UNIVERSITI PUTRA MALAYSIA

AN APPROACH TO IMPROVE DETECTING AND REMOVING CROSSSITE SCRIPTING VULNERABILITIES IN WEB APPLICATIONS

ISATOU HYDARA

FSKTM 2015 5
AN APPROACH TO IMPROVE DETECTING AND REMOVING CROSS-SITE SCRIPTING VULNERABILITIES IN WEB APPLICATIONS

By

ISATOU HYDARA

Thesis Submitted to the School of Graduate Studies, Universiti Putra Malaysia, in Fulfillment of the Requirements for the Degree of Master of Science

May, 2015
All material contained within the thesis, including without limitation text, logos, icons, photographs and all other artwork, is copyright material of Universiti Putra Malaysia unless otherwise stated. Use may be made of any material contained within the thesis for non-commercial purposes from the copyright holder. Commercial use of material may only be made with the express, prior, written permission of Universiti Putra Malaysia.

Copyright © Universiti Putra Malaysia
DEDICATION

This work is dedicated to my beloved family, my parents, my brother and my sister, for their endless love, prayer and support, and everyone else who has provided me guidance, support and encouragement.
Abstract of thesis presented to the Senate of Universiti Putra Malaysia in fulfillment of the requirement for the degree of Master of Science

AN APPROACH TO IMPROVE DETECTING AND REMOVING CROSS-SITE SCRIPTING VULNERABILITIES IN WEB APPLICATIONS

By

ISATOU HYDARA

May, 2015

Chairman: Abu Bakar Md Sultan, PhD
Faculty: Computer Science and Information Technology

Cross-Site Scripting (XSS) vulnerabilities are among the most common and most serious security vulnerabilities in Web applications. They occur due to lack of proper verification of the user inputs, which enables hackers to inject and execute malicious scripts in the Hyper Text Markup Language (HTML) pages of an application. The presence of XSS vulnerabilities in applications source codes enables XSS attacks to take place. Successful XSS attacks can lead to serious security violations such as account hijacking, denial of service, cookie theft, and web content manipulations.

XSS vulnerabilities are easy to exploit but difficult to eliminate. Many solutions have been proposed for their mitigation, however, the problem still persists. Many web applications are vulnerable to XSS and are attacked frequently. Most of the previously proposed approaches focused on preventing and detecting XSS attacks during runtime, after vulnerable applications are already deployed. Few approaches have focused on removing the vulnerabilities from the source codes before deployment of the applications. The presence of XSS vulnerabilities in an application makes it easy to attack successfully during runtime. Also most of these approaches only focused on the detection of type I and II XSS but not on type III XSS, which is more difficult to eliminate.

In this research, an approach has been proposed that explores the combination of genetic algorithms with static analysis, and a code replacement method to address the problem of XSS at the source code level. The objectives are to detect and remove XSS vulnerabilities from the source code before an application is deployed, thereby, preventing XSS attacks from taking place. The evaluation results are promising as the empirical validation has proven that the proposed approach has a higher precision of detecting XSS vulnerabilities than previously proposed solutions it is compared to. This approach is also able to remove the vulnerabilities detected in the tested web application source codes. Consequently, the objectives of the research were met and the expected results were achieved. This research work was limited to Java based web applications. In future research, the method can be extended to include other programming languages as well as other similar web application
security vulnerabilities.
ABSTRAK

Abstrak tesis yang dikemukakan kepada Senat Universiti Putra Malaysia sebagai memenuhi keperluan untuk ijazah Master Sains

PENDEKATAN UNTUK PENAMBAHBAIKAN PENGESANAN DAN PENGHAPUSAN KERENTANAN PENSKRIPAN SILANG-TAPAK DALAM APPLIKASI WEB

Oleh

ISATOU HYDARA

Mei 2015

Pengerusi: Abu Bakar Md Sultan, PhD
Fakulti: Sains Komputer dan Tekknologi Maklumat

Kerentanan penskripan silang-tapak (XSS) adalah antara kelemahan keselamatan yang paling biasa dan serius dalam aplikasi web. Ia berlaku disebabkan oleh verifikasi input pengguna yang tidak mencukupi, yang membolehkkan penggodam menyuntik dan melaksanakan skrip hasad dalam bahasa penanda teks hyper (HTML) di halaman aplikasi. Kewujudan kerentanan XSS dalam kod sumber aplikasi membolehkan serangan XSS mengambil tempat. Kejayaan serangan XSS ini boleh membawa kepada pencabulan keselamatan yang serius seperti rampasan akaun, penafian perkhidmatan, kecurian cookies, dan manipulasi kandungan web.


Dalam penyelidikan ini, suatu pendekatan telah diusulkan dengan meneroka kombinasi algoritma genetik dengan analisis statik, dan kaedah penggantian kod untuk menyelidika masalah XSS pada aras kod sumber. Objektif utama adalah untuk mengesan dan mengalih kerentanan XSS daripada kod sumber sebelum aplikasi dilancarkan, dengan itu, dapat mengelakkan serangan XSS daripada berlaku. Keputusan penilaian adalah agak memberangsangkan di mana, hasil daripada pengesahan empirical membuktikan kaedah yang disyorkan adalah lebih baik dalam mengesan kerentanan XSS berbanding kaedah sebelumnya. Kaedah ini juga mampu membuang kerentanan yang dikesan dalam kod sumber aplikasi web yang diuji. Oleh
ACKNOWLEDGEMENTS

Alhamdulillah, praise be to Allah in whose blessings I found the strength to accomplish this research work. I wish to thank all those who have contributed to the success of this venture.

First of all, I would like to say a big thank you to my supervisor, Assoc. Prof. Dr. Abu Bakar Md Sultan, and my co-supervisors Dr. Hazura Zulzalil and Dr. Novia Admodisastro for their invaluable support and guidance from the inception of my research work to its completion. Their constructive criticisms, comments, and suggestions have gone a long way in making this research successful.

A sincere and special thank you goes to my loving family for their endless encouragement, moral support, prayers, and understanding.

A big thank you goes to the Ministry of Education, Malaysia for providing me with a scholarship to pursue a Master of Science degree in Universiti Putra Malaysia. Their financial support has gone a long way in alleviating any financial burden I could face and enable me to focus on my research work.

I owe a big thank you to Dr. Hamza AdamuTanko for the valuable times he took to help me in conducting statistical analysis, editing and formatting for this thesis.

