ENHANCEMENT OF AUTOMATED BLACK-BOX WEB APPLICATION VULNERABILITY ASSESSMENT ALGORITHMS

LIM KAH SENG

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

School of Computing
Faculty of Engineering
Universiti Teknologi Malaysia

OCTOBER 2019

DEDICATION

This thesis is dedicated to my family members for their moral support and upbringing. This thesis also dedicated to Him, which the challenges poured by Him had shaped me into a better person.

ACKNOWLEDGEMENT

Many people have been pivotal in aiding me toward completing this thesis. Not only did they deliver the necessary knowledge to widen my point of view towards the study of computation and its vulnerability assessment, but also the mental supports. Without them, it would have been difficult for me to complete this research work.

First of all, I would like to express my gratitude to my supervisors, Assoc. Prof. Dr Norafida Ithnin and Dr Syed Zainudeen for their guidance. Without their guidance, it would have been impossible for me to produce this thesis work. My sincere appreciation also to CSM for the willingness to participate in my research work.

I would like to thank my family members for their moral and financial support, as well as my close friends Miss Cindy Puah and Mr Wan Mohd. Yacob, for their companionship.

Finally, all gratitude to Him for the challenges which made me a better person. In addition, I would like to also thank all those who had lent me a helping hand, and whose names are not mentioned here. The completion of this thesis would be impossible without these people.

ABSTRACT

Presently, the web application vulnerability assessment has been widely automated to shorten the web application penetration testing life-cycle. Unfortunately, in the testing environment of the black-box where web-based application codes are unreachable, the automation of web application vulnerability assessment tend to produce the false negatives. This research was conducted to enhance the present state-of-the-art automated web application vulnerability assessment and mitigate the research problems of test coverage and false negatives. In this research, three enhancements were developed to address the problems. The first enhancement involved the improvement of current web-based application reconnaissance solution, using the derived algorithms for form fills and input generation. The second enhancement improved the existing vulnerability assessment solutions by using an invented algorithm, which implemented the execution-path oriented analysis. The final enhancement improved the present vulnerability detection solution with an algorithm that detects vulnerability using a proposed execution path-oriented data flow analysis and fuzzy set theory. This research was conducted based on applied research method, which covered literature reviews, requirement analysis, and preliminary experimentation that led to the creation of the stated algorithms. In addition, a prototype automated black-box web application vulnerability assessment tool was conceived using Java programming language as well as Selenium and Crawljax frameworks. An experimentation was conducted to quantitatively benchmark the validity of the algorithm using twelve test-beds, composed of vulnerable web-based applications, and eight existing automated black-box web application vulnerability assessment tools. The experimental results showed there was an improvement of test coverage by 14.35% and a reduction of false negative by 64%. In conclusion, the enhancements made using the proposed algorithms have improved the automated web application vulnerability assessment test coverage and reduced the false negatives.

ABSTRAK

Pada masa kini, penilaian kerantauan aplikasi web secara automatik telah diguna dengan meluas untuk memendekkan kitaran hayat ujian penembusan web aplikasi. Malangnya, dalam persekitaran ujian kotak hitam di mana kod web aplikasi tidak boleh dicapai, penilaian kerantauan aplikasi web secara automatik cenderung untuk menghasilkan negatif palsu. Kajian ini telah dilaksanakan untuk memperbaiki kelemahan dalam penilaian kerantauan aplikasi web secara automatik untuk mengurangkan masalah liputan ujian dan negatif palsu. Dalam kajian ini, tiga penambahbaikan telah dibangunkan untuk menangani masalah. Peningkatan pertama melibatkan penambahbaikan penyelesaian pengiktirafan aplikasi berasaskan web semasa menggunakan algoritma yang diciptakan untuk mengisi borang web dan penjanaan input. Penambahbaikan yang kedua mengandungi penambahbaikan penyelesaian penilaian kerantauan semasa dengan penciptaan algoritma yang melaksanakan analisa berorientasikan laluan pelaksanaan. Penambahbaikan terakhir mengandungi penambahbaikan penyelesaian pengesanan kerantauan dengan algoritma yang melaksanakan analisa aliran data berasaskan laluan pelaksanaan dan teori set fuzzy. Kajian ini dilaksanakan berdasarkan kaedah penyelidikan penggunaan di mana hasil daripada kajian literatur, analisis keperluan, dan eksperimen awal telah membawa kepada ciptaan algoritma yang dinyatakan. Di samping itu, prototaip penilaian kerantauan aplikasi web kotak hitam automatik disusun menggunakan bahasa pengaturcaraan Java, serta kerangka Selenium and Crawljax. Kajian telah dijalankan untuk menilai kesahan algoritma secara kuantitatif dengan menggunakan dua belas ujian katil, terdiri daripada web aplikasi terdedah dan lapan alat penilaian kerantauan aplikasi web kotak hitam automatik yang terkini. Hasil kajian menunjukkan terdapat peningkatan ujian liputkan sebanyak 14.35% serta mengurangkan negatif palsu sebanyak 64%. Kesimpulannya, penambahbaikan yang dibuat menggunakan algoritma yang dicadangkan telah berjaya meningkatkan liputan ujian penilaian kerantauan aplikasi web secara automatik dan mengurangkan negatif palsu.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

WWW	-	World Wide Web
ICT	-	Information and Communication Technology
DEP	-	Data Entry Point
RIA	-	Rich Internet Application
DoS	-	Denial-of-Service
PTES	-	Penetration Testing Execution Standard
PCI DSS	-	PCI Data Security Standard
SUT	-	System Under Test
LITE	-	Layout-based Extraction Technique
IKM	-	Information Knowledge Manager
AADT	-	Access Authorization Data Table
DDES	-	Double Duplication Elimination Strategy
JavaSye	-	Java Symbolic Execution Engine
OWASP	-	Open Web Application Security Project
WASC	-	Web Application Security Consortium
IDS	-	Intrusion Detection System
LCTS	-	Longest Common Tag Sequence
RKR-GST	-	Running Karp-Rabin Greedy String Tilling
WASSEC	-	Web Application Security Scanner Evaluation Criteria
SQL	-	Structure Query Language
HTML	-	HyperText Markup Language
XQuery	-	XML Query
XML	-	Extended Markup Langauge
URL	-	Uniform Markup Language
UTF	-	Unicode Transformation Format
mXSS	-	Mutation Cross-site Scripting

XSS	-	Cross-site Scripting
LAN	-	Local Area Network
ACM	-	The Association of Computing
IEEEXplore	-	IEEE Xplore Digital Library
CSM	-	CyberSecurity Malaysia
Wivet	-	Web Input Vector Extractor Teasor

LIST OF SYMBOLS

T	-	DEP data type
S_a	-	A set of data entry points
A_n	-	Parameters of data entry points
T_i	-	Data entry points data type
C	-	Cookie
S_r	-	Cookie parameter
TG	-	Data entry point URL
N	-	Data entry point name
H	-	Data entry point hash value
V	-	Data entry point input value
M	-	A finite state machine
S, Q	-	A set of states
δ	-	A set of input functions
$\sum$	-	The input to trigger web application event.
I_o	-	The initial node in a finite state machine
G	-	A graph
V	-	A set of vertexes
E, e	-	A set of edges
I	-	An input
v	-	A vector
$dompath$	-	The DOM path to href link
$action$	-	The attribute to split href link by '/'
$params$	-	The form names and anchors
$value$	-	The value to input web form
$delta(x)$	-	The infinite section in a web application
$\text{HTTP}_{\text{request}}$	-	The expected HTTP request

$\text{expected}_{\text{condition}}$	-	The expected web application response
$\text{LCTS}_{\text{amount}}$	-	The longest amount of a DOM document tag sequence
$\text{Tag}_{\text{amount}}$	-	DOM document HTML tags
h_i	-	A DOM document
$coverage$	-	The test coverage
$length(h_i)$	-	The length of a DOM document
V_D	-	The vulnerability detection rate
V_d	-	The number of vulnerability detected
T_d	-	The total number of vulnerability presents
FP_d	-	The number of false positive detected
FN_d	-	The number of false negative detected
WD	-	The number of web pages visited
ST	-	The differences of initial and ending scanning time
T_{end}	-	The ending scanning time
$\text{T}_{\text{initial}}$	-	The initial scanning time
TP	-	The number of true positives
TV	-	The total number of vulnerabilities in a test-bed
S_{TP}	-	The scanner specific true positive instance
S_{TN}	-	The scanner specific true negative instance
S_{FP}	-	The scanner specific false positive instance
S_{FN}	-	The scanner specific false negative instance
V_{TP}	-	The vulnerability specific true positive instance
V_{TN}	-	The vulnerability specific true negative instance
V_{FP}	-	The vulnerability specific false positive instance
V_{FN}	-	The vulnerability specific false negative instance
P	-	A web application
$f(x)_n$	-	A finite set of input function
ψ	-	A transition function
w	-	The web page transition

e	-	The web application event
x	-	An input value
$links$	-	The web element of data type link
$forms$	-	The web element of data type web forms
$options$	-	The web element of data type option
$textarea$	-	The web element of data type textarea
$buttons$	-	The web element of data type button
S_{link}	-	A set of web element of data type link
S_{form}	-	A set of web element of data type form
S_{option}	-	A set of web element of data type option
$S_{textarea}$	-	A set of web element of data type textarea
S_{button}	-	A set of web element of data type button
H	-	An instance of web application execution behaviour
a	-	A user action on web application
id	-	The number of id of a node of a directed graph
$name$	-	The name of a node of a directed graph
$elements$	-	A set of candidate elements on a web page
DOM	-	The DOM document
$stripped_{DOM}$	-	The DOM document with HTML tags only
tag	-	The web element HTML tag
$type$	-	The web element HTML type
$attribute$	-	The attributes belong a web element
P_i	-	A program path
ep_i	-	An execution path
$/\backslash C$	-	The input constraint
S_i	-	An application state in an execution path
$target_{vectori}$	-	An attack vector where attack payload is placed
$related_{vectori}$	-	An attack vector where Innocent input is placed
R	-	The web application response

ex	-	An instance of successful exploitation
y	-	The attack payload
$f(y)_n$	-	An input function that accepts attack string
U	-	The universal set
m	-	The fuzzy set membership function
μa	-	The value of membership function of attack string
μb	-	The value of membership function of attack vector
μc	-	The value of membership function of application state
μd	-	The value of membership function of attack consequence
$m(a)$	-	The membership function of attack string
$m(b)$	-	The membership function of attack vector
$m(c)$	-	The membership function of application state
$m(d)$	-	The membership function of attack consequence
$state_i, S_i$	-	An instance of state
v_i	-	An instance of attack vector
a_i	-	An instance of attack string
c_i	-	An instance of attack consequence

