTS -j ACCEPT
                   fi
                   if [ $VERBOSE -gt 0 ]; then
                              echo "firewall: Outbound TRACEROUTE enabled"
                   fi
                fi
                if [ $INBOUND_TRACEROUTE -gt 0 ]; then
                   # Enable incoming TRACEROUTE query
                   iptables -A INPUT -i $EXTERNAL_INTERFACE -p UDP \
                --sport $TRACEROUTE_SRC_PORTS --dport $TRACEROUTE_DEST_PORTS \
                -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT

                   if [ $MASQUERADING -gt 0 ]; then
                     iptables -A FORWARD -p UDP -d $INTERNAL_NETWORK --sport $TRACEROUTE_SRC_PORTS \
                              --dport $TRACEROUTE_DEST_PORTS -j ACCEPT
                   fi
                   if [ $VERBOSE -gt 0 ]; then
                              echo "firewall: Inbound TRACEROUTE enabled"
                   fi
                fi
        #
        # DHCP Server
                #
        # This assumes that you're running a DHCP server on your firewall to
        # supply IP addresses to your internal network using dhcpd.  See any
        # of several DHCP HowTo sites for the actual server setup.
        #
        if [ $DHCP_SERVER -gt 0 ]; then
           iptables -A INPUT -i $INTERNAL_INTERFACE -p udp -s $BROADCAST_SRC \
        -d $BROADCAST_DEST --sport 67:68 --dport 67:68 -j ACCEPT
           iptables -A OUTPUT -o $INTERNAL_INTERFACE -p udp -s $INTERNAL_IP \
        --sport 67:68 --dport 67:68 -j ACCEPT
           iptables -A FORWARD -p udp -s $INTERNAL_NETWORK --sport 67:68 --dport 67:68 -j ACCEPT
                   iptables -A FORWARD -p udp -d $INTERNAL_NETWORK --sport 67:68 --dport 67:68 -j ACCEPT

          if [ $VERBOSE -gt 0 ]; then
              echo "firewall: DHCP Server enabled"
          fi
        fi
        #
        # NTP (123) - Accessing remote Network Time Servers
        #
        if [ $NTP_CLIENT -gt 0 ]; then
           iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p UDP \
        --sport $UNPRIVPORTS --dport 123 -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT
           iptables -A INPUT -i $EXTERNAL_INTERFACE -p UDP \
        --sport 123 --dport $UNPRIVPORTS -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT
           iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p UDP \
        --sport 123 --dport 123 -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT
           iptables -A INPUT -i $EXTERNAL_INTERFACE -p UDP \
        --sport 123 --dport 123 -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT

                   if [ $MASQUERADING -gt 0 ]; then
                     iptables -A FORWARD -p UDP -s $INTERNAL_NETWORK --sport $UNPRIVPORTS --dport 123 -j
ACCEPT
                     iptables -A FORWARD -p UDP -d $INTERNAL_NETWORK --sport 123 --dport $UNPRIVPORTS -j
ACCEPT
                     iptables -A FORWARD -p UDP -s $INTERNAL_NETWORK --sport 123 --dport 123 -j ACCEPT
```