Last, but not the least, I would like to extend my gratitude to the Department of Software Engineering and Information System and all its staff for providing a conducive and friendly environment that enabled me work and study in peace. I am also thanking all those who, in one way or the other, have contributed to the successful completion of this thesis. May Allah reward you all, ameen.
I certify that a Thesis Examination Committee has met on 22 May 2015 to conduct the final examination of Isatou Hydara on her thesis entitled "An Approach to Improve Detecting and Removing Cross-Site Scripting Vulnerabilities in Web Applications" in accordance with the Universities and University Colleges Act 1971 and the Constitution of the Universiti Putra Malaysia [P.U.(A) 106] 15 March 1998. The Committee recommends that the student be awarded the Master of Science.

Members of the Thesis Examination Committee were as follows:

**Rusli bin Hj Abdullah, PhD**  
Professor  
Faculty of Computer Science and Information Technology  
Universiti Putra Malaysia  
(Chairman)

**Md Nasir bin Sulaiman, PhD**  
Associate Professor  
Faculty of Computer Science and Information Technology  
Universiti Putra Malaysia  
(Internal Examiner)

**Mohd Juzaiddin bin Ab Aziz, PhD**  
Associate Professor  
Universiti Kebangsaan Malaysia  
Malaysia  
(External Examiner)

---

ZULKARNAIN ZAINAL, PhD  
Professor and Deputy Dean  
School of Graduate Studies  
Universiti Putra Malaysia

Date: 17 June 2015
This thesis was submitted to the Senate of Universiti Putra Malaysia and has been accepted as fulfillment of the requirement for the degree of Master of Science. The members of the Supervisory Committee were as follows:

**Abu Bakar Md Sultan, PhD**  
Associate Professor  
Faculty of Computer Science and Information Technology  
Universiti Putra Malaysia  
(Chairman)

**Hazura Zulzalil, PhD**  
Senior Lecturer  
Faculty of Computer Science and Information Technology  
Universiti Putra Malaysia  
(Member)

**Novia Admodisastro, PhD**  
Senior Lecturer  
Faculty of Computer Science and Information Technology  
Universiti Putra Malaysia  
(Member)

________________________

**BUJANG BIN KIM HUAT, PhD**  
Professor and Dean  
School of Graduate Studies  
Universiti Putra Malaysia

Date:
Declaration by graduate student

I hereby confirm that:

- this thesis is my original work
- quotations, illustrations and citations have been duly referenced
- the thesis has not been submitted previously or concurrently for any other degree at any institutions
- intellectual property from the thesis and copyright of thesis are fully-owned by Universiti Putra Malaysia, as according to the Universiti Putra Malaysia (Research) Rules 2012;
- written permission must be obtained from supervisor and the office of Deputy Vice-Chancellor (Research and innovation) before thesis is published (in the form of written, printed or in electronic form) including books, journals, modules, proceedings, popular writings, seminar papers, manuscripts, posters, reports, lecture notes, learning modules or any other materials as stated in the Universiti Putra Malaysia (Research) Rules 2012;
- there is no plagiarism or data falsification/fabrication in the thesis, and scholarly integrity is upheld as according to the Universiti Putra Malaysia (Graduate Studies) Rules 2003 (Revision 2012-2013) and the Universiti Putra Malaysia (Research) Rules 2012. The thesis has undergone plagiarism detection software

Signature: ___________________ Date: ___________________

Name and Matric No: ___________________
Declaration by Members of Supervisory Committee

This is to confirm that:

- the research conducted and the writing of this thesis was under our supervision;
- supervision responsibilities as stated in the Universiti Putra Malaysia (Graduate Studies) Rules 2003 (Revision 2012-2013) were adhered to.

Signature: __________________________  Signature: __________________________
Name of Chairman of Name of Member of Supervisory Supervisory Committee:
Committee: __________________________  Committee: __________________________

Signature: __________________________
Name of Member of Committee:
Member of Supervisory Committee: __________________________

# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>ABSTRACT ABSTRAK</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td>v</td>
</tr>
<tr>
<td>APPROVAL</td>
<td>vi</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>xii</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>xiii</td>
</tr>
<tr>
<td>LIST OF APPENDICES</td>
<td>xiv</td>
</tr>
<tr>
<td>LIST OF ABBREVIATIONS</td>
<td>xv</td>
</tr>
</tbody>
</table>

## CHAPTER

### 1 INTRODUCTION
1.1 Background 1
1.2 Problem Statement 2
1.3 Research Objectives 3
1.4 Scope of the Thesis 3
1.5 Contribution of the Thesis 4
1.6 Organization of the Thesis 4

### 2 LITERATURE REVIEW
2.1 Introduction 6
2.2 Cross-site Scripting 6
   2.2.1 Reflected Cross-site Scripting (XSS) 7
   2.2.2 Stored Cross-site Scripting (XSS) 8
   2.2.3 Document Object Model (DOM)-Based Cross-site Scripting (XSS) 8
2.3 Systematic Literature Review Findings 9
   2.3.1 Current Research Activities on XSS 9
   2.3.2 Existing Proposed Techniques to Address XSS Problems 10
   2.3.3 Areas Where Research on XSS Mostly Focuses 14
   2.3.4 The Type of XSS Addressed the Most in Literature 15
2.4 Recent Studies on Cross-Site Scripting (XSS) 16
2.5 Program Analysis 17
2.6 OWASP's XSS Prevention Rules 18
2.7 The ESAPI API 20
2.8 Summary 23

### 3 RESEARCH METHODOLOGY
3.1 Introduction 24
3.2 Literature Review’s Method 25
   3.2.1 Protocol Development 26
   3.2.2 Research Questions and Motivation 26
   3.2.3 Search Process 27
   3.2.4 Inclusion and Exclusion Criteria 28
   3.2.5 Quality Assessment Questions of Primary Studies 29
   3.2.6 Data Extraction 29
3.2.7 Data Analysis 30
3.2.8 Limitations of The Review Study 30
3.3 Design of Proposed Approach 31
3.4 Validation of the Proposed Approach 31
3.5 Experimental Design 31
3.5.1 Definition of the Experiment 32
3.5.2 Research questions/hypothesis 32
3.5.3 Variables and measurement 33
3.5.4 Design of the experiment 33
3.5.5 Instruments 33
3.5.6 Validity threats 34
3.5.7 Analysis and Interpretation 34
3.6 Comparison of Results 34
3.7 Summary 34

4 PROPOSED APPROACH FOR XSS DETECTION AND REMOVAL 35
4.1 Introduction 35
4.2 Program Analysis 36
4.3 XSS Detection 40
4.4 XSS Removal 43
4.5 Summary 44