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Area of Research

These days, a compilation of technical documents called web-based application, is no more a demonstration of a simple asset for information sharing, as it was originally intended by its inventor, Sir Tim Bernes-Lee, in 1989. The maturity and availability of fast-internet speeds, as well as the rapid growth of web-based technology, have yielded widespread usage of web-based applications in fields such as politics, education, and many more. Today, web-based applications is a ubiquitous platform which helps people stay connected, as well aid in data sharing. This includes leveraging web-based applications for business products or services promotion and delivery. Therefore, modern web-based applications are always an attractive target for intruders. In addition, the conventional Information and Communication Technology (ICT) infrastructures such as network or web servers, in particular, web-based applications, possess high accessibility, which is accessible by anyone, anywhere, and anytime, 24/7. Moreover, OWASP report OWASP (2017) showed that modern web-based applications tend to be much more susceptible to several vulnerabilities; this includes injection-based vulnerabilities, broken authentication, sensitive data exposure, XML external entities, broken access control, security misconfiguration, cross-site scripting, insecure deserialization, and usage of the component with known vulnerabilities. Thus, vulnerable web-based applications are usually the gateway to gain access to organization protected infrastructures, or data.

Conventionally, an attack is launched through the injection of malicious data onto web-based application data entry points (DEPs), to compromise web-based application confidentiality, integrity, or availability. Consequently, the web-based application's defensive mechanisms always plays a crucial role in defending web-based applications against attacks by preventing malicious data from entering web-based application DEPs. Unfortunately, writing a solid defensive mechanism is tedious and error-prone, as the combination of data that can be possibly accepted by a web-based application, DEP, is always vast in number (Jovanovic *et al.*, 2006). Therefore, besides implementing countermeasures such as input sanitization functions, it is essential to have web-based application security assessed during, and after the development phase.

Typically, test engineers assess web-based application security via the injection of predefined attack strings onto web-based application DEPs, which compromises web-based application confidentiality, integrity, or availability for identification of security loopholes (Black, 2011; Bathia *et al.*, 2011; Palsetia *et al.*, 2016). To assess web-based application security, practitioners have introduced miscellaneous black-box, white-box, and grey-box testing techniques. Related black-box testing techniques dynamically analyse web-based application execution behaviours for vulnerability detection. The white-box testing techniques inspect a web-based application's codes for defects. The grey-box testing technique is a combination of both black-box and white-box testing techniques (Moonen, 2011; Liu *et al.*, 2012). Existing well-known web-based application security assessment techniques have included fuzzing, code review, penetration testing, just to name a few (Liu *et al.*, 2012; Avramescu *et al.*, 2013).

However, the manual web-based application vulnerability assessment is time-consuming, error-prone, and tedious, as a human being tends to make mistakes. Henceforth, practitioners have invented, designed, and developed algorithms to automate the web-based application vulnerability, using the power of computation (Holm *et al.*, 2013; Awang and Manaf, 2013).The outcome of this automation process is the automated web-based application vulnerability assessment tools, which have helped automate web-based application vulnerability assessment, by simulating test engineer actions by penetrating web-based application DEPs with selected attack strings. This compromises web-based application DEP security in order for security loophole identification. The vulnerability detection is achieved through the inspection of web-based application responses towards an attack string (Black, 2011; ĐURIĆ, 2014; Kaushik and Singh, 2013). Presently, the tool has been widely used in web-based application penetration testing to automate the testing phase of the vulnerability assessment.

The web-based application codes may or may not reachable during the automated web application vulnerability assessment. To meet the challenges of assessing web-based application security in testing environments of black-boxes and white-boxes, the automated white-box and black-box web application vulnerability assessment was invented. The automated white-box web application vulnerability assessment parses web-based application codes, performs information or data flow analysis on web-based application codes for security loopholes identification. On the other hand, the automated black-box web application vulnerability assessment dynamically analyses web application execution behaviours toward the malicious data for the same purpose of vulnerability detection. Besides this, there is also a hybrid solution, which integrates the white-box and black-box testing techniques to achieve better scanning efficiency, test results, as well as test coverage (Vieira *et al.*, 2009; Tung *et al.*, 2014; Ben Jaballah and Kheir, 2016).

1.2 Research Background

Although the automation of web application vulnerability assessment shortens the testing life-cycle, this allows parallel testing, as well as enables test engineers to focus on tasks that require manual testing. However, the current state-of-the-art contains limitations of the false positives and false negatives. The false positive is the fake vulnerability, that gets reported, while the false negative, is the benign vulnerability which was not successfully detected (Doupé *et al.*, 2012; Gol and Shah, 2015; Vieira *et al.*, 2009).

According to the research outcomes of Fonseca *et al.* (2014b); Baral (2011); Antunes and Vieira (2014, 2017); Wang *et al.* (2010); Rahman *et al.* (2017), false positive tends to be yielded by automated white-box web application vulnerability assessment. In the opposite, automated black-box web application vulnerability assessment tends to produce the false negative. Between the two limitations, false positives are much worse than the false negatives. False positives cost test enginners extra time and effors for fake vulnerability elimination. However, false negatives create the confusion. The false negative misleads the test engineer into mistreating a web-based application security by putting the vulnerable web application continuously expose to intruder attacks (Díaz and Bermejo, 2013; Yeo, 2013; Suto, 2010). This is despite the fact that invention of the hybrid solution was intended to mitigate the related limitations of the false positives and false negatives. Unfortunately, in the testing environment of black-boxes, and in the event that web-based application codes are not reachable, the present hybrid solution will have behaved as another automated black-box web application vulnerability assessment, with the same unresolved limitations of false positives and false negatives. This elaborates the phenomena of why existing hybrid solutions are barely able to replace automated black-box web application vulnerability assessments (Tripp *et al.*, 2013; Medeiros *et al.*, 2014). Consequently, this research has made the choice of enhancing the state-of-the-art automated black-box web application vulnerability assessment, for mitigating the limitations of the false negatives. Besides that, this research also covered the research issues of test coverage, as it was noted that there is a close relationship between the two attributes.

The state-of-the-art automated black-box web application vulnerability assessment is language independent. However, without code accessibility, there exists challenges to include every web-based application content into testing. This including to systematically test the web-based application's security, as well as to locate the security loopholes. Therefore, algorithms which provide solutions for web-based application reconnaissance, attack vector security assessment, as well as vulnerability assessments, have been widely invented by practitioners to automate web-based application vulnerability assessment in the testing environment of black-boxes. During the automated black-box web application vulnerability assessment, the reconnaissance solution systematically crawls web-based applications to retrieve and include web-based application contents into automated web application vulnerability assessments, with web-based application DEPs discovery being the main priority. Subsequently, the solutions for attack vectors security assessment plants attack strings into web-based application DEPs for compromising web-based application confidentiality, integrity, and availability. Lastly, the solutions for vulnerability detection inspects web-based application responses for vulnerability detection.

Consequently, in an automated black-box web application vulnerability assessment, there usually exists three critical features, which are web-based application reconnaissance, attack vector security assessment, and web-based application vulnerability detection (Aliero and Ghani, 2015; Balduzzi *et al.*, 2011; Chen and Wu, 2010; Makino and Klyuev, 2015; Rocha *et al.*, 2012; Vithanage and Jeyamohan, 2016).

The automated black-box web application vulnerability assessment has historically failed in crawling the modern web-based application, to include target web-based application web contents for automated vulnerability assessments (Choudhary *et al.*, 2012; Benjamin *et al.*, 2010; Benedikt *et al.*, 2002; Muñoz and Villalba, 2015; Barbosa and Freire, 2007; Raghavan and Garcia-Molina, 2000; Wang *et al.*, 2010). Besides this, the present state-of-the-art also possesses limitations for systematically testing modern web-based applications, which continuously expand in both complexity and size. This includes producing solutions to assess attack vector security, as well as to detect successful exploitation for vulnerability detection. Consequently, the current state-of-the-art solutions have always failed to include web content into testing with vulnerabilities which are typically are missed (Khoury *et al.*, 2011a,b; Dao and Shibayama, 2010; Antunes and Vieira, 2017). In addition to that, the default heterogeneous nature of the modern web-based application makes the web-based application a complicated target. The integration of modern web technologies have improved the modern-based web application's responsiveness, performance, and functionality. However, this evolution also increased the difficulty of automating the web application vulnerability assessment. Thus, the quick pavement for the web-based application technology advancement also caused the current state-of-the-art of automated web application vulnerability assessment to come to terms of fast elimination.

