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 theDegree of Master of Science

May, 2015

© COPYRIGHT UPM

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

© COPYRIGHT UPM

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.

© COPYRIGHT UPM

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

© COPYRIGHT UPM

security vulnerabilities.

© COPYRIGHT UPM

Abstrak tesis yang dikemukakan kepadaSenat 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 membolehkan 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.

Kerentanan XSS adalah mudah untuk dieksploitasi tetapi sukar untuk dihapuskan.Banyak penyelesaian telah dicadangkan untuk menyekatnya, bagaimanapun ianya masih berlaku.Banyak aplikasi web terdedah kepada XSS dan sering diserang.Kebanyakan kaedah yang dicadangkan sebelum ini hanya memfokus kepada pencegahan dan pengesanan serangan XSS semasa waktu jalan, iaitu selepas aplikasi telah dilancarkan. Beberapa pendekatan telah tertumpu kepada menghapuskan kerentanan daripada kod sumber sebelum aplikasi dilancar. Kewujudan kerentanan XSS dalam aplikasi menjadikan ianya Berjaya untuk diserang dengan mudah semasa waktu jalan. Juga hamper kesemua pendekatan ini hanya tertumpu kepada pengesanan XSS jenis I dan II tetapi tidak pada XSS jenis III, yang lebih sukar dikawal

Dalam penyelidikan ini, suatu pendekatan telah diusulkan dengan meneroka kombinasi algoritma genetic dengan analisis statik, dan kaedah penggantian kod untuk menyelesaikan 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 memberansangkan 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

© COPYRIGHT UPM

yang demikian, objektif penyelidikan dan jangkaan keputusan telah tercapai. Penyelidikan ini adalah terhad kepada aplikasi web berasaskan Java. Penyelidikan seterusnya adalah dengan meluaskan lagi kaedah ini meliputi bahasa-bahasa pengaturcaraan yang lain serta lain-lain kerentanan keselamatan aplikasi web yang serupa.

© COPYRIGHT UPM

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.

© COPYRIGHT UPM

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:

© COPYRIGHT UPM

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 comcurrently 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: _____

© COPYRIGHT UPM

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: _____
Name of
Chairman of
Supervisory
Committee: _____

Signature: _____
Name of
Member of
Supervisory
Committee: _____

Signature: _____
Name of
Member of
Supervisory
Committee: _____

© COPYRIGHT UPM

TABLE OF CONTENTS

	Page
ABSTRACT	i
ABSTRAK	iii
ACKNOWLEDGEMENTS	v
APPROVAL	vi
LIST OF TABLES	xii
LIST OF FIGURES	xiii
LIST OF APPENDICES	xiv
LIST OF ABBREVIATIONS	xv
CHAPTER	
1 INTRODUCTION	1
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	6
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 Activitieson 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	24
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

© COPYRIGHT UPM

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

© COPYRIGHT UPM

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

© COPYRIGHT UPM

LIST OF FIGURES

Figure **Page**

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

© COPYRIGHT UPM

LIST OF APPENDICES

Appendix	Page
A. Quality score of selected studies	71
B. The systematic review studies and their sources	74
C. Journals and conferences and the no. of studies published them	84
D. Summary of review findings	86
E. Sample XSS Attack Vectors	94

© COPYRIGHT UPM

LIST OF ABBREVIATIONS

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

© COPYRIGHT UPM

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).

Securityis 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

© COPYRIGHT UPM

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

© COPYRIGHT UPM

& Tan, 2012a). 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 ofXSS 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 anapproach 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 researchfocused on the problem of XSS vulnerabilities detection and removal. Hence, thedetection 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

© COPYRIGHT UPM

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 usingGA 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.

© COPYRIGHT UPM

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.

© COPYRIGHT UPM

REFERENCES

ACM. (2015). Association for Computing Machinery. Retrieved February 12, 2015, from http://www.acm.org/

Acunetix. (2014). What is Cross Site Scripting and How Can You Fix it? *Acunetix.com*. Retrieved October 11, 2014, from http://www.acunetix.com/websitesecurity/cross-site-scripting/

Adi, E. (2012). A Design of a Proxy Inspired from Human Immune System to Detect SQL Injection and Cross-Site Scripting. *Procedia Engineering*, *50*(Icasce), 19–28. doi:10.1016/j.proeng.2012.10.003

Agosta, G., Barenghi, A., Parata, A., & Pelosi, G. (2012). Automated Security Analysis of Dynamic Web Applications through Symbolic Code Execution. In *2012 Ninth International COnference on Information Technology - New Generations* (pp. 189–194). Ieee. doi:10.1109/ITNG.2012.167

Ahmed, M. a., & Hermadi, I. (2008). GA-based multiple paths test data generator. *Computers & Operations Research*, *35*(10), 3107–3124. doi:10.1016/j.cor.2007.01.012

Al-amro, H., & El-qawasmeh, E. (2012). Discovering Security Vulnerabilities And Leaks In ASP . NET Websites. In *Cyber Security, Cyber Warfare and Digital Forensic (CyberSec), 2012 International Conference on* (pp. 329–333). Kuala Lumpur: Ieee. doi:10.1109/CyberSec.2012.6246175

Allen, F. E. (1970). Control Flow Analysis. *ACM SIGPLAN Notices*, *5*(7), 1–19.

Allen, F. E., & Cocke, J. (1976). A Program Data Flow Analysis Procedure. *Communications of the ACM*, *19*(3), 137–146.

Arulsuju, D. (2011). Hunting Malicious Attacks in Social Networks. In *Advanced Computing (ICoAC), 2011 Third International Conference on* (pp. 13–17). Chennai: Ieee. doi:10.1109/ICoAC.2011.6165172

Athanasopoulos, E., Krithinakis, A., & Markatos, E. P. (2010). Hunting Cross-Site Scripting Attacks in the Network. In *W2SP 2010: Web 2.0 Security and Privacy Workshop*. Oakland, CA. Retrieved from http://w2spconf.com/2010/papers/p12.pdf

Avancini, A., & Ceccato, M. (2010). Towards Security Testing with Taint Analysis and Genetic Algorithms. In *Proceedings of the 2010 ICSE Workshop on Software Engineering for Secure Systems* (pp. 65–71). Cape Town: ACM.