5 RESULTS AND DISCUSSION 46
5.1 Introduction 46
5.2 Experiment Environment 46
5.3 Test Subjects 47
5.4 Experiments and Results 48
5.4.1 Evaluation of XSS Detection 48
5.4.2 Evaluation of XSS Removal 49
5.5 Analysis and Discussion 49
5.5.1 Functional integrity 50
5.5.2 Effectiveness 51
5.6 Summary 52

6 CONCLUSION AND FUTURE WORK 53
6.1 Introduction 53
6.2 Conclusion 53
6.3 Future Work 54

REFERENCES 55
APPENDICES 71
BIODATA OF STUDENT 95
LIST OF PUBLICATIONS 96
LIST OF TABLES

Table                                                                                                                      Page

2.1. An Overview of Approaches to XSS Mitigation 11
2.2. Comparison between dates and proposed techniques/solutions 13
2.3. An Overview of Areas Focused on in XSS Research 14
2.4. Categories of XSS addressed by the studies and their percentage 15
2.5. The OWASP XSS Prevention Rules 19
2.6. Summary of XSS Prevention Rules Adapted from (OWASP, 2014c) 21
3.1. Online sources searched for relevant studies 27
3.2. Search Terms Used in Search of Relevant Studies 27
3.3. Independent and Dependent Variables 33
3.4. Design of the Study 33
5.1. Overview of Experiment Environment 46
5.2. Dataset for the Experiment 47
5.3. The GA Parameters 48
5.4. Experiment Results 48
5.5. Comparison of Results 49
5.6. Vulnerabilities Removed 49
5.7. Comparison of Precision Rates for XSS Detection 50
5.8. Comparison of the Means of Precision Rates 51
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1. A High Level View of Typical XSS Attack adapted from (Acunetix, 2014)</td>
<td>7</td>
</tr>
<tr>
<td>2.2. Simple Reflected XSS Vulnerabilities</td>
<td>8</td>
</tr>
<tr>
<td>2.3. Simple Stored XSS Vulnerability</td>
<td>8</td>
</tr>
<tr>
<td>2.4. Simple DOM-based XSS Vulnerability</td>
<td>8</td>
</tr>
<tr>
<td>2.5. The Number of Publications on XSS per Year</td>
<td>9</td>
</tr>
<tr>
<td>2.6. A potentially vulnerable code adapted from (Lwin Khin Shar &amp; Tan, 2012a)</td>
<td>22</td>
</tr>
<tr>
<td>2.7. Code secured with ESAPI security API adapted from (Lwin Khin Shar &amp; Tan, 2012a)</td>
<td>23</td>
</tr>
<tr>
<td>3.1. Research Methodology</td>
<td>25</td>
</tr>
<tr>
<td>3.2. Steps of the search strategy</td>
<td>28</td>
</tr>
<tr>
<td>4.1. Process of the Proposed Approach</td>
<td>36</td>
</tr>
<tr>
<td>4.2. A code snippet of a sample login servlet program</td>
<td>38</td>
</tr>
<tr>
<td>4.3. CFG of the Login Servlet Program</td>
<td>39</td>
</tr>
<tr>
<td>4.4. Potentially Vulnerable Paths of the Login Servlet Program</td>
<td>40</td>
</tr>
<tr>
<td>4.5. Genetic Algorithm Pseudocode adapted from (Avancini &amp; Ceccato, 2010)</td>
<td>41</td>
</tr>
<tr>
<td>4.6. Crossover</td>
<td>42</td>
</tr>
<tr>
<td>4.7. Login Sevlet Secured with ESAPI API</td>
<td>44</td>
</tr>
</tbody>
</table>
LIST OF APPENDICES

<table>
<thead>
<tr>
<th>Appendix</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A. Quality score of selected studies</td>
<td>71</td>
</tr>
<tr>
<td>B. The systematic review studies and their sources</td>
<td>74</td>
</tr>
<tr>
<td>C. Journals and conferences and the no. of studies published them</td>
<td>84</td>
</tr>
<tr>
<td>D. Summary of review findings</td>
<td>86</td>
</tr>
<tr>
<td>E. Sample XSS Attack Vectors</td>
<td>94</td>
</tr>
</tbody>
</table>
# LIST OF ABBREVIATIONS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWASP</td>
<td>Open Web Application Security Protocol</td>
</tr>
<tr>
<td>SDLC</td>
<td>System Development Life Cycle</td>
</tr>
<tr>
<td>SBSE</td>
<td>Search-Base Software Engineering</td>
</tr>
<tr>
<td>GA</td>
<td>Genetic Algorithm</td>
</tr>
<tr>
<td>EA</td>
<td>Evolutionary Algorithm</td>
</tr>
<tr>
<td>XSS</td>
<td>Cross Site Scripting</td>
</tr>
<tr>
<td>SLR</td>
<td>Systematic Literature Review</td>
</tr>
<tr>
<td>HTML</td>
<td>Hyper Text Markup Language</td>
</tr>
<tr>
<td>WWW</td>
<td>World Wide Web</td>
</tr>
<tr>
<td>DOM</td>
<td>Document Object Model</td>
</tr>
<tr>
<td>RQ</td>
<td>Research Question</td>
</tr>
<tr>
<td>QA</td>
<td>Quality Assessment</td>
</tr>
<tr>
<td>JGAP</td>
<td>Java Genetic Algorithm Package</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated Development Environment</td>
</tr>
<tr>
<td>CFG</td>
<td>Control Flow Graph</td>
</tr>
<tr>
<td>ESAPI</td>
<td>Enterprise Security API</td>
</tr>
<tr>
<td>LOC</td>
<td>Line Of Code</td>
</tr>
<tr>
<td>JDBC</td>
<td>Java Data Base Connector</td>
</tr>
</tbody>
</table>
CHAPTER 1

INTRODUCTION

1.1 Background

As technology grows, more and more people are relying on Web applications to accomplish their daily transactions (McGraw, 2006). Businesses and organizations also have been relying heavily on Web applications to provide many of their services. The explosive growth of the Internet and the attractive services it provides have made conducting these daily transactions online a too good opportunity to miss by anyone. However, as Web applications become very important to the success of businesses and organizations, their securities have become more complex (Fogie, Grossman, Hansen, Rager, & Petkov, 2007). As a result, a lot of security issues have emerged over the years due to the increasing number of threats that plague Web applications (McGraw, 2006).

Web applications are usually deployed to the public with lots of security holes. This is due to mainly the short time frame in which software are developed. Software project managers do not address security issues in their budgeting, scheduling and staffing their software development projects. Traditionally, software is developed with limited resources such as time, money and expertise. It is often delivered with lots of security vulnerabilities, thus providing attackers with a great opportunity to exploit these security holes.