1.3 Research Problem

The current state-of-the-art of automated black-box web application vulnerability assessment suffers from limitations of test coverage and false negatives. The literature review, as presented in Chapter 2, shows that web-based application reconnaissance solutions, such as those proposed by Huang and Lee (2005); Duchene *et al.* (2013); Fung *et al.* (2014) have delivered the necessary solutions for hidden web crawling. However, these solutions usually ignore the considerations of web-based application contexts and semantics. Consequently, the current proposed reconnaissance solutions are only capable of reaching a minor percentage of the hidden web contents. Successes which had been achieved under current situations have included test input generation, web semantic extraction, as well as form inputs for web-based application authentication scheme bypassing, as well as for navigation and event triggering. Unfortunately, current suggested web-based application reconnaissance solutions rely too much on the manual, or random approaches which tend to fail to meet practitioner's expectations. These manual approaches negatively affect the scanning performance and experiences, through iterative disrupting of the crawling process, prompting test engineers for manual web input. On the other hand, random approaches are always inaccurate, in the sense that executed computation steps are too easily discarded, or ignored through the implemented defensive mechanisms. Thus, the existing web-based reconnaissance solutions usually possess test coverage issues due to many web-based application contents which are untested (Khoury *et al.*, 2011a; Duchene *et al.*, 2013; Tripp *et al.*, 2013).

On the other hand, current solutions for vulnerability assessment still heavily rely on the traditional approaches of the capture-and-replay sequence, which is highly randomised. The solution performs random point-and-shoot approaches to have the attacker's vectors security assessed, while incorporating brute force techniques of fault injection, and fuzzing to penetrate the attacker's vector security. Practitioners have produced the necessary countermeasures such search-based testing techniques, genetic algorithms, learning-based algorithms, and perturbation techniques, just to name a few, to improve the quality of brute force mechanisms, however, the approach of assessing the attacker's vector security has received less attention. It usually remains randomised, and often does not consider the web-based application's context during the automated black-box web application vulnerability assessment. Thus, the inclusion of the discovered attacker's vectors into the automated black-box web application vulnerability assessment is not guaranteed (Doupé *et al.*, 2012; Alata *et al.*, 2013; Antunes and Vieira, 2014; Dao and Shibayama, 2010).

Lastly, there is a vulnerability detection solution to inspect web-based application responses, using the conventional pattern matching and anomaly detection technique for vulnerability detection. Unfortunately, these vulnerability detection solutions are too conservative. The existence of specific keyword or anomaly does not necessarily mean that there are security loopholes in the assessment web-based application. Moreover, without the acknowledgement of the relationship between the source and sink, current vulnerability detection solutions often fail to locate these security loopholes (Dao and Shibayama, 2010; Antunes and Vieira, 2014).

1.4 Problem Statement

The literature review outcomes, in coming Chapter 2, show that there are limitations in the current web-based application reconnaissance, vulnerability detection, and vulnerability assessment solutions for the automated black-box web application vulnerability assessment. The combination of these weaknesses have negatively affected the present state-of-the-art of automated black-box web application vulnerability assessment capability. This includes the introduction of research problems pertaining to test coverage and false negatives.

1.5 Research Aim

The research problems associated with test coverage and false negatives have sabotaged the usefulness of automated black-box web application vulnerability assessments. Therefore, this research aims to produce the necessary countermeasures to mitigate these limitations, to improve the state-of-the-art's test coverage and false negatives.

1.6 Research Question

Limitations in the current web-based application reconnaissance, vulnerability assessment and detection solutions are factors which contribute to the issues of test coverage and false negatives. Thus, this research mainly seeks to answer the research question of exploring hidden web content, as well as assessing the web-based application security for security loopholes detection. The research's main research question is presented below.

How to heuristically and automatically assess and explore modern web-based application contents without concerns the issues of code accessibility?

This main research question covers the aspect of how to produce the solutions, or the algorithms, for performing the necessary heuristic web-based application reconnaissance in the test environment's black-box. Besides this, the main research question also covers the necessary aspects of investigating the answers of how to assess modern web-based application attack vector securities, as well as to detect security loopholes in the test environment's black-box.

The production of desired algorithms requires this research work to clarify the default nature of modern web-based applications and web-based application's vulnerability, which includes defining the test approaches for locating the security loopholes. This is investigated from the perspective of intruders, which will help to answer the research question on how to manage web exploitation for revealing security loopholes revealing. This yields the following five sub-research questions:

(a) What is a modern-based application?

This sub-research question investigates the web-based application's state-of-the-art. The study covers the investigation of modern web-based application architectures and its deployment. Section 2.2 of Chapter 2 presents the answer to this research question.

(b) What is web-based application vulnerability or security loopholes?

This sub-research question investigates the default nature of web vulnerability or security loopholes, to help create effective algorithms for vulnerability or security loophole detection. The activity also includes an examination of the latest web-based application vulnerability statistic reports for vulnerability trends discovery. The study and answer to this sub-research question is available in Section 2.8 of Chapter 2. Moreover, this sub-research question also enables this research work to investigate the state-of-the-art for automated black-box web application vulnerability assessments, as presented in Section 2.6 of Chapter 2.

(c) How to penetrate web-based application security?

This sub-research question defines the web-based application exploitation, which leads to studying of the state-of-the-art for automated black-box web application's vulnerability assessment and its algorithms. By focusing on web exploitation and vulnerability assessment, this helps to produce new security assessment solutions. This relevant study is found in Section 2.5 of Chapter 2.

(d) What is the object to be tested in automated black-box web applications?

This sub-research question investigates current reconnaissance techniques of automated black-box web application vulnerability assessments. The answer to this sub-research question defines objects to be scanned in the automated black-box web application vulnerability assessment for security loophole detection. The relevant study is found in Section 2.4 of Chapter 2.

(e) How external entities relate to web-based application exploitation, vulnerability detection, and vulnerability assessment?

This sub-research question investigates relationships between external entities, web exploitation and vulnerability. Automation of web-based application exploitation, vulnerability detection, and vulnerability assessment requires valid interaction between the external entities, and the target web-based application. Thus, this research work provides a means to study and investigate the corresponding relationship. The relevant study is found in Section 4.3.1 of Chapter 4.

1.7 Research Objectives

The objectives of this research work are to mitigate research problems of test coverage and false negatives of the automated black-box web application vulnerability assessment, which includes:

(a) To enhance current web-based application reconnaissance solutions with algorithms that perform heuristic form filling and input generation.

(b) To enhance current vulnerability assessment solutions with algorithm that perform the proposed execution path-oriented vulnerability assessment.

(c) To enhance current vulnerability detection solutions with algorithm that perform the proposed execution path-oriented data flow analysis and fuzzy testing.

1.8 Research Scopes

To maintain the sustainability of this research work, the following research scopes were defined:

(a) This research work addresses limitations of test coverage and false negatives of automated black-box web application vulnerability assessment only.

The limitation of false negatives is severe in automated black-box web application vulnerability assessments, of which this research problem places vulnerable web-based application exposure to deal with attacks. Moreover, there is a close relationship between the two attributes of test coverage and false negatives. The higher the test coverage, the lower the number of false negatives. Consequently, this research considers both attributes of test coverage and false negatives only. The improvements brought by this research work benefits communities by providing a state-of-the-art solution with precise test results. Unfortunately, this research will not cover other attributes such as scanning efficiency. In addition to that, this research work also includes web-based applications without considering other web components, such as web servers, databases, or web browsers. Web browsers, however, will be used to simulate the web browsing environment for automating the web-based application's vulnerability assessment in the testing environment of the black-box.

(b) This research work considers cross-site scripting vulnerability as an attribute for bench-marking purpose only.

This research work does not have the intention to study the web-based application's vulnerability. Moreover, it is nearly impossible to include every vulnerability in this single study. Consequently, this research work uses web-based application vulnerability of cross-site scripting for bench-marking purposes only. Cross-site scripting was chosen for two reasons. Firstly, the current detection rate of the cross-site scripting is still low compared to its competitor with a SQL injection. Secondly, the cross-site scripting vulnerability usually affects the web-based application's security only, without harming other web components such as the web server or database. In this research work, a cross-site scripting attack library was used to penetrate the web-based application's security, and cross-site scripting detection rate was measured to benchmark the proposed algorithms.

(c). This research does not considers the SilverLight and Flash technologies.

The introduction of web technologies such as HTML5, CSS 3, and JavaScript ES 6 have successfully replaced conventional technologies such as Adobe Flash and Microsoft SilverLight. It delivers the necessary solutions for developing responsive and dynamic web-based applications, but with more speed. Moreover, the support for such technologies as Microsoft SilverLight and Flash will end by 2020. Consequently, this research work has chosen to exclude these technologies.

1.9 Research Significant

Overall, this research work has delivered the automated black-box web application vulnerability assessment's state-of-the-art, modern web-based application's architecture, as well as the existing research trends. Besides this, the research has also invented the algorithms to improve the existing web-based application's reconnaissance, vulnerability assessment, and vulnerability detection solutions for mitigating limitations of test coverage and false negatives.

Enhancement of the web-based reconnaissance solution with the invented algorithms for heuristic form filling algorithms and input generation algorithms successfully includes the micro hidden contents into testing. The invented reconnaissance algorithm will be discussed in Chapter 4.

Besides this, the enhancement of the vulnerability assessment solution with the invented execution path-oriented analysis algorithm successfully achieves much more complete testing. The invented vulnerability assessment algorithm will be discussed in Chapter 5.

Lastly, the enhancement of vulnerability detection solutions with the invented execution path-oriented data flow analysis algorithm and the fuzzy algorithm has improved the cross-site scripting detection rate. The invented vulnerability detection algorithm will be discussed as well in Chapter 5.

1.10 Thesis Organisation

Overall, this thesis comprises of seven chapters.