Avancini, A., & Ceccato, M. (2011). Security Testing of Web Applications: A Search-Based Approach for Cross-Site Scripting Vulnerabilities. In *2011 IEEE 11th International Working Conference on Source Code Analysis and Manipulation* (pp. 85–94). Ieee. doi:10.1109/SCAM.2011.7

© COPYRIGHT UPM

Avancini, A., Ceccato, M., & Kessler, F. B. (2012). Grammar Based Oracle for Security Testing of Web Applications. In *2012 7th International Workshop on Automation of Software Test (AST)* (pp. 15–21). Zurich: Ieee. doi:10.1109/IWAST.2012.6228984

Aydin, A., Alkhalaf, M., & Bultan, T. (2014). Automated test generation from vulnerability signatures. *Proceedings - IEEE 7th International Conference on Software Testing, Verification and Validation, ICST 2014*, 193–202. doi:10.1109/ICST.2014.32

Barhoom, T. S., & Kohail, S. N. (2011). A New Server-Side Solution for Detecting Cross Site Scripting Attack. *International Journal of Computer Information Systems*, *3*(2), 19–23.

Barnett, R. (2011). XSS Street-Fight : The Only Rule Is There Are No Rules.

Barth, A., Caballero, J., & Song, D. (2009). Secure Content Sniffing for Web Browsers, or How to Stop Papers from Reviewing Themselves. In *2009 30th IEEE Symposium on Security and Privacy* (pp. 360–371). Berkeley, CA: Ieee. doi:10.1109/SP.2009.3

Bates, D., Barth, A., & Jackson, C. (2010). Regular expressions considered harmful in client-side XSS filters. *Proceedings of the 19th International Conference on World Wide Web - WWW '10*, 91. doi:10.1145/1772690.1772701

Bathia, P., Beerelli, B. R., & Laverdière, M. (2011). Assisting Programmers Resolving Vulnerabilities in Java Web Applications. In *CCIST 2011: Communications in Computer and Information Science* (Vol. 133, pp. 268–279).

Bencsath, B., Buttyan, L., & Paulik, T. (2011). XCS based hidden firmware modification on embedded devices. In *SoftCOM 2011 19th International Conference on Software Telecommunications and Computer Networks* (pp. 1–5). Ieee.

Berndt, D., Fisher, J., Johnson, L., Pinglikar, J., Watkins, A., & Management, I. P. (2002). Breeding Software Test Cases with Genetic Algorithms. In *36th Hawaii International Conference on System Sciences* (pp. 1–10). Hawaii.

Bisht, P., & Venkatakrishnan, V. N. (2008). XSS-GUARD : Precise Dynamic Prevention of Cross-Site Scripting Attacks. *DIMVA 2008, Lecture Notes in Computer Science, 5137*, 23–43. doi:10.1007/978-3-540-70542-0_2

Bojinov, H., Bursztein, E., & Boneh, D. (2009). XCS : Cross Channel Scripting and its Impact on Web Applications. In *CCS '09: Proceedings of the 16th ACM conference on Computer and communications security* (pp. 420–431). Chicago: ACM New York, NY, USA ©2009. doi:10.1145/1653662.1653713

Bozic, J., & Wotawa, F. (2014). Security testing based on attack patterns. *Proceedings - IEEE 7th International Conference on Software Testing,*

© COPYRIGHT UPM

Verification and Validation Workshops, ICSTW 2014, 4–11. doi:10.1109/ICSTW.2014.58

Brinhosa, R. B., Westphall, C. M., & Westphall, C. B. (2012). Proposal and Development of the Web Services Input Validation Model. In *2012 IEEE Network Operations and Management Symposium (NOMS)* (pp. 643–646). Ieee.

Cao, Y., Yegneswaran, V., Porras, P., & Chen, Y. (2011). POSTER : A Path-Cutting Approach to Blocking XSS Worms in Social Web Networks. In *CCS '11: Proceedings of the 18th ACM conference on Computer and communications security* (pp. 745–747). Chicago: ACM New York, NY, USA ©2011. doi:10.1145/2046707.2093483

Chandra, S. V., & Selvakumar, S. (2011). Bixsan: Browser Independent XSS Sanitizer for prevention of XSS attacks. *ACM SIGSOFT Software Engineering Notes*, *36*(5), 1–7. doi:10.1145/2020976.2020996

Chaudhuri, A., & Foster, J. S. (2010). Symbolic security analysis of ruby-on-rails web applications. *Proceedings of the 17th ACM Conference on Computer and Communications Security - CCS '10*, 585. doi:10.1145/1866307.1866373

Chen, J.-M. C. J.-M., & Wu, C.-L. W. C.-L. (2010). An automated vulnerability scanner for injection attack based on injection point. In *2010 International Computer Symposium ICS2010* (pp. 113–118). Ieee.

Choi, J., Kim, H., Choi, C., & Kim, P. (2011). Efficient Malicious Code Detection Using N-Gram Analysis and SVM. In *2011 14th International Conference on Network Based Information Systems* (pp. 618–621). Ieee. doi:10.1109/NBiS.2011.104

Coppolino, L., Antonio, S. D., Elia, I. A., & Romano, L. (2009). From Intrusion Detection to Intrusion Detection and Diagnosis : An Ontology-Based Approach. *SEUS 2009, Lecture Notes in Computer Science*, *5860*, 192–202.

CWE. (2014). CWE - CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting') (2.5). *The MITRE Corporation*. Retrieved from http://cwe.mitre.org/data/definitions/79.html

Di Lucca, G. A., Fasolino, A. R., Mastroianni, M., & Tramontana, P. (2004). Identifying Cross Site Scripting Vulnerabilities in Web Applications. In *26th Annual International Telecommunications Energy Conference* (pp. 71–80). IEEE Computer Society. doi:10.1109/WSE.2004.10013

Dong, G., Wang, X., Wang, P., & Liu, L. (2014). Detecting Cross Site Scripting Vulnerabilities Introduced by HTML5. In *11th International Joint Conference on COmputer Science and Software Engineering* (pp. 319–323).