Web applications security testing has, therefore, become a crucial issue to the software security industry as well as governments, businesses, and organizations. Study of the major security threats in web applications has shown that XSS vulnerabilities are among the top ten vulnerabilities, as reported by the Open Web Application Security Project (OWASP) (OWASP, 2014a).

Security is lacking in most web applications that are in used today. The software engineering community is now focusing on remedying this problem by ensuring security is integrated in all the software development lifecycle (SDLC) phases of any web application. However, security vulnerabilities can still remain in web applications despite efforts of integrating security from the beginning of development. That is where the practice of testing the security level of the product comes into play.

Cross-site scripting (XSS) vulnerabilities are among those vulnerabilities found in web applications and can be exploited through XSS attacks when such applications are deployed and running online. Injecting malicious scripts where these applications accept user inputs can result to serious security breaches such as cookie theft, account hijacking, manipulation of web content and theft of private information.

Many security solutions have been proposed over time, but they are not able to withstand all the security attacks that take place daily. Despite the fact that attention on software security is increasing, the progress on research for great solutions is
slow. Notwithstanding that research on software security is very recent real solutions are in high demand due to the importance of creating software that is secure and free of vulnerabilities. New areas of research such as Search-Based Software Engineering (SBSE) technique should be explored with the hope of finding better ways of developing secure software.

SBSE is an approach that employs search-based optimization algorithms to solve software engineering problems (Harman & Mansouri, 2010; Harman, Mansouri, & Zhang, 2009). The term SBSE was first used in 2001 but the approach has been first employed in the 1970s in software testing (Harman et al., 2009). SBSE addresses optimization problems across many areas of software engineering including requirements specifications, design, software verification and model checking, testing, maintenance, project management, and metrics (Harman et al., 2009). However, more than 50% of the literature on SBSE is focused on software testing, thus giving rise to the term Search-Based Software Testing (SBST). In addition, software testing was the first area to apply search based optimization techniques (Harman & Mansouri, 2010).

In SBSE, a search problem leads to optimal or the most favourable solutions in a search space (Harman et al., 2009) and a fitness function is required to distinguish the better and worse solutions (Harman & Mansouri, 2010). According to Harman et al. (Harman et al., 2009), the only two important aspects to consider when applying search based optimization to software engineering problems are: “(1) the choice of the representation of the problem and (2) the definition of the fitness function”. The field of SBSE is still developing and many interesting application areas are emerging (Harman & Mansouri, 2010).

Genetic algorithms have been the most commonly used of all optimization algorithms in SBSE, although there are not many studies to establish any practical performance differences among the algorithms. Genetic Algorithms (GAs) are a subset of Evolutionary Algorithms (EAs), which are metaheuristic optimization algorithms based on population and inspired by biology (Weise, 2009). They employ mechanisms of natural evolution such as mutation, crossover, natural selection, and survival of the fittest (Streichert, 2002) to find optimal solutions in a search space. GAs are different from other EAs in that they have a crossover (recombination) operation and use binary coding in bits or bit-strings to represent a population (Streichert, 2002).

GAs have proven to be good solutions to many software engineering problems, since their discovery. Their successful use in software security testing (Avancini & Ceccato, 2010) and intrusion detection systems (W. Li, 2002) enlighten the possibility of their usage in detecting and removing XSS vulnerabilities in web applications.

1.2 Problem Statement

Previous approaches to solve the problem of XSS vulnerabilities include the integration of static analysis and genetic algorithm (Avancini & Ceccato, 2010) and the combination of taint analysis and pattern matching techniques (Lwin Khin Shar
In the first solution, their method showed some improvement in capturing XSS vulnerabilities and using it as a test case in software security testing. A prototype tool has implemented this approach on PHP applications and results seemed promising. However, the fitness function of the genetic algorithm needs to be strengthened. It only counted the number of branches covered from the total branches that need to be covered in a path without considering the distance between branches. In addition, their method only tested for reflected XSS and not stored and DOM-based XSS, and does not remove XSS vulnerabilities.

In the second solution (Lwin Khin Shar & Tan, 2012a), the proposed approach is able to remove cross XSS vulnerability from web applications after discovering them. The approach works in two phases. First, it uses static analysis to identify potential XSS vulnerabilities in application source codes. Secondly, it uses pattern matching techniques to come up with good escaping mechanisms. These then help prevent input values from causing script execution. This approach, however, was only implemented on reflected and stored XSS and did not consider DOM-based XSS (Lwin Khin Shar & Tan, 2012a).

This research proposed to combine the approaches proposed in (Avancini & Ceccato, 2010; Lwin Khin Shar & Tan, 2012a) and extend them. This combination proved a much better solution in both detecting and removing cross-site scripting vulnerabilities in Java applications. The first approach (Avancini & Ceccato, 2010) uses genetic algorithms to detect the paths vulnerable to XSS but does not remove them automatically, whereas the second (Lwin Khin Shar & Tan, 2012a) is able to remove those vulnerabilities using pattern matching technique, but does not include DOM-based XSS. By combining them, this research was able to use genetic algorithms to detect and remove not only the same vulnerabilities but also DOM-based cross-site scripting vulnerabilities, which are not covered by both approaches.

1.3 Research Objectives

The main objective of this research is to improve the detection and removal of XSS vulnerabilities in web applications by combining Search-Based Software Engineering techniques, notably Genetic Algorithms, taint analysis and code replacements techniques. Hence, our specific objectives are:

1. To propose an approach to detect XSS vulnerabilities in web applications using Genetic Algorithms
2. To remove the detected vulnerabilities from the web applications source code using code replacement technique by employing the OWASP security guidelines

1.4 Scope of the Thesis

This research focused on the problem of XSS vulnerabilities detection and removal. Hence, the detection and removal of all the three types of XSS vulnerabilities, i.e. reflected, stored, and DOM-based became the main aim of this research. First, a systematic literature review was conducted to gauge the state of research on cross-site scripting. The findings helped to identify some research gap in vulnerability
detection and removal. An approach was proposed to address this gap. The approach is, currently, validated on Java-based web applications only. However, it can be extended to include other programming languages with some modifications.

1.5 Contribution of the Thesis

This research has produced an approach to detecting and removing XSS vulnerabilities in Java-based web applications. The approach is an improvement based on the combination of two previously proposed approaches. Therefore, the main contributions of this research are:

- A systematic literature review (SLR) conducted to determine the current state of research on cross-site scripting and identify research gaps
- The detection of XSS vulnerabilities in the source code of web applications using GA and their removal using source replacement technique

Furthermore, this new approach will benefit web application developers by enabling them to easily test their source codes and get rid of many XSS vulnerabilities before deployment of their applications. This in turn will benefit any user who accesses such web applications by protecting them from malicious attacks.

