AN ENHANCED PERFORMANCE MODEL FOR METAMORPHIC COMPUTER VIRUS CLASSIFICATION AND DETECTION

BABAK BASHARIRAD

A thesis submitted in fulfilment of the requirements for the award of the degree of Doctor of Philosophy (Computer Science)

Faculty of Computing
Universiti Teknologi Malaysia

OCTOBER 2013
To My MOTHER, who supported me as a firm mountain overall duration of my life,
   Thanks for your cares and inspirations, Mom

To My FATHER, for his love, cares and encouragement

With Special Thanks

To My BELOVED WIFE, NAHID, who accommodated me in all circumstances, helped
   me to grow spiritually and intellectually, Thanks for your love and patience

To My WONDERFUL SON, MANI, who is all my life, His love inspired me to work hard
   and finish this work
ACKNOWLEDGEMENT

I would like to express my deepest gratitude to my dear supervisor Assoc. Prof. Dr. Maslin Masrom for her valuable guidance, patience, and encouragements. This research would never have been completed without her advices and supports. She always was ready to spend her time to counsel me, and to improve the thesis report, patiently and cordially.

My special thanks to my co-supervisor Assoc. Prof. Dr. Suhaimi Ibrahim for his supports, suggestions and valuable comments.
ABSTRACT

Metamorphic computer virus employs various code mutation techniques to change its code to become new generations. These generations have similar behavior and functionality and yet, they could not be detected by most commercial antivirus because their solutions depend on a signature database and make use of string signature-based detection methods. However, the antivirus detection engine can be avoided by metamorphism techniques. The purpose of this study is to develop a performance model based on computer virus classification and detection. The model would also be able to examine portable executable files that would classify and detect metamorphic computer viruses. A Hidden Markov Model implemented on portable executable files was employed to classify and detect the metamorphic viruses. This proposed model that produce common virus statistical patterns was evaluated by comparing the results with previous related works and famous commercial antiviruses. This was done by investigating the metamorphic computer viruses and their features, and the existing classifications and detection methods. Specifically, this model was applied on binary format of portable executable files and it was able to classify if the files belonged to a virus family. Besides that, the performance of the model, practically implemented and tested, was also evaluated based on detection rate and overall accuracy. The findings indicated that the proposed model is able to classify and detect the metamorphic virus variants in portable executable file format with a high average of 99.7% detection rate. The implementation of the model is proven useful and applicable for antivirus programs.
ABSTRAK

# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>CHAPTER</th>
<th>TITLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DECLARATION</td>
<td></td>
<td>ii</td>
</tr>
<tr>
<td>DEDICATION</td>
<td></td>
<td>iii</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENT</td>
<td></td>
<td>iv</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td></td>
<td>v</td>
</tr>
<tr>
<td>ABSTRAK</td>
<td></td>
<td>vi</td>
</tr>
<tr>
<td>TABLE OF CONTENTS</td>
<td></td>
<td>vii</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td></td>
<td>xii</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td></td>
<td>xvii</td>
</tr>
<tr>
<td>LIST OF ABBREVIATIONS</td>
<td></td>
<td>xxiv</td>
</tr>
<tr>
<td>LIST OF APPENDICES</td>
<td></td>
<td>xxv</td>
</tr>
</tbody>
</table>

## 1 INTRODUCTION

1.1 Introduction | 1 |
1.2 Background of the Study | 4 |
1.3 Problem Statement | 7 |
1.4 Purpose of the Study | 8 |
1.5 Objectives of the Study | 8 |
1.6 Significance of the Study | 8 |
1.7 Scope of the Study | 9 |
1.8 Structure of the Thesis | 9 |

## 2 LITERATURE REVIEW

2.1 Introduction | 11 |
2.2 Evolution of Viruses According to Concealment Strategy | 12 |
2.2.1 Encrypted Virus | 16 |
2.2.1.1 Encryption Methods | 19 |
2.2.2 Oligomorphic Virus
   2.2.2.1 A Case Review: Win95.Memorial

2.2.3 Polymorphic Virus
   2.2.3.1 Polymorphic Mechanism
   2.2.3.2 A Case Study on DOS Polymorphic Virus: The 1260
   2.2.3.3 A 32-bit Polymorphic Virus: Win95.Marburg

2.2.4 Metamorphic Virus
   2.2.4.1 An Overview of Techniques and Mechanism
   2.2.4.2 A Simple Metamorphic Virus

2.3 Code Obfuscation in Metamorphic Viruses
   2.3.1 Garbage Code Insertion
   2.3.2 Register or Variable Usage Exchange
   2.3.3 Instruction Replacement
   2.3.4 Instruction Permutation
   2.3.5 Code Transposition

2.4 Detection Techniques for Metamorphic Viruses
   2.4.1 Geometric Detection
   2.4.2 Wildcard String and Half-Byte Scanning
   2.4.3 Code Disassembling
   2.4.4 Code Emulation
      2.4.4.1 Searching Patterns using Negative or Positive Features
      2.4.4.2 Emulator-Based Heuristics
      2.4.4.3 Dummy Loops Detection
      2.4.4.4 Stack Decryption Detection
   2.4.5 Code Transformation Detection
   2.4.6 Subroutine De-permutation
   2.4.7 Regular Expressions and DFA

2.5 Static vs. Dynamic Virus Detection

2.6 Classification
   2.6.1 Supervised vs. Unsupervised Learning
   2.6.2 Sequence Classification
   2.6.3 Binary Classification
4 EXPERIMENTAL DESIGN

4.1 Introduction

4.2 Experimental Setup

  4.2.1 Testbed
  4.2.2 Hardware
  4.2.3 Software

4.3 Feature Extraction by Developing PE Analyzer Module

  4.3.1 Format of Portable Executable Files
    4.3.1.1 MS DOS Header
    4.3.1.2 PE Header
    4.3.1.3 Optional Header
    4.3.1.4 Section Header
    4.3.1.5 Sections