```
                iptables -A FORWARD -p UDP -d $INTERNAL_NETWORK --sport 123 --dport 123 -j ACCEPT
              fi
        if [ $VERBOSE -gt 0 ]; then
          echo "firewall: NTP Client enabled"
        fi
      fi
              #
              # ICQ (4000) - The Miribilis ICQ Client
              #
              if [ $ICQ_CLIENT -gt 0 ]; then
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p UDP \
              --sport $UNPRIVPORTS --dport 4000 -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT
                iptables -A INPUT -i $EXTERNAL_INTERFACE -p UDP \
              --sport 4000 --dport $UNPRIVPORTS -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT
                if [ $MASQUERADING -gt 0 ]; then
                  iptables -A FORWARD -p UDP -s $INTERNAL_NETWORK --sport $UNPRIVPORTS --dport 4000 -j
ACCEPT
                  iptables -A FORWARD -p UDP -d $INTERNAL_NETWORK --sport 4000 --dport $UNPRIVPORTS -j
ACCEPT
                fi
                if [ $VERBOSE -gt 0 ]; then
                  echo "firewall: ICQ Client enabled"
                fi
              fi
              #
              # GAMES
              # Half-Life/CounterStrike
              #
              if [ $HALF_LIFE -gt 0 ]; then
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p UDP \
              --sport 27000:27050 --dport $UNPRIVPORTS -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT
                iptables -A INPUT -i $EXTERNAL_INTERFACE -p UDP \
              -m state --state RELATED,ESTABLISHED,NEW --dport 27000:27050 -s $ANYWHERE \
              -d $EXTERNAL_IP -j ACCEPT
                if [ $MASQUERADING -gt 0 ]; then
                  iptables -A FORWARD -p UDP -s $INTERNAL_NETWORK --sport 27000:27050 --dport
$UNPRIVPORTS -j ACCEPT
                  iptables -A FORWARD -p UDP -d $INTERNAL_NETWORK -m state --state
RELATED,ESTABLISHED,NEW --dport 27000:27050 -j ACCEPT
                fi
                if [ $VERBOSE -gt 0 ]; then
                  echo "firewall: Half-Life/CounterStrike game ports enabled"
                fi
              fi
              #
              # Return to Castle Wolfenstein
              #
              if [ $WOLF_CLIENT -gt 0 ]; then
                iptables  -A OUTPUT -o $EXTERNAL_INTERFACE -p UDP \
              --sport $UNPRIVPORTS --dport 27950:27965 -s $EXTERNAL_IP -d $ANYWHERE -j ACCEPT
                iptables  -A INPUT -i $EXTERNAL_INTERFACE -p UDP \
              --sport 27950:27965 --dport $UNPRIVPORTS -s $ANYWHERE -d $EXTERNAL_IP -j ACCEPT
                if [ $MASQUERADING -gt 0 ]; then
                  iptables -A FORWARD -p UDP -s $INTERNAL_NETWORK --sport $UNPRIVPORTS --dport
27950:27965 -j ACCEPT
                  iptables -A FORWARD -p UDP -d $INTERNAL_NETWORK --sport 27950:27965 --dport
$UNPRIVPORTS -j ACCEPT
                fi
                if [ $VERBOSE -gt 0 ]; then
                  echo "firewall: Castle Wolfenstein game ports enabled"
                fi
              fi
              # --------------------------------------------------------------
              #
              # Spoofing and Bad Addresses
              #
              # Refuse spoofed packets.
              # Ignore blatantly illegal source addresses.
              # Protect yourself from sending to bad addresses.
              # Refuse spoofed packets pretending to be from
              # the external interface's IP address.
              iptables -A INPUT -i $EXTERNAL_INTERFACE -s $EXTERNAL_IP -j LnD
              # Refuse packets claiming to be to or from a Class-A private network.
              iptables -A INPUT -i $EXTERNAL_INTERFACE -s $CLASS_A -j LnD
              iptables -A INPUT -i $EXTERNAL_INTERFACE -d $CLASS_A -j LnD
              iptables -A OUTPUT -o $EXTERNAL_INTERFACE -s $CLASS_A -j LnD
```