1.6 Organization of the Thesis

This thesis consists of six chapters that are organized as follows:

Chapter 1 gives the general overview of the thesis. It briefly describes the importance of Web applications and the security problems that affect them. It gives a background of the areas the thesis focuses on. It also identifies the problem addressed in this thesis, the objectives the thesis aims to achieve, as well as the research scope and contributions.

Chapter 2 comprises the literature review section of this thesis. It describes in detail the literature conducted on the current state of research on XSS. It describes XSS and its types in detail. The different approaches and techniques to solve XSS problems in the literature are reviewed. Some research gaps are also identified in this chapter.

Chapter 3 describes the methodology of the research. It gives an overview of the design of the proposed approach. It also describes the implementation and evaluation of the proposed approach as well as the data used in the validation of the proposed approach.

Chapter 4 proposes an approach to the detection and removal of XSS vulnerabilities. It demonstrates in detail the techniques used in the approach and how the detection and removal techniques work. It explains why the approach was proposed and how it can be an improvement to previous approaches.

Chapter 5 presents the results of the evaluation of the proposed approach. It also discusses the implications of the results and findings.
Chapter 6 concludes this thesis and discusses the new knowledge gained from this research work. It also identifies shortcomings and limitations of this research and makes suggestions that can be followed up as future work.
REFERENCES