  4.3.2 Code Segment Extractor

  4.3.3 Opcode Sequence Generator

4.4 Data Set

4.5 Hidden Markov Model Training

  4.5.1 Underflow problem and Scaling the Algorithm

4.6 Classification Test Implementation

  4.6.1 Threshold Criteria

4.7 Summary

5 RESULTS AND ANALYSIS

5.1 Introduction

5.2 Results

5.3 Impact of Threshold on False Alarms

5.4 Performance Measurements

5.5 Threshold

5.6 Evaluation and Justification

  5.6.1 Comparison of the Performance with the Previous Works
    5.6.1.1 Comparison of all 20 Models Performances
    5.6.1.2 Comparison of Average Measures
    5.6.2 Justification by Commercial Antivirus Scanners

5.7 Summary
<table>
<thead>
<tr>
<th>6</th>
<th>DISCUSSION AND CONCLUSIONS</th>
<th>174</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>Introduction</td>
<td>174</td>
</tr>
<tr>
<td>6.2</td>
<td>Discussion and Conclusions</td>
<td>175</td>
</tr>
<tr>
<td>6.3</td>
<td>Contribution and Importance of the Study</td>
<td>177</td>
</tr>
<tr>
<td>6.4</td>
<td>Future Recommendations</td>
<td>179</td>
</tr>
</tbody>
</table>

REFERENCES

Appendices A – F 199 – 276
# LIST OF TABLES

<table>
<thead>
<tr>
<th>TABLE NO.</th>
<th>TITLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Timeline of Malware Evolution History – Chronology of Events</td>
<td>14</td>
</tr>
<tr>
<td>2.2</td>
<td>Examples for Static Encryption Key (Aycock, 2006)</td>
<td>19</td>
</tr>
<tr>
<td>2.3</td>
<td>Examples for Static Encryption Key</td>
<td>20</td>
</tr>
<tr>
<td>2.4</td>
<td>An Example of Decryptor for Win95.Memorial (Szor, 2005)</td>
<td>22</td>
</tr>
<tr>
<td>2.5</td>
<td>A Slightly Different Decryptor of Win95.Memorial (Szor, 2005)</td>
<td>23</td>
</tr>
<tr>
<td>2.6</td>
<td>Decryptor of 1260 (Szor, 2005)</td>
<td>26</td>
</tr>
<tr>
<td>2.7</td>
<td>An Instance of the Decryptor of Win95.Marburg (Szor, 2005)</td>
<td>28</td>
</tr>
<tr>
<td>2.8</td>
<td>Win95.Regswap Exploits Dissimilar Registers in Various Copies (Szor and Ferrie, 2001a)</td>
<td>35</td>
</tr>
<tr>
<td>2.9</td>
<td>Examples of Dead Codes (Borello and Me, 2008)</td>
<td>37</td>
</tr>
<tr>
<td>2.10</td>
<td>Examples of Dead Codes (Rad and Masrom, 2010)</td>
<td>38</td>
</tr>
<tr>
<td>2.11</td>
<td>Version 1 of Win32.Evol (Rad and Masrom, 2010)</td>
<td>38</td>
</tr>
<tr>
<td>2.12</td>
<td>Version 2 of Win32.Evol (junk insertion) (Rad and Masrom, 2010)</td>
<td>39</td>
</tr>
<tr>
<td>2.13</td>
<td>Version 1 of Win95.Regswap (Szor, 2000; Szor and Ferrie, 2001a)</td>
<td>40</td>
</tr>
<tr>
<td>2.14</td>
<td>Version 2 of Win95.Regswap (Szor, 2000; Szor and Ferrie, 2001a)</td>
<td>40</td>
</tr>
<tr>
<td>2.15</td>
<td>Version 1 of Win95.Bistro (Szor, 2005)</td>
<td>41</td>
</tr>
<tr>
<td>2.16</td>
<td>Version 2 of Win95.Bistro (Szor, 2005)</td>
<td>42</td>
</tr>
<tr>
<td>2.17</td>
<td>Examples of Code Permutation</td>
<td>43</td>
</tr>
</tbody>
</table>
2.18 Win95.Regswap Exploits Different Registers in Various Copies (Finones and Fernandez, 2006)

2.19 Sample Detection of Win95.Puron (Szor and Ferrie, 2001a)

2.20 A Sample Version of ACG Metamorphic Virus (Szor and Ferrie, 2001a)

2.21 Some Examples of Instructions Transformation Mapping by Simile (Finones and Fernandez, 2006)

2.22 Two Different Versions of Simile (Finones and Fernandez, 2006)

2.23 Process of Rebuilding a Permutated Virus Body (Finones and Fernandez, 2006)

2.24 Summary of More Related Works Previously Done

2.25 Critical Comparison of Previous Related Works

4.1 COFF Header (Microsoft Corporation, 2013)

4.2 Section Header (Microsoft Corporation, 2013)

5.1 LLPO Values of Data set # 1, with 2 States

5.2 LLPO values of Data Set # 1, with 3 States

5.3 Comparison of the Minimum LLPO for the NGVCK Family Virus Samples and Maximum LLPO of the other Benign and Non-Family Virus Files

5.4 Number of FPs and FNs, for Data Test Set #2, with N=3

5.5 Detection Rate (DR), False Positive Rate (FPR), and Overall Accuracy (OA) for Different Thresholds in the Range of [-4.2, -2.4], Test Set #2, N=3

5.6 Threshold Values to Obtain a Detection Rate Higher than 90%

5.7 Measure Values for all Models at Threshold = -3.8

5.8 Comparison of the Measures of Our Results with the Results of Wong (2006) and Govindaraj (2008)

5.9 Comparison of the average measures of four recent related works

5.10 Results of Commercial Antivirus Scanners for some Randomly Chosen NGVCK Virus Variants
5.11 The Complete Results of Commercial Antivirus Scanners for 40 NGVCK Virus Variants in the Data Test # 1