Duchene, F., Groz, R., Rawat, S., & Richier, J.-L. (2012). XSS Vulnerability Detection Using Model Inference Assisted Evolutionary Fuzzing. In *2012 IEEE*

© COPYRIGHT UPM

Fifth International Conference on Software Testing, Verification and Validation (pp. 815–817). Ieee. doi:10.1109/ICST.2012.181

Elsevier. (n.d.). Elsevier is a world-leading provider of scientific, technical and medical information products and services. Retrieved February 12, 2015, from http://www.elsevier.com/

Faghani, M. R., & Saidi, H. (2009). Social Networks' XSS Worms. In *2009 International Conference on Computational Science and Engineering* (pp. 1137–1141). Ieee. doi:10.1109/CSE.2009.424

Ferrante, J., Ottenstein, K. J., & Warren, J. D. (1987). The Program Dependence Graph and Its Use in Optimization. *ACM Transactions on Programming Languages and Systems*, *9*(3), 319–349.

Fogie, S., Grossman, J., Hansen, R., Rager, A., & Petkov, P. D. (2007). *XSS Attacks: Cross Site Scripting Exploits and Defense*. (S. Forgie, Ed.). Burlington, MA: Elsevier, Inc/Syngress Publishing, Inc.

Fonseca, J., Vieira, M., & Madeira, H. (2007). Testing and Comparing Web Vulnerability Scanning Tools for SQL Injection and XSS Attacks. *13th Pacific Rim International Symposium on Dependable Computing (PRDC 2007)*, 365–372. doi:10.1109/PRDC.2007.55

Fortnow, L. (2009). Time for Computer Science to Grow Up. *Communications of the ACM*, *52*(8), 33–35.

Franceschet, M. (2010). The Role of Conference Publications in CS. *Communications of the ACM*, *53*(12), 129–132.

Frenz, C. M., & Yoon, J. P. (2012). XSSmon : A Perl Based IDS for the Detection of Potential XSS Attacks. In *2012 IEEE Long Island Systems, Application and Technology Conference (LISAT)* (pp. 1–4). Farmingdale, NY. doi:10.1109/LISAT.2012.6223107

Galan, E., Alcaide, A., Orfila, A., & Blasco, J. (2010). A Multi-agent Scanner to Detect Stored-XSS Vulnerabilities. In *2010 International Conference for Internet Technology and Secured Transactions (ICITST)* (pp. 1–6). London: Ieee.

Garcia-alfaro, J., & Navarro-arribas, G. (2007). Prevention of Cross-Site Scripting Attacks on Current Web Applications. *OTM 2007, Lecture Notes in Computer Science*, *4804*, 1770–1784.

Gilad, Y., & Herzberg, A. (2012). Off-Path Attacking the Web. In *Proceedings of the 6th USENIX conference on Offensive Technologies* (pp. 1–12). Berkeley: USENIX Association.

© COPYRIGHT UPM

Grabowski, R., Hofmann, M., & Li, K. (2012). Type-Based Enforcement of Secure Programming Guidelines — Code Injection Prevention at SAP. *FAST 2011, Lecture Notes in Computer Science, 7140*, 182–197.

Gundy, M. Van, & Chen, H. (2009). Noncespaces : Using Randomization to Enforce Information Flow Tracking and Thwart Cross-Site Scripting Attacks. In *16th Annual Network and Distributed System Security Symposium Proceedings, NDSS Symposium 2009*. San Diego, CA: Internet Society. Retrieved from http://www.internetsociety.org/doc/noncespaces-using-randomization-enforce-information-flow-tracking-and-thwart-cross-site

Halfond, W. G. J., & Orso, A. (2005). AMNESIA: Analysis and Monitoring for NEutralizing SQL-injection Attacks. *Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering*, 174–183. doi:10.1145/1101908.1101935

Halfond, W. G. J., Orso, A., & Manolios, P. (2008). WASP: Protecting web applications using positive tainting and syntax-aware evaluation. *IEEE Transactions on Software Engineering*, *34*(1), 65–81. doi:10.1109/TSE.2007.70748

Halpern, J. Y., & Parkes, D. C. (2011). Journals for Certification, Conferences for Rapid Dissemination. *Communications of the ACM*, *54*(8), 36–38.

Harman, M., & Mansouri, A. (2010). Search Based Software Engineering: Introduction to the Special Issue of the IEEE Transactions on Software Engineering. *IEEE Transactions on Software Engineering*, *36*(6), 737–741. doi:10.1109/TSE.2010.106

Harman, M., Mansouri, S. A., & Zhang, Y. (2009). *Search Based Software Engineering : A Comprehensive Analysis and Review of Trends Techniques and Applications*. London, UK.

Heiderich, M., & Holz, T. (2011). Crouching Tiger – Hidden Payload : Security Risks of Scalable Vectors Graphics. In *CCS '11: Proceedings of the 18th ACM conference on Computer and communications security* (pp. 239–250). ACM. doi:10.1145/2046707.2046735

Hermadi, I. (2004). *Genetic Algorithm Based Test Data Generator*. King Fahd University of Petrolieum and Minerals.

Hermosillo, G., Gomez, R., Seinturier, L., & Duchien, L. (2007). AProSec : an Aspect for Programming Secure Web Applications. In *Second International Conference on Availability, Reliability and Security (ARES'07)* (pp. 1026–1033). Vienna: Ieee. doi:10.1109/ARES.2007.43

Hidhaya, S. F., & Geetha, A. (2012). Intrusion Protection against SQL Injection and Cross Site Scripting Attacks Using a Reverse Proxy. *Communications in Computer and Iinformation Science*, *335*, 252–263. Retrieved from http://link.springer.com/chapter/10.1007%2F978-3-642-34135-9_26

© COPYRIGHT UPM

Hooimeijer, P., Livshits, B., Molnar, D., Saxena, P., & Veanes, M. (2011). Fast and Precise Sanitizer Analysis with BEK. In *SEC'11: Proceedings of the 20th USENIX conference on Security* (pp. 1–16). USENIX Association Berkeley, CA, USA ©2011. Retrieved from http://static.usenix.org/events/sec11/tech/full_papers/Hooimeijer.pdf

IEEE. (n.d.). IEEE - The world's largest professional association for the advancement of technology. Retrieved February 12, 2015, from http://www.ieee.org/index.html

Iha, G., & Doi, H. (2009). An Implementation of the Binding Mechanism in the Web Browser for Preventing XSS Attacks: Introducing the Bind-Value Headers. In *2009 International Conference on Availability Reliability and Security* (pp. 966–971). Ieee. doi:10.1109/ARES.2009.19