61


<p>| S029 | An approach to demonstrate the possibility of conducting security attacks including off-path injection XSS attacks through spoofing the TCP/IP protocol. | Attack Implementation | Not specified |
| S030 | An approach to use type systems to automatically enforce programming guidelines that prevent XSS attacks in Java programs | Attack prevention | Not specified |
| S031 | Noncespaces: A technique that uses randomized XML namespaces to enable the server identify untrusted content and the client can use the information to enforce policies that will prevent XSS attacks. | Attack prevention | Not specified |
| S032 | An illustration of the possibility of carrying out XSS attacks with Scalable Vector Graphics (SVG) images through the use of tags, and an approach to limiting the risks of such attacks in web applications by removing the malicious content from a SVG file. | Attack implementation and prevention | Not specified |
| S033 | An approach to detect SQL injection and XSS attacks by implementing a security aspect through the use of Aspect-Oriented Programming (AOP) framework that validates and filters user input. | Attack detection | Not specified |
| S034 | A server-side solution to XSS and SQL injection attacks that uses MD5 algorithm and grammar expression rules to detect the attacks using a reverse proxy. | Attack detection | Not specified |
| S035 | A precise analysis of XSS sanitizers’ behaviour using BEK language that enables to write and analyse string manipulation routines and compile them to general purpose languages such as JavaScript. | Attack prevention | Not specified |
| S036 | A scheme that uses Bing-Value as HTTP response header in the browser, a binding mechanism that prevents XSS attacks. | Attack prevention | Not specified |
| S037 | A system that uses a proxy approach to detect and collect XSS vulnerabilities and uses the information to prevent XSS attacks. | Vulnerability detection and attack prevention | Not specified |
| S038 | A server side solution that uses signature based model to detect XSS attacks.* | Attack detection | Not specified |
| S039 | SessionSafe: A combination of three server side techniques that helps to prevent session hijacking attacks, which are threats resulting from XSS vulnerabilities. | Attack prevention | Not specified |
| S040 | An approach that enforces secure generation for programming languages and prevents the creation of string-based injection vulnerabilities including XSS. | Vulnerability prevention | Not specified |
| S041 | A server side detection system for XSS attacks that detects reflected XSS attacks and discovers stored XSS by monitoring the application’s HTTP traffic. | Attack detection | Reflected and stored XSS |
| S042 | Pixy: A tool that statically tests PHP web application source code to detect XSS vulnerabilities by using flow-sensitive, interprocedural, and context-sensitive data flow analysis techniques. | Vulnerability detection | Not specified |
| S043 | A novel precise alias analysis targeted at the unique reference semantics commonly found in scripting languages that is able to detect many vulnerabilities including XSS. | Vulnerability detection | Not specified |
| S044 | A library that helps to enforce secure coding of database web applications using libraries and object models. | Vulnerability Prevention | Not specified |
| S045 | Secubat: A security scanner that automatically checks | Vulnerability | Reflected |</p>
<table>
<thead>
<tr>
<th>Reference</th>
<th>Description</th>
<th>Detection</th>
<th>Prevention</th>
<th>Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>S046</td>
<td>A gateway solution that uses web page classification, referrer string, and cookies techniques and is deployed at the front end of web applications to prevent reflected XSS and cross site request forgery attacks.</td>
<td>Detection</td>
<td>Prevention</td>
<td>Implementation</td>
</tr>
<tr>
<td>S047</td>
<td>An automated technique that finds XSS and SQL injection vulnerabilities in web sites. The technique generates sample inputs, tracks taints through execution, and mutates inputs to produces exploits.</td>
<td>Detection</td>
<td>Prevention</td>
<td>Implementation</td>
</tr>
<tr>
<td>S048</td>
<td>A client-side solution to mitigate XSS attacks that acts as proxy and uses both manual and automatically generated rules.</td>
<td>Prevention</td>
<td>Prevention</td>
<td>Implementation</td>
</tr>
<tr>
<td>S049</td>
<td>Noxes: A client-side solution that acts as a proxy and uses both manual and automatically generated rules to block XSS attacks by preventing information leakage from the user environment.</td>
<td>Prevention</td>
<td>Prevention</td>
<td>Implementation</td>
</tr>
<tr>
<td>S050</td>
<td>An approach that uses Machine Learning techniques to learn the patterns of existing malicious codes and uses the information to classify new code as either malicious or not, thereby identifying XSS and SQL injection vulnerabilities.</td>
<td>Detection</td>
<td>Prevention</td>
<td>Not specified</td>
</tr>
<tr>
<td>S051</td>
<td>A solution that uses Java source code model checker, Bandera, to determine if secure programming guidelines are followed, and checks for XSS and SQL injection vulnerabilities.</td>
<td>Detection</td>
<td>Prevention</td>
<td>Not specified</td>
</tr>
<tr>
<td>S052</td>
<td>An approach called Perturbation-based Interactive UIV Testing (PIUIVT) that improves the effectiveness of vulnerability scanners for user-input-validation (UIV) testing for web applications by generating test inputs that reveal XSS and other vulnerabilities.</td>
<td>Detection</td>
<td>Prevention</td>
<td>Not specified</td>
</tr>
<tr>
<td>S053</td>
<td>FIRM: A system that embeds inline reference monitor (IRM) in web pages hosting Flash content and protects it through controlling DOM methods and randomizing variables with sensitive data in order to prevent security attack including XSS.</td>
<td>Prevention</td>
<td>Prevention</td>
<td>Not specified</td>
</tr>
<tr>
<td>S054</td>
<td>A scheme that helps to eliminate injection vulnerabilities including XSS in applications built on AJAX frameworks by refining the same-origin policy in the browser, thus preventing attacks.</td>
<td>Prevention</td>
<td>Prevention</td>
<td>Not specified</td>
</tr>
<tr>
<td>S055</td>
<td>BLUEPRINT: A defense strategy that seeks to minimize the trust put on browsers for interpreting untrusted content by eliminating any dependence on the browser’s parser.</td>
<td>Prevention</td>
<td>Prevention</td>
<td>Not specified</td>
</tr>
<tr>
<td>S056</td>
<td>A system that automatically generates attacks that exploit taint-based vulnerabilities including XSS in large Java web applications by using concrete model checking, dynamic monitoring, and program analysis techniques.</td>
<td>Detection</td>
<td>Prevention</td>
<td>Implementation</td>
</tr>
<tr>
<td>S057</td>
<td>An automated black-box vulnerability scanner that can find reflected and stored XSS in web applications by increasing testing depth and breadth, and using stateful fuzzing.</td>
<td>Detection</td>
<td>Prevention</td>
<td>Implementation</td>
</tr>
<tr>
<td>S058</td>
<td>A static string analyzer for PHP that detects XSS vulnerabilities in PHP programs using a context-free grammar to approximate web pages generated by a program.</td>
<td>Detection</td>
<td>Prevention</td>
<td>Not specified</td>
</tr>
<tr>
<td>S059</td>
<td>DESERVE: A monitor embedding framework that identifies exploitable statements in a source code using static backward slicing and embeds and helps to identify attacks including XSS.</td>
<td>Attack detection</td>
<td>Reflected and stored</td>
<td></td>
</tr>
<tr>
<td>S060</td>
<td>An approach that improves dynamic tainting technique with character coding and complement aware components to protect applications against stored XSS attacks.</td>
<td>Attack prevention</td>
<td>Stored</td>
<td></td>
</tr>
<tr>
<td>S061</td>
<td>A client-server architecture that enforces document structure integrity by combining randomization of web application code and runtime tracking of untrusted data to prevent reflected XSS attacks.</td>
<td>Attack prevention</td>
<td>Reflected</td>
<td></td>
</tr>
<tr>
<td>S062</td>
<td>A dynamic checking compiler that automatically adds checks into web applications used in three-tier Internet services to prevent attacks including XSS by using taint analysis and HTML parsers.</td>
<td>Attack prevention</td>
<td>Not specified</td>
<td></td>
</tr>
<tr>
<td>S063</td>
<td>A fully automated approach that is based on precisely tracking taintedness of data and checking specifically for dangerous content only in untrustworthy sources thereby preventing XSS attacks and others.</td>
<td>Attack prevention</td>
<td>Not specified</td>
<td></td>
</tr>
<tr>
<td>S064</td>
<td>SessionShield: A lightweight protection mechanism against a form of XSS attack called session hijacking, which detects session identifiers in incoming HTTP traffic and isolates them from the browser thereby preventing attacks.</td>
<td>Attack prevention</td>
<td>Not specified</td>
<td></td>
</tr>
<tr>
<td>S065</td>
<td>Using features from web document and ULR to classify patterns of cross site scripting attacks by employing machine learning techniques.</td>
<td>Attack detection</td>
<td>Not specified</td>
<td></td>
</tr>
<tr>
<td>S066</td>
<td>A browser-based defense mechanism against reflected XSS attacks that uses approximate string matching to detect reflected content.</td>
<td>Attack prevention</td>
<td>Reflected</td>
<td></td>
</tr>
<tr>
<td>S067</td>
<td>LAPSE: An Eclipse plugin that analyses Java EE applications for the detection of security vulnerabilities including XSS.</td>
<td>Vulnerability detection</td>
<td>Not specified</td>
<td></td>
</tr>
<tr>
<td>S068</td>
<td>A solution to identify categories of programming flaws leading software bugs and indexing existing vulnerability reports against those categories</td>
<td>Vulnerability detection</td>
<td>Not specified</td>
<td></td>
</tr>
<tr>
<td>S069</td>
<td>A method to control JavaScript execution by preventing or modifying inappropriate behaviour such as malicious injected scripts, thereby preventing XSS attacks.</td>
<td>Attack prevention</td>
<td>Not specified</td>
<td></td>
</tr>
<tr>
<td>S070</td>
<td>An intrusion detection system that identifies vulnerabilities including XSS and prevents attacks on such vulnerabilities using inter server communication techniques.</td>
<td>Vulnerability detection and attack prevention</td>
<td>Not specified</td>
<td></td>
</tr>
<tr>
<td>S071</td>
<td>A new technique called Dynamic Cookies Rewriting that renders cookies useless for cross site scripting attacks.</td>
<td>Attack prevention</td>
<td>Reflected and stored</td>
<td></td>
</tr>
<tr>
<td>S072</td>
<td>SCRIPTGUARD: An automatic context-sensitive sanitizer for ASP.NET applications that can detect and repair incorrect placement of sanitizers thus mitigating XSS and XCS attacks in legacy code.</td>
<td>Attack prevention</td>
<td>Not specified</td>
<td></td>
</tr>
<tr>
<td>S073</td>
<td>Prevention of website exploitation of cross site scripting and SQL injection vulnerabilities in PHP source code based on automated data type detection of input parameters, using a new tool: IPAAS (Input Parameter Analysis System)</td>
<td>Vulnerability prevention and Attack prevention</td>
<td>Not specified</td>
<td></td>
</tr>
<tr>
<td>S074</td>
<td>An approach that injects comments for legitimate JavaScript code that encode legitimate code features</td>
<td>Attack prevention</td>
<td>Not specified</td>
<td></td>
</tr>
</tbody>
</table>
in terms of method definition and call signatures, which makes it difficult to inject legitimate code thereby preventing injection attacks such as XSS.