```
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -d $CLASS_A -j LnD
                # Refuse packets claiming to be to or from a Class-B private network.
                iptables -A INPUT -i $EXTERNAL_INTERFACE -s $CLASS_B -j LnD
                iptables -A INPUT -i $EXTERNAL_INTERFACE -d $CLASS_B -j LnD
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -s $CLASS_B -j LnD
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -d $CLASS_B -j LnD
                # Refuse packets claiming to be to or from a Class-C private network.
                iptables -A INPUT -i $EXTERNAL_INTERFACE -s $CLASS_C -j LnD
                iptables -A INPUT -i $EXTERNAL_INTERFACE -d $CLASS_C -j LnD
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -s $CLASS_C -j LnD
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -d $CLASS_C -j LnD
                # Refuse packets claiming to be from the loopback.
                iptables -A INPUT -i $EXTERNAL_INTERFACE -s $LOOPBACK_NETWORK -j LnD
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -s $LOOPBACK_NETWORK -j LnD
                # Refuse malformed broadcast packets.
                iptables -A INPUT -i $EXTERNAL_INTERFACE -s $BROADCAST_DEST -j LnD
                iptables -A INPUT -i $EXTERNAL_INTERFACE -d $BROADCAST_SRC  -j LnD
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -s $BROADCAST_DEST -j LnD
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -d $BROADCAST_SRC  -j LnD
                # Refuse Class-D Multicast addresses.
                # Multicast is only illegal as a source address.
                # Multicast uses UDP.
                iptables -A INPUT -i $EXTERNAL_INTERFACE -s $CLASS_D_MULTICAST -j LnD
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -s $CLASS_D_MULTICAST -j LnR
                # Refuse Class-E reserved IP addresses.
                iptables -A INPUT -i $EXTERNAL_INTERFACE -s $CLASS_E_RESERVED_NET -j LnD
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -d $CLASS_E_RESERVED_NET -j LnR
                # ----------------------------------------------------------------
                #
                # DROP (on input), REJECT (output) and LOG anything else on the external (red) interface
                #
                iptables -A INPUT -i $EXTERNAL_INTERFACE -p TCP \
                   -s $ANYWHERE -j LnD
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p TCP \
                   -s $ANYWHERE -j LnR
                iptables -A INPUT -i $EXTERNAL_INTERFACE -p UDP \
                   -s $ANYWHERE -j LnD
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p UDP \
                   -s $ANYWHERE -j LnR
                iptables -A INPUT -i $EXTERNAL_INTERFACE -p ICMP \
                   -s $ANYWHERE -j LnD
                iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p ICMP \
                   -s $ANYWHERE -j LnR
                #
                # Masquerade internal traffic
                #
                if [ $MASQUERADING -gt 0 ]; then
                   # All internal traffic is masqueraded externally
                   iptables -t nat -A POSTROUTING -s $INTERNAL_NETWORK -o $EXTERNAL_INTERFACE -j SNAT \
                   --to $EXTERNAL_IP
        # Note: some may find this works better on machines with non-static
                   # external IP addresses:
        # iptables -t nat -A POSTROUTING -o ethX -j MASQUERADE
                   # Enable IP Forwarding
                   echo 1 >/proc/sys/net/ipv4/ip_forward
                   #
                   # Unlimited traffic within the local network
                   #
                   # All internal machines have access to the firewall machine
                   iptables -A INPUT -i $INTERNAL_INTERFACE -s $INTERNAL_NETWORK -j ACCEPT
                   iptables -A OUTPUT -o $INTERNAL_INTERFACE -d $INTERNAL_NETWORK -j ACCEPT
        if [ $VERBOSE -gt 0 ]; then
                                echo "firewall: Masquerading internal network"
                   fi
                fi
                # ----------------------------------------------------------------
                # Zero counts
                iptables -Z
                # ----------------------------------------------------------------
                echo "done"
                touch /var/lock/subsys/firewall
                echo
                ;;
status)
                if [ -f /var/lock/subsys/firewall ]; then
                   echo "Firewall started and configured"
```