Chapter 1 presents the research area of interest, which covers the research objectives, research scopes, research questions, research aims, research significance, the background of the study, as well as the problem statement.

Chapter 2 will present the research trends, patterns and the state-of-the-art of the automated black-box web application's vulnerability assessment. Besides this, Chapter 2 also illustrates the related research works related to the automated black-box web application's vulnerability assessment, and the present research gap.

Chapter 3 will present this research work's research methodology, which was designed to enhance the state-of-the-art for the automated black-box web application's vulnerability assessment using the applied research method. The designed research methodology consists of three research phases: investigation and clarification, design and development, as well as experimentation and validation.

Chapter 4 will present preliminary experiments that were conducted to investigate the requirements of the automated black-box web application's vulnerability assessment. Chapter 4 also presents the enhancements made for the web-based application's reconnaissance.

Chapter 5 will present enhancements made for the web-based application's vulnerability assessment and vulnerability detection solutions.

Chapter 6 will present and discuss the validity of the enhancements made for the the state-of-the-art automated black-box web application vulnerability assessment.

Finally, Chapter 7 will conclude this thesis with an overall presentation of the research work's contributions, and proposals for future research works.

REFERENCES

(2019). *WebGoat 8.0. Contribute to WebGoat/WebGoat development by creating an account on GitHub*. Retrievable at https://github.com/WebGoat/WebGoat, original-date: 2015-03-06T14:02:02Z.

A.A.Puntambekar (2010). *Data Structures*. Technical Publications. ISBN 978-81-8431-774-9. Google-Books-ID: 2lvbJjTITuMC.

Acunetix (2018a). *acublog news*. Retrievable at http://testaspnet.vulnweb.com/.

Acunetix (2018b). *acuforum forums*. Retrievable at http://testasp.vulnweb.com/.

Acunetix (2018c). *Home of Acunetix Art*. Retrievable at http://testphp.vulnweb.com/.

Acunetix (2018d). *SecurityTweets - HTML5 test website for Acunetix Web Vulnerability Scanner*. Retrievable at http://testhtml5.vulnweb.com/#/popular.

Akrout, R., Alata, E., Kaaniche, M. and Nicomette, V. (2014). An automated black box approach for web vulnerability identification and attack scenario generation. *Journal of the Brazilian Computer Society*. 20(1), 4.

Alata, E., Kaâniche, M., Nicomette, V. and Akrout, R. (2013). An automated approach to generate web applications attack scenarios. In *Dependable Computing (LADC), 2013 Sixth Latin-American Symposium on*. April. Rio de Janeiro, Brazil: IEEE, 78–85.

Alexa (2018). *alexa top site - Google Scholar*. Retrievable at https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=alexa+top+site&btnG=.

Ali, A. B. M., Abdullah, M. S. and Alostad, J. (2011). SQL-injection vulnerability scanning tool for automatic creation of SQL-injection attacks. *Procedia Computer Science*. 3, 453–458.

Aliero, M. S. and Ghani, I. (2015). A component based SQL injection vulnerability detection tool. In *Software Engineering Conference (MySEC), 2015 9th Malaysian*. December. Kuala Lumpur, Malaysia: IEEE, 224–229.

Alsaleh, M., Alomar, N., Alshreef, M., Alarifi, A. and Al-Salman, A. (2017). Performance-Based Comparative Assessment of Open Source Web Vulnerability Scanners. *Security and Communication Networks*. 2017.

Alssir, F. and Ahmed, M. (2012). Web security testing approaches: comparison framework. In *Proceedings of the 2011 2nd International Congress on Computer Applications and Computational Science*. September. Berlin, Heidelberg: Springer, 163–169.

Andrew, K. (2009). *Peruggia*. Retrievable at https://sourceforge.net/projects/peruggia/.

Antunes, N., Laranjeiro, N., Vieira, M. and Madeira, H. (2009). Effective detection of SQL/XPath injection vulnerabilities in web services. In *Services Computing, 2009. SCC'09. IEEE International Conference on*. September. Bangalore, India: IEEE, 260–267.

Antunes, N. and Vieira, M. (2009a). Comparing the effectiveness of penetration testing and static code analysis on the detection of sql injection vulnerabilities in web services. In *Dependable Computing, 2009. PRDC'09. 15th IEEE Pacific Rim International Symposium on*. November. Shanghai, China: IEEE, 301–306.

Antunes, N. and Vieira, M. (2009b). Detecting SQL injection vulnerabilities in web services. In *Dependable Computing, 2009. LADC'09. Fourth Latin-American Symposium on*. September. Joao Pessoa, Brazil: IEEE, 17–24.

Antunes, N. and Vieira, M. (2010). Benchmarking vulnerability detection tools for web services. In *Web Services (ICWS), 2010 IEEE International Conference on*. July. Miami, FL, USA: IEEE, 203–210.

Antunes, N. and Vieira, M. (2011). Enhancing penetration testing with attack signatures and interface monitoring for the detection of injection vulnerabilities in web services. In *Services Computing (SCC), 2011 IEEE International Conference on*. July. Washington, DC, USA: IEEE, 104–111.

Antunes, N. and Vieira, M. (2012). Defending against web application vulnerabilities. *Computer*. 45(2), 66–72.

Antunes, N. and Vieira, M. (2014). Penetration testing for web services. *Computer*. 47(2), 30–36.

Antunes, N. and Vieira, M. (2017). Designing vulnerability testing tools for web services: approach, components, and tools. *International Journal of Information Security*. 16(4), 435–457.

Appelt, D., Nguyen, C. D., Briand, L. C. and Alshahwan, N. (2014). Automated testing for SQL injection vulnerabilities: an input mutation approach. In *Proceedings of the 2014 International Symposium on Software Testing and Analysis*. July. San Jose, CA, USA: ACM, 259–269.

Ast, P., Kapfenberger, M. and Hauswiesner, S. (2008). Crawler approaches and technology. *[online]. Graz University of Technology, Styria, Austria, 2008*.

Auronen, L. (2002). Tool-based approach to assessing web application security. *Helsinki University of Technology*. 11, 12–13.

Avancini, A. and Ceccato, M. (2012). Towards a Security Oracle Based on Tree Kernel Methods. *JIMSE 2012*, 1.

Avancini, A. and Ceccato, M. (2013). Comparison and integration of genetic algorithms and dynamic symbolic execution for security testing of cross-site scripting vulnerabilities. *Information and Software Technology*. 55(12), 2209–2222.

Avramescu, G., Bucicoiu, M., Rosner, D. and Tapus, N. (2013). Guidelines for discovering and improving application security. In *Control Systems and Computer Science (CSCS), 2013 19th International Conference on*. May. Bucharest, Romania: IEEE, 560–565.

Awang, N. F. and Manaf, A. A. (2013). Detecting Vulnerabilities in Web Applications Using Automated Black Box and Manual Penetration Testing. In *Advances in Security of Information and Communication Networks*. (pp. 230–239). Springer.

Balduzzi, M., Egele, M., Kirda, E., Balzarotti, D. and Kruegel, C. (2010). A solution for the automated detection of clickjacking attacks. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*. April. Beijing, China: ACM, 135–144.

Balduzzi, M., Gimenez, C. T., Balzarotti, D. and Kirda, E. (2011). Automated Discovery of Parameter Pollution Vulnerabilities in Web Applications. In *NDSS*. February. San Diego, California.

Baral, P. (2011). Web application scanners: a review of related articles [Essay]. *IEEE Potentials*. 30(2), 10–14.

Barbosa, L. and Freire, J. (2007). An adaptive crawler for locating hidden-web entry points. In *Proceedings of the 16th international conference on World Wide Web*. May. Banff, Alberta, Canada: ACM, 441–450.

Bathia, P., Beerelli, B. R. and Laverdière, M.-A. (2011). Assisting programmers resolving vulnerabilities in Java web applications. *Advanced Computing*, 268–279.

Bau, J., Bursztein, E., Gupta, D. and Mitchell, J. (2010). State of the art: Automated black-box web application vulnerability testing. In *Security and Privacy (SP), 2010 IEEE Symposium on*. May. Berkeley/Oakland, CA, USA: IEEE, 332–345.

Bazzoli, E., Criscione, C., Maggi, F. and Zanero, S. (2014). XSS Peeker: A Systematic Analysis of Cross-site Scripting Vulnerability Scanners. *arXiv preprint arXiv:1410.4207*.

Ben Jaballah, W. and Kheir, N. (2016). A Grey-Box Approach for Detecting Malicious User Interactions in Web Applications. In *Proceedings of the 2016 International Workshop on Managing Insider Security Threats*. October. Vienna, Austria: ACM, 1–12.

Benedikt, M., Freire, J. and Godefroid, P. (2002). VeriWeb: Automatically testing dynamic web sites. In *In Proceedings of 11th International World Wide Web Conference (WW W'2002*. January. US: Citeseer.

Benjamin, K., Bochmann, G. v., Jourdan, G.-V. and Onut, I.-V. (2010). Some modeling challenges when testing rich internet applications for security. In *Software Testing, Verification, and Validation Workshops (ICSTW), 2010 Third International Conference on*. April. Montreal, QC, Canada: IEEE, 403–409.

Bennetts, S. (2013). Owasp zed attack proxy. *AppSec USA*.

Bergholz, A. and Childlovskii, B. (2003). Crawling for domain-specific hidden web resources. In *Web Information Systems Engineering, 2003. WISE 2003. Proceedings of the Fourth International Conference on*. Dec. Rome, Italy, Italy: IEEE, 125–133.

Berners-Lee, T. (1989). *Tim berners-lee*. Retrieved from Internet Pioneers website: http://www. ibiblio. org/pioneers/lee. html.