<table>
<thead>
<tr>
<th>Paper</th>
<th>Description</th>
<th>Detection</th>
<th>Prevention</th>
</tr>
</thead>
<tbody>
<tr>
<td>S075</td>
<td>An approach that employs mutation-based testing technique to generate adequate test data to test for XSS vulnerabilities in PHP applications.</td>
<td>Vulnerability detection</td>
<td>All</td>
</tr>
<tr>
<td>S076</td>
<td>To propose an automated framework to detect cross site scripting attacks at the server side based on boundary injection and policy generation.</td>
<td>Attack detection</td>
<td>Not specified</td>
</tr>
<tr>
<td>S077</td>
<td>A trustworthiness testing approach of suspected phishing web sites based on behaviour model that uses Finite State Machine techniques to determine if a website can be trusted, which can detect XSS attacks</td>
<td>Attack detection</td>
<td>Not specified</td>
</tr>
<tr>
<td>S078</td>
<td>A language independent solution to block XSS attacks using the Service-Oriented Architecture approach</td>
<td>Attack prevention</td>
<td>Not specified</td>
</tr>
<tr>
<td>S079</td>
<td>A behaviour-based anomaly detection approach that puts a security layer on top of the web application to prevent XSS attacks.</td>
<td>Attack prevention</td>
<td>Not specified</td>
</tr>
<tr>
<td>S080</td>
<td>A thread-based solution for efficient process utilization of the web server and to prevent XSS attacks on AJAX applications.</td>
<td>Attack prevention</td>
<td>Not specified</td>
</tr>
<tr>
<td>S081</td>
<td>An approach for the thorough auditing of source code to defend against XSS attacks by extracting implemented defences in the code and check them for adequacy and potential risks.</td>
<td>Attack prevention</td>
<td>Not specified</td>
</tr>
<tr>
<td>S082</td>
<td>A proposed method that will recover the defence model implemented in program source code and to check the model against attacks based on given guidelines</td>
<td>Attack prevention</td>
<td>Reflected and stored</td>
</tr>
<tr>
<td>S083</td>
<td>To detect and remove the XSS vulnerabilities web applications using static analysis and pattern matching techniques</td>
<td>Vulnerability detection and removal</td>
<td>Reflected and stored</td>
</tr>
<tr>
<td>S084</td>
<td>To classify various input sanitization methods into different types and use code attributes to represent the types. Then employ data mining techniques to predict SQL injection and cross site scripting.</td>
<td>Vulnerability prediction</td>
<td>Not specified</td>
</tr>
<tr>
<td>S085</td>
<td>An approach to predicting XSS and SQL injection vulnerabilities using input validation and input sanitization patterns.</td>
<td>Vulnerability prediction</td>
<td>Not specified</td>
</tr>
<tr>
<td>S086</td>
<td>An integrated model to prevent reflected XSS and SQL injection attacks in PHP web applications.</td>
<td>Attack prevention</td>
<td>Reflected</td>
</tr>
<tr>
<td>S087</td>
<td>The construction of a common XSS vulnerability enumeration that can help security practitioners recognise common developer patterns leading to coding errors in PHP.</td>
<td>Vulnerability Prevention</td>
<td>Not specified</td>
</tr>
<tr>
<td>S088</td>
<td>A black-box analysis methodology for public Cloud interfaces that provides countermeasures for XSS attacks.</td>
<td>Attack prevention</td>
<td>Not specified</td>
</tr>
<tr>
<td>S089</td>
<td>A framework for the evaluation of web intrusion prevention systems.</td>
<td>Attack prevention</td>
<td>Not specified</td>
</tr>
<tr>
<td>S090</td>
<td>An approach to preventing the propagation of XSS worms by monitoring outgoing request that send self-replicating payloads.</td>
<td>Vulnerability detection</td>
<td>Not specified</td>
</tr>
<tr>
<td>S091</td>
<td>A model checking method that uses the automatic modelling algorithm for the HTML code to defend against XSS attacks.</td>
<td>Attack prevention</td>
<td>Not specified</td>
</tr>
<tr>
<td>S092</td>
<td>A hybrid client-server solution that combines the benefits of both server and client-side protection mechanisms to mitigate XSS attacks using anomaly detection and control flow analysis.</td>
<td>Attack prevention</td>
<td>All</td>
</tr>
<tr>
<td>S093</td>
<td>A protection scheme against attacks deployed by hiding the violation of the same origin policy including XSS that finds mismatches between the origin and target pages of HTTP request.</td>
<td>Attack prevention</td>
<td>Reflected and stored</td>
</tr>
<tr>
<td>S094</td>
<td>Alhambra: A browser-based system for testing enforcing security policies to prevent XSS attacks using taint-tracking engine and browsing history.</td>
<td>Attack prevention</td>
<td>DOM-based</td>
</tr>
<tr>
<td>S095</td>
<td>A Webmail XSS fuzzer, which works on a lexical based mutation engine and helps to discover XSS vulnerabilities in webmail applications.</td>
<td>Vulnerability detection</td>
<td>Not specified</td>
</tr>
<tr>
<td>S096</td>
<td>A client-side solution that uses step-by-step approach to protect web applications against XSS attacks.</td>
<td>Attack prevention</td>
<td>Not specified</td>
</tr>
<tr>
<td>S097</td>
<td>A client-side solution that uses a step by step approach to detect XSS attacks.</td>
<td>Attack prevention</td>
<td>Not specified</td>
</tr>
<tr>
<td>S098</td>
<td>An optimum tuning method based on the application firewall that uses keyword filtering and re-treatment to effectively block assaults including XSS attacks.</td>
<td>Attack prevention</td>
<td>Not specified</td>
</tr>
<tr>
<td>S099</td>
<td>BIXAN: A browser independent XSS sanitizer that uses a JavaScript tester, a HTML parser, and identification of static tags to prevent XSS attacks.</td>
<td>Attack prevention</td>
<td>Not specified</td>
</tr>
<tr>
<td>S100</td>
<td>Noncespaces: A technique that enables web clients to distinguish between trusted and untrusted content to prevent exploitation of XSS vulnerabilities.</td>
<td>Attack prevention</td>
<td>Reflected and stored</td>
</tr>
<tr>
<td>S101</td>
<td>FlashOver: A system that automatically scans Rich Internet Applications for XSS vulnerabilities by using a combination of static and dynamic code analysis techniques.</td>
<td>Vulnerability detection</td>
<td>Not specified</td>
</tr>
<tr>
<td>S102</td>
<td>WebAppArmor: A framework that incorporates techniques based on static and dynamic analysis, symbolic evaluation and execution monitoring to prevent XSS and other attacks on existing web applications.</td>
<td>Attack prevention</td>
<td>Not specified</td>
</tr>
<tr>
<td>S103</td>
<td>A client-side solution that uses dynamic data tainting and static analysis to prevent XSS attacks.</td>
<td>Attack prevention</td>
<td>Not specified</td>
</tr>
<tr>
<td>S104</td>
<td>A scheme on how to collect evidence after XSS attacks and strategies to prevent XSS attacks.</td>
<td>Attack prevention</td>
<td>Not specified</td>
</tr>
<tr>
<td>S105</td>
<td>A static stored XSS detection algorithm integrated with program slicing method to detect stored XSS vulnerabilities.</td>
<td>Vulnerability detection</td>
<td>Stored</td>
</tr>
<tr>
<td>S106</td>
<td>A static analysis for finding cross site scripting vulnerabilities that addresses weak or absent input validation by combining tainted information flow with string analysis.</td>
<td>Vulnerability detection</td>
<td>Reflected and stored</td>
</tr>
<tr>
<td>S107</td>
<td>A study of the security of XSS sanitization abstractions provided by frameworks that shows the gap between the abstractions and the application requirements.</td>
<td>Vulnerability detection</td>
<td>Not specified</td>
</tr>
<tr>
<td>S108</td>
<td>SWAP: A server-side solution for detecting and preventing XSS attacks using a reverse proxy that intercepts all HTML responses.</td>
<td>Attack Detection and Attack Prevention</td>
<td>Not specified</td>
</tr>
<tr>
<td>S109</td>
<td>A static analysis tool to detect XSS attacks and SQL injection vulnerabilities on ASP programs based on taint analysis.</td>
<td>Vulnerability detection</td>
<td>Not specified</td>
</tr>
<tr>
<td>S110</td>
<td>A model-based penetration testing approach for web vulnerability detection</td>
<td>Vulnerability detection</td>
<td>Not specified</td>
</tr>
<tr>
<td>S111</td>
<td>STRANGER: An automata-based string analysis tool for finding and eliminating string-related vulnerabilities including XSS in PHP applications.</td>
<td>Vulnerability detection</td>
<td>Not specified</td>
</tr>
<tr>
<td>S112</td>
<td>A set of sound abstractions for strings and string operations that allow for both efficient and precise verification of string manipulating programs to show absence of vulnerabilities.</td>
<td>Vulnerability Detection and Attack Detection</td>
<td>Not specified</td>
</tr>
<tr>
<td>S113</td>
<td>D-WAV: A web application vulnerability detection tool that uses characteristics of web forms to detect vulnerabilities including XSS.</td>
<td>Vulnerability detection</td>
<td>Not specified</td>
</tr>
<tr>
<td>S114</td>
<td>An execution-flow analysis technique is proposed that analyses the execution flow of the client-side JavaScript before the requested arrives at the browser.</td>
<td>Attack detection</td>
<td>Not specified</td>
</tr>
<tr>
<td>S115</td>
<td>MBDS: A model-based, client-side system that automatically detects XSS vulnerabilities using both primitive and advanced models.</td>
<td>Vulnerability detection</td>
<td>Not specified</td>
</tr>
</tbody>
</table>
Appendix E. Sample XSS Attack Vectors