```
                else
                   echo "Firewall stopped"
                fi
                exit 0
                ;;
restart|reload)
                $0 stop
                $0 start
                ;;
stop)
                echo "Shutting down Firewall services"
                # Turn off IP Forwarding
                echo 0 >/proc/sys/net/ipv4/ip_forward
                # Turn off dynamic IP hacking
                echo 0 > /proc/sys/net/ipv4/ip_dynaddr
                # Flush the rule chains
                iptables -F
                # Delete custom chains
                iptables -X
                # Zero counts
                iptables -Z
                # Set the default policy to DROP
                iptables -P INPUT DROP
                iptables -P OUTPUT DROP
                iptables -P FORWARD DROP
                # Allow unlimited traffic on the loopback interface
                iptables -A INPUT -i $LOOPBACK_INTERFACE -j ACCEPT
                iptables -A OUTPUT -o $LOOPBACK_INTERFACE -j ACCEPT
                # Open the configuration file
                if [ -f /etc/firewall/firewall.conf.iptables ]; then
                   . /etc/firewall/firewall.conf.iptables
        if [ $MASQUERADING -gt 0 ]; then
           # Allow unlimited local traffic on the internal interface
           iptables -A INPUT -i $INTERNAL_INTERFACE -j ACCEPT
           iptables -A OUTPUT -o $INTERNAL_INTERFACE -j ACCEPT
                   fi
                else
                   echo "firewall: No configuration file found at /etc/firewall/firewall.conf.iptables"
                   exit 1
                fi
                rm -f /var/lock/subsys/firewall
                echo
                ;;
*)
                echo "Usage: /etc/rc.d/init.d/firewall.iptables {start|stop|status|restart|reload}"
                exit 1
esac
exit 0
```

APPENDIX F2

Dynamic Firewall Script

```
#!/bin/bash

source /usr/local/share/dynfw.sh

args 2 $# "${0} IPADDR {on/off}" "Drops packets to/from IPADDR. Good for obnoxious
                        networks/hosts/DoS"

if [ "$2" == "on" ]
then
  #rules will be appended or inserted as normal
 APPEND="-A"
  INSERT="-I"
  rec_check ipdrop $1 "$1 already blocked" on
  record ipdrop $1
elif [ "$2" == "off" ]
then
  #rules will be deleted instead
 APPEND="-D"
  INSERT="-D"
  rec_check ipdrop $1 "$1 not currently blocked" off
  unrecord ipdrop $1
else
  echo "Error: \"off\" or \"on\" expected as second argument"
  exit 1
fi

#block outside IP address that's causing problems
#attacker's incoming TCP connections will take a minute or so to time out,
#reducing DoS effectiveness.

iptables $INSERT INPUT   -s $1 -j DROP
iptables $INSERT OUTPUT  -d $1 -j DROP
iptables $INSERT FORWARD -d $1 -j DROP
iptables $INSERT FORWARD -s $1 -j DROP