Biggs, N. (2002). *Discrete Mathematics*. OUP Oxford. ISBN 978-0-19-850717-8. Google-Books-ID: Mj9gzZMrXDIC.

Black, P. E. (2011). Counting bugs is harder than you think. In *Source Code Analysis and Manipulation (SCAM), 2011 11th IEEE International Working Conference on*. September. Williamsburg, VA: IEEE, 1–9.

Botella, J., Legeard, B., Peureux, F. and Vernotte, A. (2014). Risk-based vulnerability testing using security test patterns. In *International Symposium On Leveraging Applications of Formal Methods, Verification and Validation*. October. Imperial, Corfu, Greece: Springer, 337–352.

Bozic, J., Simos, D. E. and Wotawa, F. (2014). Attack pattern-based combinatorial testing. In *Proceedings of the 9th International Workshop on Automation of Software Test*. Aug. ancouver, BC, Canada: ACM, 1–7.

Bozic, J. and Wotawa, F. (2013). XSS pattern for attack modeling in testing. In *Proceedings of the 8th International Workshop on Automation of Software Test*. May. San Francisco, CA, USA: IEEE Press, 71–74.

Browne, P. S. (1972). Computer security: a survey. *ACM SIGMIS Database: the DATABASE for Advances in Information Systems*. 4(3), 1–12.

Cardwell, K. (2016). *Building Virtual Pentesting Labs for Advanced Penetration Testing*. Packt Publishing Ltd. ISBN 978-1-78588-495-5. Google-Books-ID: SKbWDQAAQBAJ.

Charikar, M. S. (2002). Similarity Estimation Techniques from Rounding Algorithms. In *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*. STOC '02. May. New York, NY, USA: ACM. ISBN 978-1-58113-495-7, 380–388. doi:10.1145/509907.509965. Retrievable at http://doi.acm.org/10.1145/509907.509965.

Chell, D., Erasmus, T., Colley, S. and Whitehouse, O. (2015). *The mobile application Hacker's handbook*. John Wiley & Sons.

Chen, J.-M. (2013). A Crawler Guard for Quickly Blocking Unauthorized Web Robot. In *Cyberspace Safety and Security*. (pp. 1–13). Springer.

Chen, J.-M. and Wu, C.-L. (2010). An automated vulnerability scanner for injection attack based on injection point. In *Computer Symposium (ICS), 2010 International*. December. Tainan, Taiwan: IEEE, 113–118.

Choudhary, S., Dincturk, M. E., Bochmann, G. V., Jourdan, G.-V., Onut, I. V. and Ionescu, P. (2012). Solving some modeling challenges when testing rich internet applications for security. In *Software Testing, Verification and Validation (ICST), 2012 IEEE Fifth International Conference on*. April. Montreal, QC, Canada: IEEE, 850–857.

Choudhary, S., Dincturk, M. E., Mirtaheri, S. M., von Bochmann, G., Jourdan, G.-V. and Onut, I.-V. (2014). Model-Based Rich Internet Applications Crawling:" Menu" and" Probability" Models. *J. Web Eng.* 13(3&4), 243–262.

Consortium, W. W. W. (1995). *W3C HTML home page*.

Dao, T.-B. and Shibayama, E. (2009). Idea: Automatic Security Testing for Web Applications. *Engineering Secure Software and Systems*, 180–184.

Dao, T.-B. and Shibayama, E. (2010). Coverage Criteria for Automatic Security Testing of Web Applications. In *ICISS*. December. Gandhinagar, India: Springer, 111–124.

Dawes, R. (2011). OWASP WebScarab Project. *Retrieved December*. 16, 2011.

Díaz, G. and Bermejo, J. R. (2013). Static analysis of source code security: Assessment of tools against SAMATE tests. *Information and software technology*. 55(8), 1462–1476.

Deepa, G. and Thilagam, P. S. (2016). Securing web applications from injection and logic vulnerabilities: Approaches and challenges. *Information and Software Technology*. 74, 160–180.

Deepa, G., Thilagam, P. S., Khan, F. A., Praseed, A., Pais, A. R. and Palsetia, N. (2018). Black-box detection of XQuery injection and parameter tampering vulnerabilities in web applications. *International Journal of Information Security*. 17(1), 105–120.

Dessiatnikoff, A., Akrout, R., Alata, E., Kaâniche, M. and Nicomette, V. (2011). A clustering approach for web vulnerabilities detection. In *Dependable Computing (PRDC), 2011 IEEE 17th Pacific Rim International Symposium on*. December. Pasadena, CA, USA: IEEE, 194–203.

Dincturk, M. E., Jourdan, G.-V., Bochmann, G. V. and Onut, I. V. (2014). A model-based approach for crawling rich internet applications. *ACM Transactions on the Web (TWEB)*. 8(3), 19.

Djuric, Z. (2013). A black-box testing tool for detecting SQL injection vulnerabilities. In *Informatics and Applications (ICIA), 2013 Second International Conference on*. September. Lodz, Poland: IEEE, 216–221.

Doupé, A. (2016). *WackoPicko vulnerable website*.

Doupé, A., Cavedon, L., Kruegel, C. and Vigna, G. (2012). Enemy of the State: A State-Aware Black-Box Web Vulnerability Scanner. In *USENIX Security Symposium*, vol. 14. August. Bellevue, WA.

Doupé, A., Cova, M. and Vigna, G. (2010). Why Johnny can't pentest: An analysis of black-box web vulnerability scanners. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. July. Bonn, Germany: Springer, 111–131.

Druin, J. (2012). Mutillidae: Brute Force Page Names using Burp-Suite Intruder. *Retrieved June*. 30, 2013.

Duchene, F., Rawat, S., Richier, J.-L. and Groz, R. (2013). LigRE: Reverse-engineering of control and data flow models for black-box XSS detection. In *Reverse Engineering (WCRE), 2013 20th Working Conference on*. October. Koblenz, Germany: IEEE, 252–261.

Duchene, F., Rawat, S., Richier, J.-L. and Groz, R. (2014). KameleonFuzz: evolutionary fuzzing for black-box XSS detection. In *Proceedings of the 4th ACM conference on Data and application security and privacy*. March. San Antonio, Texas, USA: ACM, 37–48.

Dukes, L., Yuan, X. and Akowuah, F. (2013). A case study on web application security testing with tools and manual testing. In *Southeastcon, 2013 Proceedings of IEEE*. April. Jacksonville, FL, USA: IEEE, 1–6.

Edmundson, A., Holtkamp, B., Rivera, E., Finifter, M., Mettler, A. and Wagner, D. (2013). An empirical study on the effectiveness of security code review. In *International Symposium on Engineering Secure Software and Systems*. February. Paris, France: Springer, 197–212.

Engebretson, P. (2013). *The Basics of Hacking and Penetration Testing: Ethical Hacking and Penetration Testing Made Easy*. Elsevier. ISBN 978-0-12-411641-2. Google-Books-ID: 69dEUBJKMiYC.

Faheem, M. (2012). Intelligent crawling of Web applications for Web archiving. In *Proceedings of the 21st International Conference on World Wide Web*. July. Berlin, Germany: ACM, 127–132.

Ferrucci, F., Sarro, F., Ronca, D. and Abrahao, S. (2011). A crawljax based approach to exploit traditional accessibility evaluation tools for AJAX applications. In *Information technology and innovation trends in organizations*. (pp. 255–262). Springer.

Fong, E., Gaucher, R., Okun, V., Black, P. E. and Dalci, E. (2008). Building a test suite for web application scanners. In *Hawaii International Conference on System Sciences, Proceedings of the 41st Annual*. January. Waikoloa, HI, USA: IEEE, 478–478.

Fong, E. and Okun, V. (2007). Web application scanners: definitions and functions. In *System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on*. January. Waikoloa, HI, USA: IEEE, 280b–280b.

Fonseca, J., Seixas, N., Vieira, M. and Madeira, H. (2014a). Analysis of field data on web security vulnerabilities. *IEEE transactions on dependable and secure computing*. 11(2), 89–100.

Fonseca, J., Vieira, M. and Madeira, H. (2007). Testing and comparing web vulnerability scanning tools for SQL injection and XSS attacks. In *Dependable Computing, 2007. PRDC 2007. 13th Pacific Rim International Symposium on*. December. Melbourne, Qld., Australia: IEEE, 365–372.

Fonseca, J., Vieira, M. and Madeira, H. (2009). Vulnerability & attack injection for web applications. In *Dependable Systems & Networks, 2009. DSN'09. IEEE/IFIP International Conference on*. June. Lisbon, Portugal: IEEE, 93–102.

Fonseca, J., Vieira, M. and Madeira, H. (2014b). Evaluation of web security mechanisms using vulnerability & attack injection. *IEEE Transactions on Dependable and Secure Computing*. 11(5), 440–453.

Foundation, C. S. a. P. (2019). *Vulnerable Java based Web Application. Contribute to CSPF-Founder/JavaVulnerableLab development by creating an account on GitHub*. Retrievable at https://github.com/CSPF-Founder/JavaVulnerableLab, original-date: 2015-01-07T11:11:46Z.

Fraternali, P., Rossi, G. and Sánchez-Figueroa, F. (2010). Rich internet applications. *IEEE Internet Computing*. 14(3), 9–12.

Frauenfelder, M. (2004). Sir Tim Berners-Lee: He created the Web. Now he's working on Internet 2.0. *Technology Review (October)*, 40–45.

Fu, X. and Qian, K. (2008). SAFELI: SQL injection scanner using symbolic execution. In *Proceedings of the 2008 workshop on Testing, analysis, and verification of web services and applications*. July. Seattle, Washington: ACM, 34–39.