5.12 The Result of the Proposed HMM-based Detector with a Threshold = -3.8

A.1 LLPO Values of Data Set # 1, with 4 States
A.2 LLPO Values of Data Set # 1, with 5 States
A.3 LLPO Values of Data Set # 2, with 2 States
A.4 LLPO Values of Data Set # 2, with 3 States
A.5 LLPO Values of Data Set # 2, with 4 States
A.6 LLPO Values of Data Set # 2, with 5 States
A.7 LLPO Values of Data Set # 3, with 2 States
A.8 LLPO Values of Data Set # 3, with 3 States
A.9 LLPO Values of Data Set # 3, with 4 States
A.10 LLPO Values of Data Set # 3, with 5 States
A.11 LLPO Values of Data Set # 4, with 2 States
A.12 LLPO Values of Data Set # 4, with 3 States
A.13 LLPO Values of Data Set # 4, with 4 States
A.14 LLPO Values of Data Set # 4, with 5 States
A.15 LLPO Values of Data Set # 5, with 2 States
A.16 LLPO Values of Data Set # 5, with 3 States
A.17 LLPO Values of Data Set # 5, with 4 States
A.18 LLPO Values of Data Set # 5, with 5 States

B.1 Number of FPs and FNs, for Data Test Set #1, with N=2
B.2 Number of FPs and FNs, for Data Test Set #1, with N=3
B.3 Number of FPs and FNs, for Data Test Set #1, with N=4
B.4 Number of FPs and FNs, for Data Test Set #1, with N=5
B.5 Number of FPs and FNs, for Data Test Set #2, with N=2
B.6 Number of FPs and FNs, for Data Test Set #2, with N=3
B.7 Number of FPs and FNs, for Data Test Set #2, with N=4

B.8 Number of FPs and FNs, for Data Test Set #2, with N=5

B.9 Number of FPs and FNs, for Data Test Set #3, with N=2

B.10 Number of FPs and FNs, for Data Test Set #3, with N=3

B.11 Number of FPs and FNs, for Data Test Set #3, with N=4

B.12 Number of FPs and FNs, for Data Test Set #3, with N=5

B.13 Number of FPs and FNs, for Data Test Set #4, with N=2

B.14 Number of FPs and FNs, for Data Test Set #4, with N=3

B.15 Number of FPs and FNs, for Data Test Set #4, with N=4

B.16 Number of FPs and FNs, for Data Test Set #4, with N=5

B.17 Number of FPs and FNs, for Data Test Set #5, with N=2

B.18 Number of FPs and FNs, for Data Test Set #5, with N=3

B.19 Number of FPs and FNs, for Data Test Set #5, with N=4

B.20 Number of FPs and FNs, for Data Test Set #5, with N=5

C.1 DR, FPR, and OA for Different Thresholds in the Range of [-4.2, -2.4], Test Set #1, N=2

C.2 DR, FPR, and OA for Different Thresholds in the Range of [-4.2, -2.4], Test Set #1, N=3

C.3 DR, FPR, and OA for Different Thresholds in the Range of [-4.2, -2.4], Test Set #1, N=4

C.4 DR, FPR, and OA for Different Thresholds in the Range of [-4.2, -2.4], Test Set #1, N=5

C.5 DR, FPR, and OA for Different Thresholds in the Range of [-4.2, -2.4], Test Set #2, N=2

C.6 DR, FPR, and OA for Different Thresholds in the Range of [-4.2, -2.4], Test Set #2, N=3

C.7 DR, FPR, and OA for Different Thresholds in the Range of [-4.2, -2.4], Test Set #2, N=4