echo "IP ${1} drop ${2}."
```

APPENDIX F3

Calculating Parameters for No Firewall Configuration

$f_{initialization} : \forall c(n,m) = 1$

$f_{runtime} : \forall c(n,m) = 1$

F3.1 Probability of Available Services

$$\begin{aligned} P(AS-NF) &= P(c(n,m)=1 \mid f_{initialization}) \wedge P(c(n,m)=1 \mid f_{runtime}) \\ &= P(c(n,m)=1 \mid \forall c(n,m)=1) \wedge P(c(n,m)=1 \mid \forall c(n,m)=1) \\ &= 1 \wedge 1 \\ &= 1 \end{aligned}$$

F3.2 Probability of Expose Line

$P(AS-NF) = 1$

- For $t < t_r$ or $t > t_{ft}$, $P(EL-NF) = 1$
- For $th(n) > 0$, $P(EL-NF) = 1$
- For $protection(n) < risk(m)$, $P(EL-NF) = 1$

F3.3 Probability of Denial of Service

$$\begin{aligned} P(DS-NF) &= P(c(n,m)=0 \mid f_{initialization}) \vee P(c(n,m)=0 \mid f_{runtime}) \\ &= P(c(n,m)=1 \mid \forall c(n,m)=1) \vee P(c(n,m)=1 \mid \forall c(n,m)=1) \\ &= 0 \vee 0 \\ &= 0 \end{aligned}$$

APPENDIX F4

Calculating Parameters for Static Firewall Configuration

$f_{initialization} : \exists c(n,m) = 1 + \exists c(n,m) = 0$

$f_{runtime} : \exists c(n,m) = 1 + \exists c(n,m) = 0$

F4.1 Probability of Available Services

$$P(AS-SF) = P(c(n,m) = 1 \mid f_{initialization}) \wedge P(c(n,m) = 1 \mid f_{runtime})$$
$$= P(c(n,m) = 1 \mid \exists c(n,m) = 1 + \exists c(n,m) = 0) \wedge P(c(n,m) = 1 \mid \exists c(n,m) = 1 + \exists c(n,m) = 0)$$
$$= P(c(n,m) = 1 \mid \exists c(n,m) = 1 + \exists c(n,m) = 0)$$

F4.2 Probability of Expose Line

$$P(AS-SF) = P(c(n,m) = 1 \mid \exists c(n,m) = 1 + \exists c(n,m) = 0)$$

- For $t < t_r$ or $t > t_{ft}$, $P(EL-SF) = P(c(n,m) = 1 \mid \exists c(n,m) = 1 + \exists c(n,m) = 0)$
- For $th(n) > 0$, $P(EL-SF) = P(c(n,m) = 1 \mid \exists c(n,m) = 1 + \exists c(n,m) = 0)$
- For $protection(n) < risk(m)$,
  $P(EL-SF) = P(c(n,m) = 1 \mid \exists c(n,m) = 1 + \exists c(n,m) = 0)$

F4.3 Probability of Denial of Service

$$P(DS-SF) = P(c(n,m) = 0 \mid f_{initialization}) \vee P(c(n,m) = 0 \mid f_{runtime})$$
$$= P(c(n,m) = 0 \mid \exists c(n,m) = 1 + \exists c(n,m) = 0) \vee P(c(n,m) = 0 \mid \exists c(n,m) = 1 + \exists c(n,m) = 0)$$
$$= P(c(n,m) = 0 \mid \exists c(n,m) = 1 + \exists c(n,m) = 0)$$

APPENDIX F5

Calculating Parameters for Dynamic Firewall Configuration

$f_{initialization} : \exists c(n,m) = 1 + \exists c(n,m) = 0$

$f_{runtime} : \exists c(n,m) = 0 \Leftrightarrow th(n) > 0$

F5.1 Probability of Available Services

$$P(AS-DF) = P(c(n,m) = 1 \mid f_{initialization}) \wedge P(c(n,m) = 1 \mid f_{runtime})$$
$$= P(c(n,m) = 1 \mid \exists c(n,m) = 1 + \exists c(n,m) = 0) \wedge P(c(n,m) = 1 \mid \exists c(n,m) = 0 \Leftrightarrow th(n) > 0)$$
$$= P(c(n,m) = 1 \mid \exists c(n,m) = 1 + \exists c(n,m) = 0) \wedge 1$$
$$= P(c(n,m) = 1 \mid \exists c(n,m) = 1 + \exists c(n,m) = 0)$$

F5.2 Probability of Expose Line

$$P(AS-DF) = P(c(n,m) = 1 \mid \exists c(n,m) = 1 + \exists c(n,m) = 0) \wedge P(c(n,m) = 1 \mid \exists c(n,m) = 0 \Leftrightarrow th(n) > 0)$$

- For $t < t_r$ or $t > t_{ft}$, $P(EL-DF) = P(c(n,m) = 1 \mid \exists c(n,m) = 1 + \exists c(n,m) = 0)$
- For $th(n) > 0$, $P(EL-DF) = 0$
- For $protection(n) < risk(m)$,
  $P(EL-DF) = P(c(n,m) = 1 \mid \exists c(n,m) = 1 + \exists c(n,m) = 0)$

F5.3 Probability of Denial of Service