Fung, A. P., Wang, T., Cheung, K. W. and Wong, T. Y. (2014). Scanning of real-world web applications for parameter tampering vulnerabilities. In *Proceedings of the 9th ACM symposium on Information, computer and communications security*. June. Kyoto, Japan: ACM, 341–352.

Galán, E., Alcaide, A., Orfila, A. and Blasco, J. (2010). A multi-agent scanner to detect stored-XSS vulnerabilities. In *Internet Technology and Secured Transactions (ICITST), 2010 International Conference for*. November. London, UK: IEEE, 1–6.

Ghafari, M., Shoja, H. and Amirani, M. Y. (2012). Detection and prevention of data manipulation from client side in web applications. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on*. June. Liverpool, UK: IEEE, 1132–1136.

Giralte, L. C., Conde, C., De Diego, I. M. and Cabello, E. (2013). Detecting denial of service by modelling web-server behaviour. *Computers & Electrical Engineering*. 39(7), 2252–2262.

Godefroid, P., Kiezun, A. and Levin, M. Y. (2008). Grammar-based Whitebox Fuzzing. In *Proceedings of the 29th ACM SIGPLAN Conference on Programming Language Design and Implementation*. PLDI '08. June. New York, NY, USA: ACM. ISBN 978-1-59593-860-2, 206–215. doi:10.1145/1375581.1375607. Retrievable at http://doi.acm.org/10.1145/1375581.1375607.

Godefroid, P., Levin, M. Y. and Molnar, D. (2012). SAGE: Whitebox Fuzzing for Security Testing. *Commun. ACM*. 55(3), 40–44. ISSN 0001-0782. doi:10.1145/2093548.2093564. Retrievable at http://doi.acm.org/10.1145/2093548.2093564.

Gol, D. and Shah, N. (2015). Detection of web appication vulnerability based on RUP model. In *Recent Advances in Electronics & Computer Engineering (RAECE), 2015 National Conference on*. February. Roorkee, India: IEEE, 96–100.

Gross, J. L., Yellen, J. and Zhang, P. (2013). *Handbook of Graph Theory, Second Edition*. CRC Press. ISBN 978-1-4398-8018-0. Google-Books-ID: cntcAgAAQBAJ.

Grossman, J. (2007). *XSS Attacks: Cross-site scripting exploits and defense*. Syngress.

Halfond, W. G., Choudhary, S. R. and Orso, A. (2011). Improving penetration testing through static and dynamic analysis. *Software Testing, Verification and Reliability*. 21(3), 195–214.

Hasegawa, Y. (2005). UTF-7 XSS cheat sheet. *retrieved at*, 2.

Hazel, J. J., Valarmathie, P. and Saravanan, R. (2015). Guarding web application with multi-Angled attack detection. In *Soft-Computing and Networks Security (ICSNS), 2015 International Conference on*. Feb. Coimbatore, India: IEEE, 1–4.

He, Y., Xin, D., Ganti, V., Rajaraman, S. and Shah, N. (2013). Crawling deep web entity pages. In *Proceedings of the sixth ACM international conference on Web search and data mining*. February. Rome, Italy: ACM, 355–364.

Heiderich, M., Schwenk, J., Frosch, T., Magazinius, J. and Yang, E. Z. (2013). mxss attacks: Attacking well-secured web-applications by using innerhtml mutations. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. November. Berlin, Germany: ACM, 777–788.

Holm, H., Ekstedt, M. and Sommestad, T. (2013). Effort estimates on web application vulnerability discovery. In *System Sciences (HICSS), 2013 46th Hawaii International Conference on*. March. Wailea, Maui, HI, USA: IEEE, 5029–5038.

Holm, H., Sommestad, T., Almroth, J. and Persson, M. (2011). A quantitative evaluation of vulnerability scanning. *Information Management & Computer Security*. 19(4), 231–247.

Huang, J. C. (2007). *Path-oriented program analysis*. Cambridge University Press.

Huang, Y.-W., Huang, S.-K., Lin, T.-P. and Tsai, C.-H. (2003). Web application security assessment by fault injection and behavior monitoring. In *Proceedings of the 12th international conference on World Wide Web*. May. Budapest, Hungary: ACM, 148–159.

Huang, Y.-W. and Lee, D. T. (2005). Web Application Security—Past, Present, and Future. In *Computer Security in the 21st Century*. (pp. 183–227). Springer.

Huang, Y.-W., Tsai, C.-H., Lee, D. T. and Kuo, S.-Y. (2004a). Non-detrimental web application security scanning. In *Software Reliability Engineering, 2004. ISSRE 2004. 15th International Symposium on*. November. Saint-Malo, Bretagne, France: IEEE, 219–230.

Huang, Y.-W., Tsai, C.-H., Lin, T.-P., Huang, S.-K., Lee, D. T. and Kuo, S.-Y. (2005). A testing framework for Web application security assessment. *Computer Networks*. 48(5), 739–761.

Huang, Y.-W., Yu, F., Hang, C., Tsai, C.-H., Lee, D.-T. and Kuo, S.-Y. (2004b). Securing web application code by static analysis and runtime protection. In *Proceedings of the 13th international conference on World Wide Web*. May. New York, NY, USA: ACM, 40–52.

Huiyao, A., Yang, S., Tao, Y., Hui, L., Peng, Z. and Jun, Z. (2014). A New Architecture of Ajax Web Application Security Crawler with Finite-State Machine. In *Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2014 International Conference on*. Oct. Shanghai, China: IEEE, 112–117.

Hydara, I., Sultan, A. B. M., Zulzalil, H. and Admodisastro, N. (2015). Current state of research on cross-site scripting (XSS)–A systematic literature review. *Information and Software Technology*. 58, 170–186.

Imperva (2017). *The State of Web Application Vulnerabilities in 2017| Imperva*. Retrievable at https://www.imperva.com/blog/2017/12/the-state-of-web-application-vulnerabilities-in-2017/.

Ipeirotis, P. G., Agichtein, E., Jain, P. and Gravano, L. (2006). To search or to crawl?: towards a query optimizer for text-centric tasks. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*. June. Chicago, IL, USA: ACM, 265–276.

ISO, I. (2010). IEEE, Systems and Software Engineering–Vocabulary. *IEEE computer society, Piscataway, NJ*.

J, B. (1999). *Bach: Risk and requirements-based testing - Google Scholar*. Retrievable at https://scholar.google.com/scholar_lookup?title=Risk%20and%

20Requirements-Based%20Testing&author=J..%20Bach&journal=Computer&volume=32&issue=6&pages=113-114&publication_year=1999.

Jason, S. and Dan, T. (2006). *Paros Proxy | TestingSecurity.com*. Retrievable at http://www.testingsecurity.com/paros_proxy.

Jovanovic, N., Kruegel, C. and Kirda, E. (2006). Pixy: A static analysis tool for detecting web application vulnerabilities. In *Security and Privacy, 2006 IEEE Symposium on*. June. Berkeley/Oakland, CA, USA: IEEE, 1–6.

Kals, S., Kirda, E., Kruegel, C. and Jovanovic, N. (2006). Secubat: a web vulnerability scanner. In *Proceedings of the 15th international conference on World Wide Web*. May. Edinburgh, Scotland: ACM, 247–256.

Kaushik, M. and Singh, G. (2013). A STUDY OF TESTING TECHNIQUES FOR WEB APPLICATIONS.

Khoury, N., Zavarsky, P., Lindskog, D. and Ruhl, R. (2011a). An analysis of black-box web application security scanners against stored SQL injection. In *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on*. October. Boston, MA, USA: IEEE, 1095–1101.

Khoury, N., Zavarsky, P., Lindskog, D. and Ruhl, R. (2011b). Testing and assessing web vulnerability scanners for persistent SQL injection attacks. In *Proceedings of the First International Workshop on Security and Privacy Preserving in e-Societies*. June. Beirut, Lebanon: ACM, 12–18.

Klein, A. (2002). Cross site scripting explained. *Sanctum White Paper*, 1–7.

Klein, A. (2005). DOM based cross site scripting or XSS of the third kind. *http://www. webappsec. org/projects/articles/071105. shtml*.

Kosuga, Y., Kono, K., Hanaoka, M., Hishiyama, M. and Takahama, Y. (2007). Sania: Syntactic and semantic analysis for automated testing against sql injection. In *Computer Security Applications Conference, 2007. ACSAC 2007. Twenty-Third Annual*. December. Miami Beach, FL, United States: IEEE, 107–117.

Kozina, M., Golub, M. and Groš, S. (2009). A method for identifying Web applications. *Int. J. Inf. Secur.* 8(6), 455. ISSN 1615-5262, 1615-5270. doi:

10.1007/s10207-009-0092-3. Retrievable at https://link.springer.com/article/10.1007/s10207-009-0092-3.

Lebeau, F., Legeard, B., Peureux, F. and Vernotte, A. (2013). Model-based vulnerability testing for web applications. In *Software Testing, Verification and Validation Workshops (ICSTW), 2013 IEEE Sixth International Conference on*. March. Luxembourg, Luxembourg: IEEE, 445–452.

Legeard, B., Gupta, S., Schoch, J.-P. and Wilkinson, J. S. (2017). *Software Testing Industrialization: A Model-Based Testing Perspective*. Wiley. ISBN 978-0-470-93951-2.

Lei, L., Jing, X., Minglei, L. and Jufeng, Y. (2013). A Dynamic SQL injection vulnerability test case generation model based on the multiple phases detection approach. In *Computer Software and Applications Conference (COMPSAC), 2013 IEEE 37th Annual*. July. Kyoto, Japan: IEEE, 256–261.

Levenshtein, V. I. (1965). Binary codes capable of correcting deletions, insertions and reversals. *Doklady. Akademii Nauk SSSR*. 163(4), 845–848.