C.8 DR, FPR, and OA for Different Thresholds in the Range of [-4.2, -2.4], Test Set #2, N=5
| C.9 | DR, FPR, and OA for Different Thresholds in the Range of [-4.2, -2.4], Test Set #3, N=2 | 248 |
| C.10 | DR, FPR, and OA for Different Thresholds in the Range of [-4.2, -2.4], Test Set #3, N=3 | 249 |
| C.11 | DR, FPR, and OA for Different Thresholds in the Range of [-4.2, -2.4], Test Set #3, N=4 | 250 |
| C.12 | DR, FPR, and OA for Different Thresholds in the Range of [-4.2, -2.4], Test Set #3, N=5 | 251 |
| C.13 | DR, FPR, and OA for Different Thresholds in the Range of [-4.2, -2.4], Test Set #4, N=2 | 252 |
| C.14 | DR, FPR, and OA for Different Thresholds in the Range of [-4.2, -2.4], Test Set #4, N=3 | 253 |
| C.15 | DR, FPR, and OA for Different Thresholds in the Range of [-4.2, -2.4], Test Set #4, N=4 | 254 |
| C.16 | DR, FPR, and OA for Different Thresholds in the Range of [-4.2, -2.4], Test Set #4, N=5 | 255 |
| C.17 | DR, FPR, and OA for Different Thresholds in the Range of [-4.2, -2.4], Test Set #5, N=2 | 256 |
| C.18 | DR, FPR, and OA for Different Thresholds in the Range of [-4.2, -2.4], Test Set #5, N=3 | 257 |
| C.19 | DR, FPR, and OA for Different Thresholds in the Range of [-4.2, -2.4], Test Set #5, N=4 | 258 |
| C.20 | DR, FPR, and OA for Different Thresholds in the Range of [-4.2, -2.4], Test Set #5, N=5 | 259 |
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>FIGURE NO.</th>
<th>TITLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Storage and Sharing of Information in Modern Digital World</td>
<td>1</td>
</tr>
<tr>
<td>1.2</td>
<td>Symantec Corp. Status of Enterprise Security in 2010: “Most Significant Risks and Cyber Attacks in Last Year” (Symantec, 2010)</td>
<td>2</td>
</tr>
<tr>
<td>2.1</td>
<td>Encrypted Virus (a General Format) (Aycock, 2006)</td>
<td>17</td>
</tr>
<tr>
<td>2.2</td>
<td>Oligomorphic Virus Structure</td>
<td>21</td>
</tr>
<tr>
<td>2.3</td>
<td>Polymorphic Virus Structure</td>
<td>25</td>
</tr>
<tr>
<td>2.4</td>
<td>Metamorphic Virus Propagation</td>
<td>29</td>
</tr>
<tr>
<td>2.5</td>
<td>Stages of Program Transformation in a Mutation Engine (Lakhotia et al., 2004)</td>
<td>31</td>
</tr>
<tr>
<td>2.6</td>
<td>Structure of a Metamorphic Engine (Walenstein et al., 2007)</td>
<td>32</td>
</tr>
<tr>
<td>2.7</td>
<td>Production of a Metamorphic Binary From Sources of Original Program and Sources of Metamorphic Engine (Borello et al., 2009)</td>
<td>35</td>
</tr>
<tr>
<td>2.8</td>
<td>Example of Code Transposition in Different Generations (Szor and Ferrie, 2001a)</td>
<td>43</td>
</tr>
<tr>
<td>2.9</td>
<td>Mutation Successive Steps Executed by <em>Win32.Similie</em></td>
<td>52</td>
</tr>
<tr>
<td>2.10</td>
<td>Subroutine De-permutation (Finones and Fernandez, 2006)</td>
<td>56</td>
</tr>
<tr>
<td>2.11</td>
<td>DFA Building Process (Finones and Fernandez, 2006)</td>
<td>59</td>
</tr>
<tr>
<td>2.12</td>
<td>DFA Simulation Process (Finones and Fernandez, 2006)</td>
<td>60</td>
</tr>
<tr>
<td>2.13</td>
<td>The process of Supervised Machine Learning (Kotsiantis, 2007)</td>
<td>65</td>
</tr>
<tr>
<td>2.14</td>
<td>An Example of Hidden Markov Model</td>
<td>68</td>
</tr>
<tr>
<td>2.15</td>
<td>Hidden Markov Model (Stamp, 2012)</td>
<td>70</td>
</tr>
<tr>
<td>2.16</td>
<td>Inductive Forward Procedure to Calculate $\alpha_t(i)$ from $\alpha_{t-1}(j)$ (Rabiner, 1989)</td>
<td>75</td>
</tr>
<tr>
<td>2.17</td>
<td>Inductive Backward Procedure to Find $\beta_t(i)$ from $\beta_{t+1}(j)$ (Rabiner, 1989)</td>
<td>78</td>
</tr>
<tr>
<td>2.18</td>
<td>Variables and Sequence of Necessary Operations for Calculation of the Joint Probability $\gamma_t(i,j)$ (Rabiner, 1989)</td>
<td>79</td>
</tr>
<tr>
<td>2.19</td>
<td>Gap of the Study</td>
<td>89</td>
</tr>
<tr>
<td>3.1</td>
<td>Research Framework</td>
<td>99</td>
</tr>
<tr>
<td>3.2</td>
<td>Research Design</td>
<td>101</td>
</tr>
<tr>
<td>3.3</td>
<td>High-Level Architecture of the Proposed Solution</td>
<td>102</td>
</tr>
<tr>
<td>3.4</td>
<td>Architecture of Proposed Solution</td>
<td>104</td>
</tr>
<tr>
<td>3.5</td>
<td>Two phases of Hidden Markov Model</td>
<td>106</td>
</tr>
<tr>
<td>3.6</td>
<td>Training Process and Specifying Threshold Value for Classification</td>
<td>108</td>
</tr>
<tr>
<td>3.7</td>
<td>Classification Process</td>
<td>109</td>
</tr>
<tr>
<td>3.8</td>
<td>A Well Balanced Protection (Vrabec and Harley, 2010)</td>
<td>112</td>
</tr>
<tr>
<td>3.9</td>
<td>True and False Detection</td>
<td>113</td>
</tr>
<tr>
<td>3.10</td>
<td>5-fold Cross Validation (Bramer, 2007)</td>
<td>116</td>
</tr>
<tr>
<td>4.1</td>
<td>Sections of PE files (‘.text’, ‘.data’, ‘.rsrc’, and ‘.reloc’) (Singh, 2009)</td>
<td>126</td>
</tr>
<tr>
<td>4.2</td>
<td>General Layout of Portable EXE File (Microsoft Corporation, 2013)</td>
<td>127</td>
</tr>
<tr>
<td>4.3</td>
<td>PE DOS Header (Singh, 2009)</td>
<td>129</td>
</tr>
<tr>
<td>4.4</td>
<td>PE Header (Singh, 2009)</td>
<td>129</td>
</tr>
<tr>
<td>4.5</td>
<td>Layout of PE File (Pietrek, 1994)</td>
<td>133</td>
</tr>
<tr>
<td>4.6</td>
<td>Code Segment Extractor</td>
<td>134</td>
</tr>
<tr>
<td>4.7</td>
<td>A Screen Capture of Header of a Sample PE File (“dinotify.exe”), in Binary Format, using UltraEdit Professional Text/Hex Editor</td>
<td>134</td>
</tr>
</tbody>
</table>
A Screen Capture of a Piece of the Code Segment of a Sample PE File (“dinotify.exe”), in Binary Format, using UltraEdit Professional Text/Hex Editor

Flowchart of PE Code Section Extraction (Govindaraj, 2008)

Opcode Sequence Generator

A Screen Capture of Instructions Opcode of the Code Segment of a Sample PE file (“dinotify.exe”), in Binary Format, using Turbo Debugger 32

A Screen Capture of the Opcode Sequence for a PE Program (“dinotify.ops”), using UltraEdit Professional Text/Hex Editor

Physical Distinction of Data Collection

Five-Fold Data Groups for Cross-Validation

Flowchart of generating Hidden Markov Models

Separation of LLPO scores for data set #1, N=2

Separation of LLPO Scores for Data Set #1, N=3

False Positives Produced for Data Set #4, N=3, Threshold = -3.90922, FP = 5, FN=0