$$P(DS-DF) = P(c(n,m) = 0 \mid f_{initialization}) \vee P(c(n,m) = 0 \mid f_{runtime})$$
$$= P(c(n,m) = 0 \mid \exists c(n,m) = 1 + \exists c(n,m) = 0) \vee P(c(n,m) = 0 \mid \exists c(n,m) = 0 \Leftrightarrow th(n) > 0)$$
$$= P(c(n,m) = 0 \mid \exists c(n,m) = 1 + \exists c(n,m) = 0) \vee 0$$
$$= P(c(n,m) = 0 \mid \exists c(n,m) = 1 + \exists c(n,m) = 0)$$

PUBLICATIONS

1. Crysdian, C., and Abdul Hanan bin Abdullah. (2005). *Network Firewall with Multilevel Security Policy*. Accepted, to appear in Journal of Information Technology in Asia.

2. Crysdian, C., and Abdul Hanan bin Abdullah. (2005). *Lattice-Based Firewall for Safety Internet Access*. In the FSKSM Postgraduate Annual Research Seminar (PARS'05). 17-18 May 2005. Johor, Malaysia.

3. Crysdian, C., and Abdul Hanan bin Abdullah. (2004). *Network Firewall Development with Multilevel Security Policy*. 14th International Conference on Computer Theory and Applications, Egypt, Alexandria.

4. Crysdian, C., and Abdul Hanan bin Abdullah. (2003). *Access Controlling Firewall using Bell-LaPadula Security Model*. The Quality in Research Seminar 2003 (QIR 2003), Jakarta, Indonesia, October 2003.

5. Crysdian, C., Abdul Hanan bin Abdullah, and Mohd. Aizaini bin Maarof. (2003). *Enforcing Multilevel Security Policies for Network Firewall*. The 3rd International Conference on Information Technology in Asia (CITA'03), Kuching, Malaysia, July 2003.

6. Crysdian, C., and Abdul Hanan bin Abdullah. (2003). *Survey on Access Control* Method. ASITT03, Johor Bahru, 2003.

7. Crysdian, C., and Abdul Hanan bin Abdullah. (2003). *Authentication Protocol: Methods Review.* The 2002 International Technical Conference on

Circuit/Systems, Computers and Communications (ITC-CSCC), Phuket, Thailand, July 2002.

PUBLICATIONS

1. Crysdian, C., and Abdul Hanan bin Abdullah. (2005). *Network Firewall with Multilevel Security Policy*. Accepted, to appear in Journal of Information Technology in Asia.

2. Crysdian, C., and Abdul Hanan bin Abdullah. (2005). *Lattice-Based Firewall for Safety Internet Access*. In the FSKSM Postgraduate Annual Research Seminar (PARS'05). 17-18 May 2005. Johor, Malaysia.

3. Crysdian, C., and Abdul Hanan bin Abdullah. (2004). *Network Firewall Development with Multilevel Security Policy*. 14th International Conference on Computer Theory and Applications, Egypt, Alexandria.

4. Crysdian, C., and Abdul Hanan bin Abdullah. (2003). *Access Controlling Firewall using Bell-LaPadula Security Model*. The Quality in Research Seminar 2003 (QIR 2003), Jakarta, Indonesia, October 2003.

5. Crysdian, C., Abdul Hanan bin Abdullah, and Mohd. Aizaini bin Maarof. (2003). *Enforcing Multilevel Security Policies for Network Firewall*. The 3rd International Conference on Information Technology in Asia (CITA'03), Kuching, Malaysia, July 2003.

6. Crysdian, C., and Abdul Hanan bin Abdullah. (2003). *Survey on Access Control* Method. ASITT03, Johor Bahru, 2003.

7. Crysdian, C., and Abdul Hanan bin Abdullah. (2003). *Authentication Protocol: Methods Review.* The 2002 International Technical Conference on

Circuit/Systems, Computers and Communications (ITC-CSCC), Phuket, Thailand, July 2002.