Ismail, O., Etoh, M., Kadobayashi, Y., & Yamaguchi, S. (2004). A Proposal and Implementation of Automatic Detection / Collection System for Cross-Site Scripting Vulnerability. In *18th International Conference on Advanced Information Networking and Applications, 2004. AINA 2004* (pp. 145–151). IEEE Computer Society. doi:10.1109/AINA.2004.1283902

Javed, A., & Schwenk, J. (2014). Towards elimination of cross-site scripting on mobile versions of web applications. *Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 8267 LNCS*, 103–123. doi:10.1007/978-3-319-05149-9-7

Jayamsakthi, S., & Ponnavaikko, M. (2007). Risk mitigation for cross site scripting attacks using signature based model on the server side. *Second International Multi-Symposiums on Computer and Computational Sciences (IMSCCS 2007)*, 398–405. doi:10.1109/IMSCCS.2007.82

JGAP. (2014). JGAP: Java Genetic Algorithms Package. Retrieved from http://jgap.sourceforge.net/

Johns, M. (2006). SessionSafe : Implementing XSS Immune Session Handling. *Lecture Notes in COmputer Science, 4189*, 444–460.

Johns, M., Beyerlein, C., Giesecke, R., & Posegga, J. (2010). Secure Code Generation for Web Applications. *Lecture Notes in Computer Science, 5965*, 96–113.

Johns, M., Engelmann, B., & Posegga, J. (2008). XSSDS : Server-side Detection of Cross-site Scripting Attacks. In *2008 Annual Computer Security Applications Conference* (pp. 335–344). Ieee. doi:10.1109/ACSAC.2008.36

Jovanovic, N., Kruegel, C., & Kirda, E. (2006a). Pixy : A Static Analysis Tool for Detecting Web Application Vulnerabilities (Short Paper). In *Proceedings of the 2006 IEEE Symposium on Security and Privacy*.

© COPYRIGHT UPM

Jovanovic, N., Kruegel, C., & Kirda, E. (2006b). Precise Alias Analysis for Static Detection of Web Application Vulnerabilities. In *PLAS '06: Proceedings of the 2006 workshop on Programming languages and analysis for security* (pp. 27–36). ACM.

Juillerat, N. (2007). Enforcing code security in database web applications using libraries and object models. *Proceedings of the 2007 Symposium on Library-Centric Software Design - LCSD '07*, 31–41. doi:10.1145/1512762.1512766

Kals, S., Kirda, E., Kruegel, C., & Jovanovic, N. (2006). SecuBat : A Web Vulnerability Scanner. In *WWW '06: Proceedings of the 15th international conference on World Wide Web* (pp. 247–256). ACM.

Kerschbaum, F. (2007). Simple cross-site attack prevention. *2007 Third International Conference on Security and Privacy in Communications Networks and the Workshops - SecureComm 2007*, 464–472. doi:10.1109/SECCOM.2007.4550368

Kieyzun, A., Guo, P. J., Jayaraman, K., & Ernst, M. D. (2009). Automatic creation of SQL Injection and cross-site scripting attacks. In *2009 IEEE 31st International Conference on Software Engineering* (pp. 199–209). Vancouver: Ieee. Retrieved from http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5070521

Kirda, E., Jovanovic, N., Kruegel, C., & Vigna, G. (2009). Client-side cross-site scripting protection. *Computers & Security*, *28*(7), 592–604. doi:10.1016/j.cose.2009.04.008

Kirda, E., Kruegel, C., Vigna, G., & Jovanovic, N. (2006). Noxes : A Client-Side Solution for Mitigating Cross-Site Scripting Attacks. In *SAC '06: Proceedings of the 2006 ACM symposium on Applied computing* (pp. 330–337). Dijon: ACM. doi:10.1145/1141277.1141357

Kitchenham, B. (2004). *Procedures for Performing Systematic Reviews*. Keele.

Kitchenham, B. A.; Charters, S. (2007). *Guidelines for performing Systematic Literature Reviews in Software Engineering*. Keele.

Komiya, R., Paik, I., & Hisada, M. (2011). Classification of malicious web code by machine learning. In *2011 3rd International Conference on Awareness Science and Technology iCAST* (pp. 406–411). Ieee.

Lengauer, T., & Tarjan, R. E. (1979). A Fast Algorithm for Finding Dominators in a Flowgraph. *ACM Transactions on Programming Languages and Systems*, *1*(1), 121–141.

Li, K. L. K. (2010). Towards Security Vulnerability Detection by Source Code Model Checking. In *Software Testing Verification and Validation Workshops ICSTW 2010 Third International Conference on* (pp. 381–387). Ieee. doi:10.1109/ICSTW.2010.23

© COPYRIGHT UPM

Li, N., Xie, T., Jin, M., & Liu, C. (2010). Perturbation-based user-input-validation testing of web applications. *Journal of Systems and Software*, *83*(11), 2263–2274. doi:10.1016/j.jss.2010.07.007

Li, W. (2002). Using Genetic Algorithm for Network Intrusion Detection, 1–9.

Li, Z., & Wang, X. (2010). FIRM : Capability-based Inline Mediation of Flash Behaviors. In *ACSAC '10: Proceedings of the 26th Annual Computer Security Applications Conference* (pp. 181–190). Austin: ACM New York, NY, USA ©2010. doi:10.1145/1920261.1920289

Liang, B., Qu, B., Jiang, S., & Ye, C. (2013). Research on Vulnerability Detection for Software Based on Taint Analysis. In *2nd International Symposium on Computer, Communication, Control and Automation* (pp. 434–437). Paris, France: Atlantis Press.