False Positives and False Negatives Produced for Data Set #4, N=3, Threshold = -3.69471, FP = 1, FN=5

The Tradeoff between FPs and FNs Changing by Different Thresholds in the Range [-5,-2], for Data Test Set #2, with N=3

Comparison of Detection Rate (DR), False Positive Rate (FPR), and Overall Accuracy (OA) for Different Thresholds in the Range of [-5, -2], Test Set #2, N=3

Comparison of Detection Rate with the Previous Works

Comparison of False Positive Rate with the Previous Works

Comparison of False Negatives with the Previous Works

Comparison of Overall Accuracy with the Previous Works

Comparison of Detection Rate with the Previous Works
5.12 Comparison of False Positive Rate with the Previous Works

5.13 Comparison of Overall Accuracy with the Previous Works

A.1 Separation of LLPO Scores for Data Set #1, N=4

A.2 Separation of LLPO Scores for Data Set #1, N=5

A.3 Separation of LLPO Scores for Data Set #2, N=2

A.4 Separation of LLPO Scores for Data Set #2, N=3

A.5 Separation of LLPO Scores for Data Set #2, N=4

A.6 Separation of LLPO Scores for Data Set #2, N=5

A.7 Separation of LLPO Scores for Data Set #3, N=2

A.8 Separation of LLPO Scores for Data Set #3, N=3

A.9 Separation of LLPO Scores for Data Set #3, N=4

A.10 Separation of LLPO Scores for Data Set #3, N=5

A.11 Separation of LLPO Scores for Data Set #4, N=2

A.12 Separation of LLPO Scores for Data Set #4, N=3

A.13 Separation of LLPO Scores for Data Set #4, N=4

A.14 Separation of LLPO Scores for Data Set #4, N=5

A.15 Separation of LLPO Scores for Data Set #5, N=2

A.16 Separation of LLPO Scores for Data Set #5, N=3

A.17 Separation of LLPO Scores for Data Set #5, N=4

A.18 Separation of LLPO Scores for Data Set #5, N=5

B.1 The Tradeoff between FPs and FNs Changing by Different Thresholds in the Range [-5,-2], for Data Test Set #1, with N=2

B.2 The Tradeoff between FPs and FNs Changing by Different Thresholds in the Range [-5,-2], for Data Test Set #1, with N=3

B.3 The Tradeoff between FPs and FNs Changing by Different Thresholds in the Range [-5,-2], for Data Test Set #1, with N=4
B.4 The Tradeoff between FPs and FNs Changing by Different Thresholds in the Range [-5,-2], for Data Test Set #1, with N=5

B.5 The Tradeoff between FPs and FNs Changing by Different Thresholds in the Range [-5,-2], for Data Test Set #2, with N=2

B.6 The Tradeoff between FPs and FNs Changing by Different Thresholds in the Range [-5,-2], for Data Test Set #2, with N=3

B.7 The Tradeoff between FPs and FNs Changing by Different Thresholds in the Range [-5,-2], for Data Test Set #2, with N=4

B.8 The Tradeoff between FPs and FNs Changing by Different Thresholds in the Range [-5,-2], for Data Test Set #2, with N=5

B.9 The Tradeoff between FPs and FNs Changing by Different Thresholds in the Range [-5,-2], for Data Test Set #3, with N=2

B.10 The Tradeoff between FPs and FNs Changing by Different Thresholds in the Range [-5,-2], for Data Test Set #3, with N=3

B.11 The Tradeoff between FPs and FNs Changing by Different Thresholds in the Range [-5,-2], for Data Test Set #3, with N=4

B.12 The Tradeoff between FPs and FNs Changing by Different Thresholds in the Range [-5,-2], for Data Test Set #3, with N=5

B.13 The Tradeoff between FPs and FNs Changing by Different Thresholds in the Range [-5,-2], for Data Test Set #4, with N=2

B.14 The Tradeoff between FPs and FNs Changing by Different Thresholds in the Range [-5,-2], for Data Test Set #4, with N=3

B.15 The Tradeoff between FPs and FNs Changing by Different Thresholds in the Range [-5,-2], for Data Test Set #4, with N=4

B.16 The Tradeoff between FPs and FNs Changing by Different Thresholds in the Range [-5,-2], for Data Test Set #4, with N=5
B.17 The Tradeoff between FPs and FNs Changing by Different Thresholds in the Range $[-5, -2]$, for Data Test Set #5, with $N=2$ 235

B.18 The Tradeoff between FPs and FNs Changing by Different Thresholds in the Range $[-5, -2]$, for Data Test Set #5, with $N=3$ 236

B.19 The Tradeoff between FPs and FNs Changing by Different Thresholds in the Range $[-5, -2]$, for Data Test Set #5, with $N=4$ 237

B.20 The Tradeoff between FPs and FNs Changing by Different Thresholds in the Range $[-5, -2]$, for Data Test Set #5, with $N=5$ 238

C.1 Comparison of DR, FPR, and OA for Different Thresholds in the Range of $[-5, -2]$, Test Set #1, $N=2$ 240

C.2 Comparison of DR, FPR, and OA for Different Thresholds in the Range of $[-5, -2]$, Test Set #1, $N=3$ 241

C.3 Comparison of DR, FPR, and OA for Different Thresholds in the Range of $[-5, -2]$, Test Set #1, $N=4$ 242

C.4 Comparison of DR, FPR, and OA for Different Thresholds in the Range of $[-5, -2]$, Test Set #1, $N=5$ 243

C.5 Comparison of DR, FPR, and OA for Different Thresholds in the Range of $[-5, -2]$, Test Set #2, $N=2$ 244

C.6 Comparison of DR, FPR, and OA for Different Thresholds in the Range of $[-5, -2]$, Test Set #2, $N=3$ 245

C.7 Comparison of DR, FPR, and OA for Different Thresholds in the Range of $[-5, -2]$, Test Set #2, $N=4$ 246

C.8 Comparison of DR, FPR, and OA for Different Thresholds in the Range of $[-5, -2]$, Test Set #2, $N=5$ 247