Li, N., Xie, T., Jin, M. and Liu, C. (2010). Perturbation-based user-input-validation testing of web applications. *Journal of Systems and Software*. 83(11), 2263–2274.

Li, X., Si, X. and Xue, Y. (2014). Automated black-box detection of access control vulnerabilities in web applications. In *Proceedings of the 4th ACM conference on Data and application security and privacy*. March. San Antonio, Texas, USA: ACM, 49–60.

Li, X., Yan, W. and Xue, Y. (2012). SENTINEL: securing database from logic flaws in web applications. In *Proceedings of the second ACM conference on Data and Application Security and Privacy*. September. Lyon, France: ACM, 25–36.

Liu, B., Shi, L., Cai, Z. and Li, M. (2012). Software vulnerability discovery techniques: A survey. In *Multimedia Information Networking and Security (MINES), 2012 Fourth International Conference on*. November. Nanjing, China: IEEE, 152–156.

Liu, L., Su, G., Xu, J., Zhang, B., Kang, J., Xu, S., Li, P. and Si, G. (2017). An Inferential Metamorphic Testing Approach to Reduce False Positives in SQLIV Penetration Test. In *Computer Software and Applications Conference (COMPSAC), 2017 IEEE 41st Annual*, vol. 1. July. Turin, Italy: IEEE, 675–680.

Livshits, V. B. (2004). Findings security errors in Java applications using lightweight static analysis. In *Computer Security Applications Conference, Tucson, AZ*. July. Baltimore, MD, 2.

Loh, P. K. K. and Subramanian, D. (2010). Fuzzy classification metrics for scanner assessment and vulnerability reporting. *IEEE Transactions on Information Forensics and security*. 5(4), 613–624.

Lounis, O., Guermeche, S. E. B., Saoudi, L. and Benaicha, S. E. (2014). A new algorithm for detecting SQL injection attack in Web application. In *Science and Information Conference (SAI), 2014*. Aug. London, UK: IEEE, 589–594.

Makino, Y. and Klyuev, V. (2015). Evaluation of web vulnerability scanners. In *Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), 2015 IEEE 8th International Conference on*, vol. 1. September. Warsaw, Poland: IEEE, 399–402.

McAllister, S., Kirda, E. and Kruegel, C. (2008). Leveraging user interactions for in-depth testing of web applications. In *Raid*, vol. 8. September. Cambridge, MA, USA: Springer, 191–210.

McCune, R. (2019). *Contribute to raesene/bWAPP development by creating an account on GitHub*. Retrievable at https://github.com/raesene/bWAPP, original-date: 2015-09-17T12:46:24Z.

Medeiros, I., Neves, N. F. and Correia, M. (2014). Automatic detection and correction of web application vulnerabilities using data mining to predict false positives. In *Proceedings of the 23rd international conference on World wide web*. April. Seoul, Korea: ACM, 63–74.

Meier, J. D., Hill, D., Homer, A., Jason, T., Bansode, P., Wall, L., Boucher Jr, R. and Bogawat, A. (2009). Microsoft application architecture guide. *Microsoft Corporation*.

Mesbah, A., Bozdag, E. and Van Deursen, A. (2008). Crawling Ajax by inferring user interface state changes. In *Web Engineering, 2008. ICWE'08. Eighth International Conference on*. July. Yorktown Heights, NJ, USA: IEEE, 122–134.

Monga, M., Paleari, R. and Passerini, E. (2009). A hybrid analysis framework for detecting web application vulnerabilities. In *Proceedings of the 2009 ICSE Workshop* *on Software Engineering for Secure Systems*. May. Vancouver, BC, Canada: IEEE Computer Society, 25–32.

Moonen, L. (2011). *Static Analysis for Software Verification*.

Muñoz, F. R. and Villalba, L. J. G. (2015). Web from preprocessor for crawling. *Multimedia Tools and Applications*. 74(19), 8559–8570.

Myers, G. J., Sandler, C. and Badgett, T. (2011). *The Art of Software Testing*. John Wiley & Sons. ISBN 978-1-118-13315-6. Google-Books-ID: GjyEFPkMCwcC.

Negara, N. and Stroulia, E. (2012). Automated acceptance testing of javascript web applications. In *Reverse Engineering (WCRE), 2012 19th Working Conference on*. Oct. Kingston, ON, Canada: IEEE, 318–322.

Nguyen-Tuong, A., Guarnieri, S., Greene, D., Shirley, J. and Evans, D. (2005). Automatically hardening web applications using precise tainting. *Security and Privacy in the Age of Ubiquitous Computing*, 295–307.

Orchard, D. (2006). *[Editorial Draft] State in Web application design*. Retrievable at https://www.w3.org/2001/tag/doc/state.html.

OWASP, T. (2017). *Application Security Risks-2017, Open Web Application Security Project (OWASP)*.

Palsetia, N., Deepa, G., Khan, F. A., Thilagam, P. S. and Pais, A. R. (2016). Securing native XML database-driven web applications from XQuery injection vulnerabilities. *Journal of Systems and Software*. 122, 93–109.

Parvez, M., Zavarsky, P. and Khoury, N. (2015). Analysis of effectiveness of black-box web application scanners in detection of stored SQL injection and stored XSS vulnerabilities. In *Internet Technology and Secured Transactions (ICITST), 2015 10th International Conference for*. Dec. London, UK: IEEE, 186–191.

Patil, S., Marathe, N. and Padiya, P. (2016). Design of efficient web vulnerability scanner. In *Inventive Computation Technologies (ICICT), International Conference on*, vol. 2. Aug. Coimbatore, India: IEEE, 1–6.

Pellegrino, G., Tschürtz, C., Bodden, E. and Rossow, C. (2015). jÄk: Using Dynamic Analysis to Crawl and Test Modern Web Applications. In *International Workshop on Recent Advances in Intrusion Detection*. November. Kyoto, Japan: Springer, 295–316.

Pennington, W., Grossman, J., Stone, R. and Pazirandeh, S. (2013). *Using fuzzy classification models to perform matching operations in a web application security scanner*.

Pérez, P. M., Filipiak, J. and Sierra, J. M. (2011). LAPSE+ static analysis security software: Vulnerabilities detection in java EE applications. *Future Information Technology*, 148–156.

Raghavan, S. and Garcia-Molina, H. (2000). *Crawling the hidden web*. Technical report. Stanford.

Rahman, T. F. A., Buja, A. G., Abd, K. and Ali, F. M. (2017). SQL Injection Attack Scanner Using Boyer-Moore String Matching Algorithm. *JCP*. 12(2), 183–189.

Razzaq, A., Anwar, Z., Ahmad, H. F., Latif, K. and Munir, F. (2014a). Ontology for attack detection: An intelligent approach to web application security. *computers & security*. 45, 124–146.

Razzaq, A., Latif, K., Ahmad, H. F., Hur, A., Anwar, Z. and Bloodsworth, P. C. (2014b). Semantic security against web application attacks. *Information Sciences*. 254, 19–38.

Reshef, E., El-Hanany, Y., Raanan, G. and Tsarfati, T. (2007). *System for determining web application vulnerabilities*.

Riancho, A. (2013). w3af. *Retrieved June*. 30, 2013.

Rick, H. (2012). *The Penetration Testing Execution Standard*. Retrievable at http://www.pentest-standard.org/index.php/Main_Page.

Rocha, D., Kreutz, D. and Turchetti, R. (2012). A free and extensible tool to detect vulnerabilities in web systems. In *Information Systems and Technologies (CISTI), 2012 7th Iberian Conference on*. June. Madrid, Spain: IEEE, 1–6.

Rocha, T. S. and Souto, E. (2014). ETSSDetector: a tool to automatically detect Cross-Site Scripting vulnerabilities. In *Network Computing and Applications (NCA), 2014 IEEE 13th International Symposium on*. Aug. Cambridge, MA, USA: IEEE, 306–309.

Ruse, M. E. (2013). Model checking techniques for vulnerability analysis of Web applications.

Sadeghian, A., Zamani, M. and Manaf, A. A. (2013). A Taxonomy of SQL Injection Detection and Prevention Techniques. In *2013 International Conference on Informatics and Creative Multimedia*. September. Kuala Lumpur, Malaysia, 53–56. doi:10.1109/ICICM.2013.18.

Salas, M. I. P. and Martins, E. (2014). Security testing methodology for vulnerabilities detection of xss in web services and ws-security. *Electronic Notes in Theoretical Computer Science*. 302, 133–154.

Saleh, A. Z. M., Rozali, N. A., Buja, A. G., Jalil, K. A., Ali, F. H. M. and Rahman, T. F. A. (2015). A method for web application vulnerabilities detection by using boyer-moore string matching algorithm. *Procedia Computer Science*. 72, 112–121.

Schanes, C., Hübler, A., Fankhauser, F. and Grechenig, T. (2013). Generic Approach for Security Error Detection Based on Learned System Behavior Models for Automated Security Tests. In *Software Testing, Verification and Validation Workshops (ICSTW), 2013 IEEE Sixth International Conference on*. March. Luxembourg, Luxembourg: IEEE, 453–460.

Scott, D. and Sharp, R. (2002). Abstracting Application-level Web Security. In *Proceedings of the 11th International Conference on World Wide Web*. WWW '02. May. New York, NY, USA: ACM. ISBN 978-1-58113-449-0, 396–407. doi:10.1145/511446.511498. Retrievable at http://doi.acm.org/10.1145/511446.511498.

Security Standards Council (2017). *Information Supplement: Penetration Testing Guidance*. Retrievable at https://www.pcisecuritystandards.org/documents/Penetration-Testing-Guidance-v1_1.pdf.