XSS Attack Vectors

\`
';alert(String.fromCharCode(88,83,83))//
\`

\`<SCRIPT>alert(String.fromCharCode(88,83,83))</SCRIPT>\`

"';alert(String.fromCharCode(88,83,83))//

\`<IMG SRC="javascript:alert('XSS');"\`

\`<IMG SRC=javascript:alert('XSS')\`

\`<IMG SRC=""\`<SCRIPT>alert('XSS')\`</SCRIPT>">

\`<IMG SRC="javascript:alert('XSS');"\`

<iframe src=google.de></iframe>

\`<BODY onload="#%&{*}_-,;@/\"^=alert('XSS')\`

\`<SCRIPT>alert(document.cookie)</SCRIPT>\`

\`%253cscript%253ealert(document.cookie)%253c/script%253d%25d\`

\`%22%3E%3CBODY%20onload=document.write(%22%3Cs%22%2b%22b%22script%20src=http://my.box.com/xss.js%3E%3C/script%3E%22)\`

\`<img src=asdfonerror=alert(document.cookie)>\`
BIODATA OF STUDENT

Isatou Hydara was born in the Republic of The Gambia on the 15th of June, 1985. In December, 2008, she enrolled at the Faculty of Computer Science and Information Technology, Universiti Putra Malaysia for the program of Bachelor of Computer Science (Software Engineering). She graduated from the said program in October, 2012. After graduation, she continued her studies in the same university pursuing a Master of Science degree program, specializing in Software Engineering. Her research interests are focused on Software Security, particularly Web application security and Genetic Algorithms.
LIST OF PUBLICATIONS

Current state of research on cross-site scripting (XSS) – A systematic literature review, Information and Software Technology. Vol. 58 (2015), pp 170-186 (ISI Q1, IF:1.58)


Cross-Site Scripting Detection Based on an Enhanced Genetic Algorithm. In Proceedings of the 4th International Conference on Computer Science and Computational Mathematics (ICCSCM) 2015, May 7 – 8, Langkawi, Malaysia (under consideration for acceptance in ISI journal)


An Enhanced Approach for Detecting and Removing Cross-Site Scripting Vulnerabilities in Web Applications (submitted to journal)
UNIVERSITI PUTRA MALAYSIA
STATUS CONFIRMATION FOR THESIS / PROJECT REPORT AND COPYRIGHT
ACADEMIC SESSION: __________________

TITLE OF THESIS / PROJECT REPORT:
__________________________________________________________________________
__________________________________________________________________________

NAME OF STUDENT: ________________________________________________________

I acknowledge that the copyright and other intellectual property in the thesis/project report
belonged to Universiti Putra Malaysia and I agree to allow this thesis/project report to be placed at
the library under the following terms:

1. This thesis/project report is the property of Universiti Putra Malaysia.

2. The library of Universiti Putra Malaysia has the right to make copies for educational purposes
   only.

3. The library of Universiti Putra Malaysia is allowed to make copies of this thesis for academic
   exchange.

I declare that this thesis is classified as:

*Please tick (V)

☐ CONFIDENTIAL (Contain confidential information under Official Secret
  Act 1972).

☐ RESTRICTED (Contains restricted information as specified by the
  organization/institution where research was done).

☐ OPEN ACCESS I agree that my thesis/project report to be published
   as hard copy or online open access.

This thesis is submitted for:

☐ PATENT Embargo from _______ until _______  
   (date) (date)

Approved by:

(Signature of Student)  (Signature of Chairman of Supervisory Committee)
New IC No/ Passport No.:  Name:
Date:  Date:

[Note: If the thesis is CONFIDENTIAL or RESTRICTED, please attach with the letter from the
organization/institution with period and reasons for confidentially or restricted.]