C.9 Comparison of DR, FPR, and OA for Different Thresholds in the Range of $[-5, -2]$, Test Set #3, $N=2$ 248

C.10 Comparison of DR, FPR, and OA for Different Thresholds in the Range of $[-5, -2]$, Test Set #3, $N=3$ 249

C.11 Comparison of DR, FPR, and OA for Different Thresholds in the Range of $[-5, -2]$, Test Set #3, $N=4$ 250

C.12 Comparison of DR, FPR, and OA for Different Thresholds in the Range of $[-5, -2]$, Test Set #3, $N=5$ 251
| C.13 | Comparison of DR, FPR, and OA for Different Thresholds in the Range of [-5, -2], Test Set #4, N=2 | 252 |
| C.14 | Comparison of DR, FPR, and OA for Different Thresholds in the Range of [-5, -2], Test Set #4, N=3 | 253 |
| C.15 | Comparison of DR, FPR, and OA for Different Thresholds in the Range of [-5, -2], Test Set #4, N=4 | 254 |
| C.16 | Comparison of DR, FPR, and OA for Different Thresholds in the Range of [-5, -2], Test Set #4, N=5 | 255 |
| C.17 | Comparison of DR, FPR, and OA for Different Thresholds in the Range of [-5, -2], Test Set #5, N=2 | 256 |
| C.18 | Comparison of DR, FPR, and OA for Different Thresholds in the Range of [-5, -2], Test Set #5, N=3 | 257 |
| C.19 | Comparison of DR, FPR, and OA for Different Thresholds in the Range of [-5, -2], Test Set #5, N=4 | 258 |
| C.20 | Comparison of DR, FPR, and OA for Different Thresholds in the Range of [-5, -2], Test Set #5, N=5 | 259 |
**LIST OF ABBREVIATIONS**

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CFG</td>
<td>Control Flow Graph</td>
</tr>
<tr>
<td>DFA</td>
<td>Deterministic Finite Automaton</td>
</tr>
<tr>
<td>DOS</td>
<td>Disk Operating System</td>
</tr>
<tr>
<td>DR</td>
<td>Detection Rate</td>
</tr>
<tr>
<td>EPO</td>
<td>Entry Point Obfuscation/Obscuring</td>
</tr>
<tr>
<td>FN</td>
<td>False Negative</td>
</tr>
<tr>
<td>FP</td>
<td>False Positive</td>
</tr>
<tr>
<td>FPR</td>
<td>False Positive Rate</td>
</tr>
<tr>
<td>GD</td>
<td>General Decryption</td>
</tr>
<tr>
<td>HMM</td>
<td>Hidden Markov Model</td>
</tr>
<tr>
<td>LLPO</td>
<td>Log Likelihood Per Opcode</td>
</tr>
<tr>
<td>NGVCK</td>
<td>Next Generation Virus Creation Kit</td>
</tr>
<tr>
<td>OA</td>
<td>Overall Accuracy</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>PE</td>
<td>Portable Executable</td>
</tr>
<tr>
<td>RPME</td>
<td>Real Permutation Engine</td>
</tr>
<tr>
<td>TN</td>
<td>True Negative</td>
</tr>
<tr>
<td>TP</td>
<td>True Positive</td>
</tr>
<tr>
<td>VCL</td>
<td>Virus Creation Lab</td>
</tr>
<tr>
<td>Sens</td>
<td>Sensitivity</td>
</tr>
<tr>
<td>Spec</td>
<td>Specificity</td>
</tr>
</tbody>
</table>
# LIST OF APPENDICES

<table>
<thead>
<tr>
<th>APPENDIX</th>
<th>TITLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Tables and Scatter Plots of LLPO values for all 20 HMMs</td>
<td>199</td>
</tr>
<tr>
<td>B</td>
<td>Comparison Tables and Scatter Plots of False Negatives, False Positives, True Negatives, True Positives</td>
<td>218</td>
</tr>
<tr>
<td>C</td>
<td>Comparison Tables and Scatter Plots of Detection Rate, False Positive Rate, and Overall Accuracy</td>
<td>239</td>
</tr>
<tr>
<td>D</td>
<td>Source Code of “Feature Extraction” in C++</td>
<td>260</td>
</tr>
<tr>
<td>E</td>
<td>Source Code of “Proposed Model” in MATLAB</td>
<td>268</td>
</tr>
<tr>
<td>F</td>
<td>List of Publications</td>
<td>275</td>
</tr>
</tbody>
</table>
CHAPTER 1

INTRODUCTION

1.1 Introduction

These days, data are stored and shared in the digitally linked storage devices in the modern electronic world. The way we study, buy, play, work, earn money and live has become very different. The large majority of business transactions, which include extremely important and sensitive information, are performed via computers and over the digital networks and the internet. Therefore, it is imperative to care for information safety as a concern of dominant significance. Some digital applications, data processor and electronic storage devices, which can be connected through computer networks, are displayed in Figure 1.1.

![Figure 1.1 Storage and Sharing of Information in Modern Digital World](image-url)
Hence, we expose the safety and secrecy of private and personal information to danger in this way. We can see, even in this new kind of data storage and access system, information is even being stolen. We hear all the time, people’s personal information and money are stolen. Even worse, malicious programs wipe out valuable information and wealth of companies and organizations. Because the current digital world comprises multi-faceted vulnerabilities, the important issue is to protect data from being damaged or removed by malware programs. The destruction induced by malicious codes is more dangerous in today’s modern society, where personal and social communications and commercial business strongly depend on digital networks. In the latest study conducted by Symantec Corp., it is shown that the security is the top issue for 42 percent of organizations. In the past 12 months, 75 percent of organizations have suffered from computer network attacks that cost the enterprise commerce an average of 2 million dollars yearly. The result of the study is presented in Figure 1.2 (Symantec, 2010).