Shahriar, H. and Zulkernine, M. (2009). Automatic testing of program security vulnerabilities. In *Computer Software and Applications Conference, 2009. COMPSAC'09. 33rd Annual IEEE International*, vol. 2. July. Seattle, WA, USA: IEEE, 550–555.

Shang, K. and Hossen, Z. (2013). Applying fuzzy logic to risk assessment and decision-making. *Casualty Actuarial Society, Canadian Institute of Actuaries, Society of Actuaries*, 1–59.

Shay, C. (2017). *The Prices vs. Features of Web Application Vulnerability Scanners*. Retrievable at http://www.sectoolmarket.com/price-and-feature-comparison-of-web-application-scanners-unified-list.html.

Shklar, L. and Rosen, R. (2009). *Web Application Architecture: Principles, Protocols and Practices*. Wiley. ISBN 978-0-470-51860-1. Google-Books-ID: gcHJQwAACAAJ.

Sima, C., Kelly, R., Millar, S., Raboud, R., Sullivan, B., Sullivan, J. and Tillery, D. (2006). *Interactive web crawling*. Retrievable at http://www.google.com/patents/US20060282494, u.S. Classification 709/200; International Classification H04L9/00, H04L29/06, H04L9/32; Cooperative Classification H04L63/20, H04L63/12; European Classification H04L63/12, H04L63/20.

Sima, C., Millar, S., Kelly, R., Sullivan, B., Sullivan, G. and Tillery, D. (2010). *Integrated crawling and auditing of web applications and web content*.

Simos, D. E., Kleine, K., Ghandehari, L. S. G., Garn, B. and Lei, Y. (2016). A Combinatorial Approach to Analyzing Cross-Site Scripting (XSS) Vulnerabilities in Web Application Security Testing. In *IFIP International Conference on Testing Software and Systems*. October. Graz, Austria: Springer, 70–85.

Singh, A. K. and Roy, S. (2012). A network based vulnerability scanner for detecting SQLI attacks in web applications. In *Recent Advances in Information Technology (RAIT), 2012 1st International Conference on*. March. Dhanbad, India: IEEE, 585–590.

Singh, A. K. and Tiwari, L. (2011). Vulnerability Assessment and penetration Testing. In *National Conference on Information & Communication Technology (NCICT–2011), ISBN*. September. Lyon, France, 978–93.

Spett, K. (2005). Cross-site scripting. *SPI Labs*. 1, 1–20.

Su, Z. and Wassermann, G. (2006). The essence of command injection attacks in web applications. In *ACM SIGPLAN Notices*, vol. 41. January. Charleston, South Carolina, USA: ACM, 372–382.

Subramanian, D., Le, H. T., Loh, P. K. K. and Premkumar, A. B. (2010). Quantitative Evaluation of Related Web-Based Vulnerabilities. In *Secure Software Integration* *and Reliability Improvement Companion (SSIRI-C), 2010 Fourth International Conference on*. June. Singapore, Singapore: IEEE, 118–125.

Surribas, N. (2006). Wapiti, web application vulnerability scanner/security auditor. *URL: http://wapiti. sourceforge. net*.

Suto, L. (2007). Analyzing the effectiveness and coverage of Web application security scanners. *San Francisco, October*.

Suto, L. (2010). Analyzing the accuracy and time costs of web application security scanners. *San Francisco, February*.

Sutton, M., Greene, A. and Amini, P. (2007). *Fuzzing: Brute Force Vulnerability Discovery*. Pearson Education. ISBN 978-0-321-68085-3. Google-Books-ID: DPAwwn7QDy8C.

Takanen, A., Demott, J. D. and Miller, C. (2008). *Fuzzing for software security testing and quality assurance*. Artech House.

Tatli, E. s. and Urgun, B. (2017). WIVET—Benchmarking Coverage Qualities of Web Crawlers. *The Computer Journal*. 60(4), 555–572.

Tetskyi, A., Kharchenko, V. and Uzun, D. (2018). Neural networks based choice of tools for penetration testing of web applications. In *2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT)*. May. Kiev, Ukraine: IEEE.

Thomé, J., Gorla, A. and Zeller, A. (2014). Search-based security testing of web applications. In *Proceedings of the 7th International Workshop on Search-Based Software Testing*. November. Williamsburg, VI, USA: ACM, 5–14.

Tian-yang, G., Yin-Sheng, S. and You-yuan, F. (2010). Research on software security testing. *World Academy of science, engineering and Technology*. 70, 647–651.

Tripp, O., Pistoia, M., Cousot, P., Cousot, R. and Guarnieri, S. (2013). Andromeda: Accurate and Scalable Security Analysis of Web Applications. In *FASE*, vol. 7793. March. Rome, Italy: Springer, 210–225.

Tung, Y.-H., Tseng, S.-S., Shih, J.-F. and Shan, H.-L. (2013). A cost-effective approach to evaluating security vulnerability scanner. In *Network Operations and Management Symposium (APNOMS), 2013 15th Asia-Pacific*. September. Hiroshima, Japan: IEEE, 1–3.

Tung, Y.-H., Tseng, S.-S., Shih, J.-F. and Shan, H.-L. (2014). W-VST: A Testbed for Evaluating Web Vulnerability Scanner. In *Quality Software (QSIC), 2014 14th International Conference on*. October. Dallas, TX, USA: IEEE, 228–233.

ĐURIĆ, Z. (2014). WAPTT-Web Application Penetration Testing Tool. *Advances in Electrical and Computer Engineering*. 14(1).

Utting, M. and Legeard, B. (2010). *Practical Model-Based Testing: A Tools Approach*. Elsevier. ISBN 978-0-08-046648-4.

Valiente, G. (2013). *Algorithms on Trees and Graphs*. Springer Science & Business Media. ISBN 978-3-662-04921-1. Google-Books-ID: rOmpCAAAQBAJ.

van der Loo, F. (2011). *Comparison of penetration testing tools for web applications*. PhD Thesis. Master's thesis, University of Radboud, Netherlands.

van Eyk, E., van Leeuwen, W., Larson, M. A. and Hermans, F. (2014). *Performance of near-duplicate detection algorithms for Crawljax*. Citeseer.

Van Rijsbergen, C. J. (1979). Information retrieval. dept. of computer science, university of glasgow. *URL: citeseer. ist. psu. edu/vanrijsbergen79information. html*. 14.

Veracode (2015). *Four Out of Five Applications Written in Web Scripting Languages Fail OWASP Top 10 Upon First Assessment*. Retrievable at http://www.veracode.com/.

Vernotte, A., Dadeau, F., Lebeau, F., Legeard, B., Peureux, F. and Piat, F. (2014). Efficient detection of multi-step cross-site scripting vulnerabilities. In *International Conference on Information Systems Security*. December. Hyderabad, India: Springer, 358–377.

Vieira, M., Antunes, N. and Madeira, H. (2009). Using web security scanners to detect vulnerabilities in web services. In *Dependable Systems & Networks, 2009. DSN'09. IEEE/IFIP International Conference on*. June. Lisbon, Portugal: IEEE, 566–571.

Vithanage, N. M. and Jeyamohan, N. (2016). WebGuardia-an integrated penetration testing system to detect web application vulnerabilities. In *Wireless Communications, Signal Processing and Networking (WiSPNET), International Conference on*. March. Chennai, India: IEEE, 221–227.

W3C (2018). *Browser Statistics*. Retrievable at https://www.w3schools.com/browsers/default.asp.

Wang, S., Gong, Y., Chen, G., Sun, Q. and Yang, F. (2013). Service vulnerability scanning based on service-oriented architecture in Web service environments. *Journal of Systems Architecture*. 59(9), 731–739.

Wang, X., Wang, L., Wei, G., Zhang, D. and Yang, Y. (2010). Hidden web crawling for SQL injection detection. In *Broadband Network and Multimedia Technology (IC-BNMT), 2010 3rd IEEE International Conference on*. January. Beijing, China: IEEE, 14–18.

Wassermann, G. and Su, Z. (2008). Static detection of cross-site scripting vulnerabilities. In *Proceedings of the 30th international conference on Software engineering*. May. Leipzig, Germany: ACM, 171–180.

Wei, K., Muthuprasanna, M. and Kothari, S. (2006). Preventing SQL injection attacks in stored procedures. In *Software Engineering Conference, 2006. Australian*. April. Sydney, NSW, Australia: IEEE, 8–pp.

Weidman, G. (2014). *Penetration testing: a hands-on introduction to hacking*. No Starch Press.

WhiteHat (2017). *2017 Application Security Statistics Report - WhiteHat Security*. Retrievable at https://www.whitehatsec.com/resources-category/premium-content/web-application-stats-report-2017/.

Xiong, P., Stepien, B. and Peyton, L. (2009). Model-based penetration test framework for web applications using TTCN-3. *E-Technologies: Innovation in an Open World*, 141–154.

Yang, J.-M., Cai, R., Wang, Y., Zhu, J., Zhang, L. and Ma, W.-Y. (2009). Incorporating site-level knowledge to extract structured data from web forums. In *Proceedings of the 18th international conference on World wide web*. April. Madrid, Spain: ACM, 181–190.

Yang, X., Chen, Y., Zhang, W. and Zhang, S. (2011). Exploring injection prevention technologies for security-aware distributed collaborative manufacturing on the Semantic Web. *The International Journal of Advanced Manufacturing Technology*. 54(9), 1167–1177.

Yeo, J. (2013). Using penetration testing to enhance your company's security. *Computer Fraud & Security*. 2013(4), 17–20.

Zalewski, M., Heinen, N. and Roschke, S. (2011). *Skipfish-web application security scanner*. URL: http://code. google. com/p/skipfish/(visited on 06/03/2012).