Liu, H., & Kuan Tan, H. B. (2008). Testing input validation in Web applications through automated model recovery. *Journal of Systems and Software*, *81*, 222–233. doi:10.1016/j.jss.2007.05.007

Livshits, B., & Erlingsson, Ú. (2007). Using web application construction frameworks to protect against code injection attacks. *Proceedings of the 2007 Workshop on Programming Languages and Analysis for Security - PLAS '07*, 95. doi:10.1145/1255329.1255346

Louw, M. Ter, & Venkatakrishnan, V. N. (2009). Blueprint: Robust Prevention of Cross-site Scripting Attacks for Existing Browsers. In *2009 30th IEEE Symposium on Security and Privacy* (pp. 331–346). Ieee. doi:10.1109/SP.2009.33

Martin, M., & Lam, M. S. (2008). Automatic Generation of XSS and SQL Injection Attacks with Goal-Directed Model Checking. In *17th Conference on Security Symposium* (pp. 31–43). USENIX Association Berkeley, CA, USA ©2008. Retrieved from https://www.usenix.org/legacy/event/sec08/tech/full_papers/martin/martin.pdf

Mcallister, S., Kirda, E., & Kruegel, C. (2008). Leveraging User Interactions for In-Depth Testing of Web Applications. *Lecture Notes in Computer Science*, *5230*, 191–210.

McGraw, G. (2006). *Software Security: Building Security In*. Boston, MA: Addison Wesley Profesional.

Mewara, B., Bairwa, S., & Gajrani, J. (2014). Browser's Defenses Against Reflected Cross-Site Scripting Attacks. In *International Conference on Signal Propagation and Computer Technology* (pp. 662–667).

Minamide, Y. (2005). Static Approximation of Dynamically Generated Web Pages. In *WWW '05: Proceedings of the 14th international conference on World Wide*

© COPYRIGHT UPM

Web (pp. 432–441). Chiba: ACM New York, NY, USA ©2005. doi:10.1145/1060745.1060809

Mohosina, A., & Zulkernine, M. (2012). DESERVE: A Framework for Detecting Program Security Vulnerability Exploitations. In *2012 IEEE Sixth International Conference on Software Security and Reliability* (pp. 98–107). Ieee. doi:10.1109/SERE.2012.22

Mui, R., & Frankl, P. (2011). Preventing Web Application Injections with Complementary Character Coding. *Lecture Notes in Computer Science*, *6879*, 80–99.

Nadji, Y., Saxena, P., & Song, D. (2009). Document Structure Integrity : A Robust Basis for Cross-site Scripting Defense. In *16th Annual Network and Distributed System Security Symposium Proceedings, NDSS Symposium 2009*. San Diego, CA: Internet Society. Retrieved from http://www.internetsociety.org/doc/document-structure-integrity-robust-basis-cross-site-scripting-defense

Nanda, S., Lam, L.-C., & Chiueh, T. (2007). Dynamic multi-process information flow tracking for web application security. *Proceedings of the 8th ACM/IFIP/USENIX International Conference on Middleware - Middleware '07*, 1. doi:10.1145/1377943.1377956

Nguyen-tuong, A., Guarnieri, S., Greene, D., Shirley, J., & Evans, D. (2005). AUTOMATICALLY HARDENING WEB APPLICATIONS USING PRECISE TAINTING. *IFIP Advances in Information and Communication Technology*, *181*, 295–307. doi:10.1007/0-387-25660-1_20

Nikiforakis, N., Meert, W., Younan, Y., Johns, M., & Joosen, W. (2011). SessionShield : Lightweight Protection against Session Hijacking. In *ESSoS'11: Proceedings of the Third international conference on Engineering secure software and systems* (pp. 87–100). Madrid: Springer-Verlag Berlin. doi:10.1007/978-3-642-19125-1_7

Nunan, A. E., Souto, E., Santos, E. M., & Feitosa, E. (2012). Automatic Classification of Cross-Site Scripting in Web Pages Using Document-based and URL-based Features. In *Computers and Communications (ISCC), 2012 IEEE Symposium on* (pp. 702–707). Cappadocia: Ieee. doi:10.1109/ISCC.2012.6249380

Orso, A., Sinha, S., & Harrold, M. J. (2004). Classifying Data Dependences in the Presence of Pointers for Program Comprehension , Testing , and Debugging. *ACM Transactions on Software Engineering and Methodology*, *13*(2), 199–239.

OWASP. (n.d.). OWASP Enterprise Security API - OWASP. Retrieved September 19, 2014, from https://www.owasp.org/index.php/Category:OWASP_Enterprise_Security_API

© COPYRIGHT UPM

OWASP. (2014a). Cross-site Scripting (XSS) - OWASP. *OWASP*. Retrieved from https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)

OWASP. (2014b). DOM based XSS Prevention Cheat Sheet. *OWASP*. Retrieved from https://www.owasp.org/index.php/DOM_based_XSS_Prevention_Cheat_Sheet

OWASP. (2014c). XSS (Cross Site Scripting) Prevention Cheat Sheet. *OWASP*. Retrieved from https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet

OWASP. (2015a). OWASP Java HTML Sanitizer. *OWASP*. Retrieved April 26, 2014, from https://www.owasp.org/index.php/OWASP_Java_HTML_Sanitizer_Project

OWASP. (2015b). XSS Filter Evasion Cheat Sheet - OWASP. Retrieved January 29, 2015, from https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet

Paper, W. (n.d.). Detecting Persistent Cross-site Scripting, 1–12.

Pelizzi, R., & Sekar, R. (2012). Protection, usability and improvements in reflected XSS filters. In *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security - ASIACCS '12* (p. 5). New York, New York, USA: ACM Press. doi:10.1145/2414456.2414458

Pérez, P. M., Filipiak, J., & Sierra, J. M. (2011). LAPSE + Static Analysis Security Software : Vulnerabilities Detection in Java EE Applications. *Communications in Computer and Information Science*, *184*, 148–156. doi:10.1007/978-3-642-22333-4_17

Petkov, K. (2005). Overcoming Programming Flaws : Indexing of Common Software Vulnerabilities. In *InfoSecCD '05: Proceedings of the 2nd annual conference on Information security curriculum development* (pp. 127–134). Kennesaw: ACM. doi:10.1145/1107622.1107652

Phung, P. H., Sands, D., & Chudnov, A. (2009). Lightweight Self-Protecting JavaScript. In *ASIACCS '09: Proceedings of the 4th International Symposium on Information, Computer, and Communications Security* (pp. 47–60). Sydney: ACM New York, NY, USA ©2009. doi:10.1145/1533057.1533067

PMD. (n.d.). PMD: Source Code Analyzer. Retrieved from http://pmd.sourceforge.net/

Priyadarshini, R., Jagadiswaree, D., Fareedha, A., & Janarthanan, M. (2011). A cross platform intrusion detection system using inter server communication technique. In *2011 International Conference on Recent Trends in Information Technology ICRTIT* (pp. 1259–1264). Ieee.