![Figure 1.2](image)

**Figure 1.2** Symantec Corp. Status of Enterprise Security in 2010: “Most Significant Risks and Cyber Attacks in Last Year” (Symantec, 2010)

The security problem has contributed to the development of antivirus production and it is now practically mandatory for all computer users to have antivirus softwares on their personal computers. Nevertheless, we still hear much news on recent malware and cautions against them daily, so therefore we need to update database files of our antivirus software to stay away from corruption and
prevent wider proliferation of malware. The most important reason is that existing antivirus products mostly work based on byte-to-byte comparison of files, where binary string sequences extracted from the analyzed viral files are exploited as signatures (Wang et al., 2003; Zhang, 2008). If a given file contains corresponding signatures existed in the patterns database, then it is accepted as a viral program. Therefore, there is a weakness in finding previously non-accurately analyzed viruses (Mori, 2004; Xu et al., 2013).

On the other hand, research and practical investigation in computer virology is still a contentious issue (Filiol et al., 2006; Marpaung et al., 2012). There is an extensively misconceived standpoint, which argues that research on computer virus and related areas not only is not productive, but also is potentially dangerous. The reason for this view is that the proponents of this opinion posit that it may lead to the growth of more overwhelming and harmful techniques of viral infection. They also believe that it is only waste of time, since fighting against the computer viruses is restricted to the found, analyze, extracted and identified signature of virus, the usual cycle of the antivirus software production (Filiol et al., 2006).

For this reason, only a few research groups and laboratories in organizations and universities worldwide investigate malware and study computer virology (Bonfante et al., 2006). Lack of knowledge in this subject has resulted in small number of significant scientific findings in this field have been obtained for this belief, while a logical and accurate view on the problem shows that more researches and practical examinations on the issue of computer virology are really vital and crucial (Filiol et al., 2006; Mansfield-Devine, 2013).

In order to protect our systems in an efficient way and predict computer malware risks before they practically appear as tools for attackers and virus creators, we need to profoundly, understand the threat we are facing (Mansfield-Devine, 2013).

As far as computer virology is concerned, unfortunately, there are several open problems remained unsolved, in aspects of modeling and implementation. Numerous new problems will certainly appear soon in the future, because of the progresses of malware creators. Whilst upcoming computer systems grow swiftly to
be complicated and sensitive, outdated defense and protection methods and models become even more inadequate (Filiol et al., 2006).

When we deal with metamorphic viruses, the problem becomes even worse (Singhal and Raul, 2012). Metamorphic viruses propagate by creating new transformed instances of its code (Leder et al., 2009). In other words, for a signature-based detector, metamorphic variants are considered as new viruses (Han et al., 2011b). Metamorphic viruses are created, propagated and spread more and more, because with virus creation kits and automatic metamorphic engines it is not necessary to be an expert and spend much time to create new malware. Consequently, the time needed for analyzing and reverse engineering of viral codes to extract new signatures is even more prolonged, and the size of database is becoming bigger.

In this study, a machine learning-based method, which uses an enhanced Hidden Markov Model is developed and implemented to analyze, classify and detect metamorphic computer viruses. This chapter concentrates on the background of the study, statement of the problem, research questions and objectives of the study. Then, the objectives of the research are presented, in more details. Next, the scope of the study, and in the last section, the significance of the research is explained.

1.2 Background of the Study

Early computer viruses were broadly spread in the 1980s, as a sequence of wide usage of personal computers and Microsoft’s new operating system (Sanok Jr., 2005; Makowsky, 2009). In late 1992, the reported number of malware spread in the cyber world was about 2,300 maximally, while this number reached to 60,000 known viruses until 2002, and in 2008, more than 100,000 computer viruses were detected and reported (Al Daoud et al., 2008). Unfortunately, the number of computer viruses and other varieties of malware are growing, tremendously.

In today’s digital world, although the reasons to create the new generations of computer viruses and malware programs have changed, we still hear about them in
digital news daily. There are new ways and causes, which motivate computer experts to work on new kinds of these programs, and consequently, information security researches should concentrate on this area to find the new methods to be able and to improve defensive systems against them. Computer networks, Internet and web-based applications are the main streams in spreading such codes over the electronic world and they help virus authors to share their malware programs, computer by computer.

To be safe from harms of various malicious codes, antivirus software producers utilize different methodologies to protect computer systems. They employ scanning methods to detect the file signature. Scanning methods usually consist of scanning email attachments, scanning downloaded files, and static file scanning. In addition, they make use of scanning by heuristic methods and General Decryption to fight against with modern and more complicated and advanced computer viruses (Sanok Jr., 2005).

A computer virus is usually a part of a saved program, which when it is run is able to generate a duplication of itself in another saved program (Cohen, 1987; Agapow, 1993; Cohen and Cohen, 1994; Johansson, 1994; Colombell, 2002; Rajala, 2004; Aycock, 2006; Khosrow Pour, 2007; Alsagoff, 2008). During the procedure of reproduction, virus can modify its code in many various methods. It is important to mention that the term “virus” is sometimes incorrectly used to describe different kinds of computer malware. A genuine virus is able to replicate itself and spread from a computer to another, usually via different forms of executables, but this ability is not contained in other kinds of malware, such as worm, trojans, and spywares. In this report, these terms are employed.

In computer virology, metamorphic code is a piece of program, which is able to rewrite itself with new format. Usually, it performs this action through converting its own program into a provisional version, revise this intermediate copy, and finally rewrite its code back to standard code, another time. This process is carried out on the virus by its own body, and consequently the metamorphic engine becomes different (Al Daoud et al., 2008). Some viruses use this approach while they are infecting new victims; as a result, the offspring are never similar to its original producer. Actually, this technique is utilized by the malware with the purpose of
keeping away from the signature detection of static scanning engines used in antivirus softwares. Therefore, the main function and purpose of the malware do not vary by this conversion, but all others may change.

In practice, metamorphic computer viruses effectively change some sequences of their instructions with syntactic or semantic corresponding instructions sequences, in consecutive offspring (Webster and Malcolm, 2006). Thus, although their code format actually looks different, the behaviors of all generations are similar. Typically, it is performed to prevent detection methods based on static analysis, which are normally used by antivirus software engines. Thus, static code scanning method to detect metamorphic viruses is not applicable; instead, heuristic analysis can be exploited to find unknown variants of a metamorphic virus.

Several researches have deployed different techniques of machine learning to apply heuristic analysis in detection and classification of metamorphic viruses. One of the newest methods, which have achieved heuristic detection of metamorphic variants, is Hidden Markov Model (HMM), which is an appropriate technique in statistical pattern recognition. In the early 1970s, firstly HMM was applied to speech recognition problem (Rabiner, 1989). Until now, it has been used in many other areas, especially in biological sequence problems (Krogh, 1998). In recent years, many attempts are carried out to apply HMM in the problem of metamorphic virus detection. Wong and Stamp (2006) trained their models on disassembled executable codes of viruses. The pre-process of disassembly needs a lot of time and output of the process depends upon the capability of the disassembler. In (Attaluri et al., 2009), authors used profile HMM and applied the method to analyze the metamorphic variants. Their findings proved that Profile Hidden Markov Models is successfully applicable to model the virus families. However, she also performed the implementation on assembly source codes. Govindaraj (2008), tried to implement the HMM algorithm on PE files. She followed the same methodology of (Wong, 2006), but she extended it into code segment of Portable Executable files.
1.3 Problem Statement

Since signature detection is the most commonly used detection strategy, virus writers have developed many techniques to evade such detection (Shanmugam et al., 2013). Metamorphic computer viruses change their code as they spread, with the intention of preventing detection by static signature-based virus scanners (Konstantinou, 2008; Leder et al., 2009; Murad et al., 2010; Santos et al., 2010; You and Yim, 2010; Xu et al., 2013). In order to achieve this goal, metamorphic viruses employ different code mutation methods to contest intense static analysis (Tabish et al., 2009; Runwal et al., 2012). They are able to defeat dynamic analyzers, as well, when they sense they are running and examining in an environment, like an emulator, which is monitoring their behaviors (O’Kane et al., 2011).

Detection of metamorphic computer viruses is not easy, because their authors have the knowledge of the feeblenesses of antivirus scanners. Static and dynamic analysis methods bring the limitations for antivirus scanners (Lee et al., 2011). Metamorphic and obfuscation techniques make virus finding by means of signature scanning practically impossible (Santos et al., 2010; Han et al., 2011b; O’Kane et al., 2011; Toderici and Stamp 2012). To detect a metamorphic computer virus, some other more complicated techniques such as inspecting the structure of the file, testing the behavior of the program, or machine learning methods must be used (Desai and Stamp, 2010). In other words, antivirus software should use heuristic techniques rather than string scanning to analyze and detect this kind of viruses (Konstantinou, 2008; Kasina et al., 2010). Given this situation, it is highly required that researchers in information security area make serious efforts in studying the metamorphism (Xufang et al., 2011). Today, a few researches are carried out to develop and implement new methods in order to improve the weakness of antivirus software’s against the modern techniques which are exploited by metamorphic virus authors (Santos et al., 2011). In response to this concern, this study was set up to investigate this issue further.

In this study, the following research questions will be answered:
1.4 Purpose of the Study

The purpose of this research is to propose and develop an enhanced model to classify and detect metamorphic viruses in format of portable executable. The performance of proposed model would also be evaluated and justified.

1.5 Objectives of the Study

The specific objectives of this study are:

i. To investigate the metamorphic computer viruses and their features.

ii. To analyze the existing metamorphic computer viruses classification and detection methods.

iii. To propose a new model for metamorphic computer virus classification and detection.

iv. To evaluate the performance of the proposed model.

1.6 Significance of the Study

The result of this research presents a noticeable knowledge in the computer security and virology science area. In addition, the result of this study can be
exploited by antivirus software vendors. They can use the proposed model to improve antivirus products in defending against metamorphic computer viruses.

1.7 Scope of the Study

The focus of this study is on applying an enhanced form of Hidden Markov Model, particularly on the Portable Executable files. This study is to train the proposed model on the binary format of PEs, directly. The proposed model is applied on metamorphic virus family to involve the statistical features of the family.

The metamorphic viruses chosen for the data set used in the experiments in this study include Next Generation Virus Creation Kit (NGVCK) virus family. The NGVCK virus family have been chosen, because based on the study done by Wong (Wong, 2006), the NGVCK virus family is able to create viruses that share only a few percent of similarity. It means the NGVCK is the most powerful morphing engine to create the metamorphic virus.

1.8 Structure of the Thesis

In Chapter 1, an introduction to digital security and computer virology is given. A brief description of the metamorphic computer virus is provided. The problem statement, purpose and objectives of the study are presented.

In Chapter 2, a literature review on evolution of the computer virus concealment strategies, code obfuscation techniques, metamorphic computer virus detection techniques are given. In addition, some definitions of classification are introduced and Hidden Markov Model is explained, in details. In the last part of chapter 2, some of more related recent works are reviewed and gap of the study is presented.

Chapter 3 begins with an introduction to different research paradigms, research framework, and research design. Then, the proposed method and its
evaluation are explained in details. At the end of chapter 3, instrumentation, assumptions and limitations of the study are presented.

Chapter 4 focuses on the experimental design. This chapter contains experimental setup, feature extraction, data set and implementation of the model.

In Chapter 5, the results of the experiments are given; the analysis of the proposed model and the threshold value are presented. Moreover, evaluation and justification of the proposed model based on a comparative study between the proposed model and four more related recent works are also given and discussed. The results of the proposed model are also compared with 44 famous commercial antiviruses.

Chapter 6 contains discussion and conclusions. Finally, the contributions and importance of this study, and the some recommendations for the future studies are given.
REFERENCES


Roll-Hansen, N. (2009). *Why the distinction between basic (theoretical) and applied (practical) research is important in the politics of science*. 