© COPYRIGHT UPM

Putthacharoen, R., & Bunyatnoparat, P. (2011). Protecting cookies from Cross Site Script attacks using Dynamic Cookies Rewriting technique. In *13th International Conference on Advanced Communication Technology ICACT2011* (pp. 1090–1094). Ieee.

Rafail, J. (2001). *Cross-Site Scripting Vulnerabilities*. Retrieved from http://resources.sei.cmu.edu/asset_files/WhitePaper/2001_019_001_52452.pdf

Rapid7. (n.d.). *Web Application Security - Managing Cross-Site Scripting, The Number One Item on OWASP's Top Ten List*. Retrieved from http://www.rapid7.com/docs/XSS_WP.pdf

Rathore, A. (2011). Application of Genetic Algorithm and Tabu Search in Software Testing. In *Proceedings of the Fourth Annual ACM Bangalore Conference* (pp. 1–4). Bangalore: ACM.

Rocha, T. S., & Souto, E. (2014). ETSSDetector : a tool to automatically detect Cross- Site Scripting vulnerabilities. In *13th IEEE International Symposium on Network Computing and Applications* (pp. 306–309). doi:10.1109/NCA.2014.53

Salas, M. I. P., & 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. doi:10.1016/j.entcs.2014.01.024

Saxena, P., Molnar, D., & Livshits, B. (2011). ScriptGard : Automatic Context-Sensitive Sanitization for Large-Scale Legacy Web Applications Categories and Subject Descriptors. In *CCS '11: Proceedings of the 18th ACM conference on Computer and communications security* (pp. 601–614). Chicago: ACM. doi:10.1145/2046707.2046776

Scholte, T., Robertson, W., Balzarotti, D., & Kirda, E. (2012). Preventing Input Validation Vulnerabilities in Web Applications through Automated Type Analysis. In *2012 IEEE 36th Annual Computer Software and Applications Conference* (pp. 233–243). Ieee. doi:10.1109/COMPSAC.2012.34

Shahriar, H., & Zulkernine, M. (2009). MUTEC : Mutation-based Testing of Cross Site Scripting School of Computing. In *Software Engineering for Secure Systems, 2009. SESS '09. ICSE Workshop on* (pp. 47–53). Vancouver: Ieee. doi:10.1109/IWSESS.2009.5068458

Shahriar, H., & Zulkernine, M. (2011a). Injecting Comments to Detect JavaScript Code Injection Attacks. In *2011 IEEE 35th Annual Computer Software and Applications Conference Workshops* (pp. 104–109). Ieee. doi:10.1109/COMPSACW.2011.27

Shahriar, H., & Zulkernine, M. (2011b). S2XS2: A Server Side Approach to Automatically Detect XSS Attacks. In *2011 IEEE Ninth International*

© COPYRIGHT UPM

Conference on Dependable, Autonomic and Secure Computing (pp. 7–14). Ieee. doi:10.1109/DASC.2011.26

Shahriar, H., & Zulkernine, M. (2012). Trustworthiness testing of phishing websites: A behavior model-based approach. *Future Generation Computer Systems*, *28*(8), 1258–1271. doi:10.1016/j.future.2011.02.001

Shalini, S., & Usha, S. (2011). Prevention Of Cross-Site Scripting Attacks (XSS) On Web Applications In The Client Side. *International Journal of Computer Science Issues*, *8*(4), 650–654.

Shanmugam, J., & Ponnavaikko, M. (2007a). A solution to block Cross Site Scripting Vulnerabilities based on Service Oriented Architecture. *6th IEEE/ACIS International Conference on Computer and Information Science (ICIS 2007)*, (Icis), 861–866. doi:10.1109/ICIS.2007.45

Shanmugam, J., & Ponnavaikko, M. (2007b). Behavior-Based Anomaly Detection on the Server Side to Reduce the Effectiveness of Cross Site Scripting Vulnerabilities. *Third International Conference on Semantics, Knowledge and Grid (SKG 2007)*, 350–353. doi:10.1109/SKG.2007.63

Shanmugam, J., & Ponnavaikko, M. (2007c). XSS Application Worms: New Internet Infestation and Optimized Protective Measures. *Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD 2007)*, 1164–1169. doi:10.1109/SNPD.2007.514

Shar, L. K., & Tan, H. B. K. (2010). Auditing the defense against cross site scripting in web applications. In *2010 International Conference on Security and Cryptography SECRYPT* (pp. 1–7). Ieee.

Shar, L. K., & Tan, H. B. K. (2012). Auditing the XSS defence features implemented in web application programs. *IET Software*, *6*(4), 377. doi:10.1049/iet-sen.2011.0084

Shar, L. K., & Tan, H. B. K. (2012a). Automated removal of cross site scripting vulnerabilities in web applications. *Information and Software Technology*, *54*(5), 467–478. doi:10.1016/j.infsof.2011.12.006

Shar, L. K., & Tan, H. B. K. (2012). Mining input sanitization patterns for predicting SQL injection and cross site scripting vulnerabilities. In *Proceedings - 34th International Conference on Software Engineering, ICSE 2012* (pp. 1293–1296). Zurich: Ieee. doi:10.1109/ICSE.2012.6227096

Shar, L. K., & Tan, H. B. K. (2012b). Predicting common web application vulnerabilities from input validation and sanitization code patterns. *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering - ASE 2012*, 310. doi:10.1145/2351676.2351733

© COPYRIGHT UPM

Sharma, P., Johari, R., & Sarma, S. S. (2012). Integrated approach to prevent SQL injection attack and reflected cross site scripting attack. *International Journal of System Assurance Engineering and Management*, *3*(4), 343–351. doi:10.1007/s13198-012-0125-6

Shuai, B., Li, M., Li, H., Zhang, Q., & Tang, C. (2013). Software vulnerability detection using genetic algorithm and dynamic taint analysis. *2013 3rd International Conference on Consumer Electronics, Communications and Networks*, 589–593. doi:10.1109/CECNet.2013.6703400

Sinha, S., Harrold, M. J., & Rothermel, G. (2001). Interprocedural control dependence. *ACM Transactions on Software Engineering and Methodology*, *10*(2), 209–254. doi:10.1145/367008.367022

Sivakumar, K., & Garg, K. (2007). Constructing a " Common Cross Site Scripting Vulnerabilities Enumeration (CXE)" Using CWE and CVE. *Lecture Notes in Computer Science*, *4812*, 277–291.

Somorovsky, J., Heiderich, M., Bochum, R., Gruschka, N., & Iacono, L. Lo. (2011). All Your Clouds are Belong to us – Security Analysis of Cloud Management Interfaces. In *CCSW '11: Proceedings of the 3rd ACM workshop on Cloud computing security workshop* (pp. 3–14). Chicago: ACM. doi:10.1145/2046660.2046664

Streichert, F. (2002). Introduction to Evolutionary Algorithms. In *MathFinance Workshop* (pp. 1–21). Frankfurt.

Stuckman, J., & Purtilo, J. (2011). A Testbed for the Evaluation of Web Intrusion Prevention Systems. In *2011 Third International Workshop on Security Measurements and Metrics* (pp. 66–75). Ieee. doi:10.1109/Metrisec.2011.14

Sun, F., Xu, L., & Su, Z. (2009). Client-Side Detection of XSS Worms by Monitoring Payload Propagation. *Lecture Notes in Computer Science*, *5789*, 539–554.

Sun, Y., & He, D. (2012). Model Checking for the Defense against Cross-Site Scripting Attacks. In *2012 International Conference on Computer Science and Service System* (pp. 2161–2164). Ieee. doi:10.1109/CSSS.2012.537

Sundareswaran, S., & Squicciarini, A. C. (2012). XSS-Dec : A Hybrid Solution to Mitigate Cross-Site Scripting Attacks. *Lecture Notes in Computer Science*, *7371*, 223–238.

Takesue, M. (2008). A Protection Scheme against the Attacks Deployed by Hiding the Violation of the Same Origin Policy. In *2008 Second International Conference on Emerging Security Information Systems and Technologies* (pp. 133–138). Ieee. doi:10.1109/SECURWARE.2008.24

Tang, S., Grier, C., Aciicmez, O., & King, S. T. (2010). Alhambra : A System for Creating , Enforcing , and Testing Browser Security Policies. In *WWW '10:*

© COPYRIGHT UPM

Proceedings of the 19th international conference on World wide web (pp. 941–950). ACM. doi:10.1145/1772690.1772786

Tang, Z., Zhu, H., Cao, Z., & Zhao, S. (2011). L-WMxD: Lexical based Webmail XSS Discoverer. In *2011 IEEE Conference on Computer Communications Workshops INFOCOM WKSHPS* (pp. 976–981). Ieee.

The Mendeley Support Team. (2011). Getting Started with Mendeley. *Mendeley Desktop*. London: Mendeley Ltd. Retrieved from http://www.mendeley.com

Tiwari, S., Bansal, R., & Bansal, D. (2008). Optimized client side solution for cross site scripting. In *2008 16th IEEE International Conference on Networks* (pp. 1–4). Ieee. Retrieved from http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4772647

Tsai, D.-R. T. D.-R., Chang, a Y., Liu, P. L. P., & Chen, H.-C. C. H.-C. (2009). Optimum tuning of defense settings for common attacks on the web applications. In *43rd Annual 2009 International Carnahan Conference on Security Technology* (pp. 89–94). Ieee. Retrieved from http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5335560

Uma, E., & Kannan, A. (2014). Automatic Prevention of XSS Attacks Using Intermediary Filter for Web Services. *International Journal of Soft Computing*, *9*(5), 280–284.

Uma, E., & Kannan, A. (2014). Improved cross site scripting filter for input validation against in web services. *Kuwait Journal of Science*, *41*(2), 175–203.

Van Gundy, M., & Chen, H. (2012). Noncespaces: Using randomization to defeat cross-site scripting attacks. *Computers & Security*, *31*(4), 612–628. doi:10.1016/j.cose.2011.12.004

Van-Acker, S., Nikiforakis, N., Desmet, L., Joosen, W., & Piessens, F. (2012). FlashOver : Automated Discovery of Cross-site Scripting Vulnerabilities in Rich Internet Applications. In *ASIACCS '12: Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security* (pp. 12–13). Seoul: ACM. doi:10.1145/2414456.2414462

Venkatakrishnan, V. N., Bisht, P., Louw, M. Ter, & Zhou, M. (2010). WebAppArmor : A Framework for Robust Prevention of Attacks on Web Applications (Invited Paper). *Lecture Notes in Computer Science*, *6503*, 3–26.

Veracode Inc. (2011). *How to prevent cross-site scripting attacks : Expert tactics*. Retrieved from http://docs.media.bitpipe.com/io_10x/io_101611/item_447716/Veracode_sSecurity_IO%23101611_E-Guide_112811.pdf

Vogt, P., Nentwich, F., Jovanovic, N., Kirda, E., Kruegel, C., & Vigna, G. (2007). Cross-Site Scripting Prevention with Dynamic Data Tainting and Static Analysis. In *14th Annual Network and Distributed System Security Symposium*

© COPYRIGHT UPM

Proceedings, NDSS Symposium 2007. Retrieved from http://publik.tuwien.ac.at/files/pub-inf_5310.pdf

Wang, S., Chang, Y., Chiang, W., & Juang, W. (2007). Investigations in Cross-site Script on Web-systems Gathering Digital Evidence against Cyber-Intrusions. In *Future Generation Communication and Networking (FGCN 2007) (Volume:2)* (pp. 125–129). Jeju. doi:10.1109/FGCN.2007.156

Wang, Y., Li, Z., & Guo, T. (2011). Program Slicing Stored XSS Bugs in Web Application. In *2011 Fifth International Conference on Theoretical Aspects of Software Engineering* (pp. 191–194). Ieee. doi:10.1109/TASE.2011.43

Wassermann, G., & Su, Z. (2008). Static Detection of Cross-Site Scripting Vulnerabilities. In *ICSE '08: Proceedings of the 30th international conference on Software engineering* (pp. 171–180). ACM. doi:10.1145/1368088.1368112

Weinberger, J., Saxena, P., Akhawe, D., Finifter, M., Shin, R., & Song, D. (2011). A Systematic Analysis of XSS Sanitization in Web Application Frameworks. In *ESORICS'11: Proceedings of the 16th European conference on Research in computer security* (pp. 150–171). Berlin: Springer-Verlag. Retrieved from http://link.springer.com/chapter/10.1007%2F978-3-642-23822-2_9

Weise, T. (2009). *Global Optimization Algorithms – Theory and Application –* (2nd Editio.).

Weissbacher, M., Lauinger, T., & Robertson, W. (2014). Why Is CSP Failing ? Trends and Challenges in CSP Adoption. *RAID 2014, Lecture Notes in Computer Science*, *8688*, 212–233.

Wohlin, C., Runeson, P., Bost, M., Ohlsson, M. C., Regnell, B., & Wesslen, A. (2000). *Experimentation in software engineering An Introduction*. (V. R. Basili, Ed.). Springer Science+Business Media, LLC.

Wurzinger, P., Platzer, C., Ludl, C., Kirda, E., & Kruegel, C. (2009). SWAP : Mitigating XSS Attacks using a Reverse Proxy. In *Software Engineering for Secure Systems, 2009. SESS '09. ICSE Workshop on* (pp. 33–39). Vancouver: Ieee. doi:10.1109/IWSESS.2009.5068456

Xiao, W., Sun, J., Chen, H., & Xu, X. (2014). Preventing client side XSS with Rewrite Based Dynamic Information Flow. In *Sixth International Symposium on Parallel Architectures, Algorithms and Programming* (pp. 238–243). doi:10.1109/PAAP.2014.10

Xin-hua, Z., & Zhi-jian, W. (2010). A Static Analysis Tool for Detecting Web Application Injection Vulnerabilities for ASP Program. In *2nd International Conference on e-Business and Information Security (EBISS)* (pp. 1–5). Wuhan. doi:10.1109/EBISS.2010.5473561

Xiong, P., Stepien, B., & Peyton, L. (2009). Model-Based Penetration Test Framework for Web Applications Using TTCN-3. *Lecture Notes in Business Information Processing*, *26*, 141–154.

© COPYRIGHT UPM

Yu, F., Alkhalaf, M., & Bultan, T. (2010). STRANGER : An Automata-Based String Analysis Tool for PHP. *Lecture Notes in Computer Science*, *6015*, 154–157.

Yu, F., Bultan, T., & Hardekopf, B. (2011). String Abstractions for String Verification. *Lecture Notes in Computer Science*, *6823*, 20–37.

Zhang, L. Z. L., Gu, Q. G. Q., Peng, S. P. S., Chen, X. C. X., Zhao, H. Z. H., & Chen, D. C. D. (2010). D-WAV: A Web Application Vulnerabilities Detection Tool Using Characteristics of Web Forms. In *Software Engineering Advances ICSEA 2010 Fifth International Conference on* (pp. 501–507). Ieee. doi:10.1109/ICSEA.2010.85

Zhang, Q., Chen, H., & Sun, J. (2010). An execution-flow based method for detecting Cross-site Scripting attacks. In *Software Engineering and Data Mining SEDM 2010 2nd International Conference on* (pp. 160–165).

Zhenyu, Q., Jing, X., Baoguo, L., & Fang, T. (2007). MBDS: Model-Based Detection System for Cross Site Scripting. In *IET Conference on Wireless, Mobile and Sensor Networks, 2007* (pp. 849–852). Shanghai: IET.

Zimmer, D. (2008). Real World XSS. *XSSed.com*. Retrieved from http://www.xssed.com/article/21/Paper_Real_World_XSS/

© COPYRIGHT UPM

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
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

© COPYRIGHT UPM

	web applications for XSS and SQL injection vulnerabilities by executing some form of attacks.	detection	
S046	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.	Attack prevention	Reflected
S047	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.	Attack implementation	Reflected and stored
S048	A client-side solution to mitigate XSS attacks that acts as proxy and uses both manual and automatically generated rules.	Attack prevention	Reflected and stored
S049	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.	Attack prevention	Reflected and stored
S050	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.	Vulnerability detection	Not specified
S051	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.	Vulnerability detection	Not specified
S052	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 the reveal XSS and other vulnerabilities.	Vulnerability detection	Not specified
S053	FIRM: A system that embeds inline reference monitor (IRM) in web pages hosting Flash content and protects it though controlling DOM methods and randomizing variables with sensitive data in order to prevent security attack including XSS.	Attack prevention	Not specified
S054	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. *	Attack prevention	Not specified
S055	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.	Attack prevention	Not specified
S056	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.	Vulnerability detection and Attack implementation	Not specified
S057	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.	Vulnerability detection	Reflected and stored
S058	A static string analyser for PHP that detects XSS vulnerabilities in PHP programs using a context-free grammar to approximate web pages generated by a program.	Vulnerability detection	Not specified

S059	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.	Attack detection	Reflected and stored
S060	An approach that improves dynamic tainting technique with character coding and complement aware components to protect applications against stored XSS attacks.	Attack prevention	Stored
S061	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.	Attack prevention	Reflected
S062	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.	Attack prevention	Not specified
S063	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.	Attack prevention	Not specified
S064	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.	Attack prevention	Not specified
S065	Using features from web document and ULR to classify patterns of cross site scripting attacks by employing machine learning techniques.	Attack detection	Not specified
S066	A browser-based defense mechanism against reflected XSS attacks that uses approximate string matching to detect reflected content.	Attack prevention	Reflected
S067	LAPSE: An Eclipse plugin that analyses Java EE applications for the detection of security vulnerabilities including XSS.	Vulnerability detection	Not specified
S068	A solution to identify categories of programming flaws leading software bugs and indexing existing vulnerability reports against those categories	Vulnerability detection	Not specified
S069	A method to control JavaScript execution by preventing or modifying inappropriate behaviour such as malicious injected scripts, thereby preventing XSS attacks.	Attack prevention	Not specified
S070	An intrusion detection system that identifies vulnerabilities including XSS and prevents attacks on such vulnerabilities using inter server communication techniques.	Vulnerability detection and attack prevention	Not specified
S071	A new technique called Dynamic Cookies Rewriting that renders cookies useless for cross site scripting attacks.	Attack prevention	Reflected and stored
S072	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.	Attack prevention	Not specified
S073	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)	Vulnerability prevention and Attack prevention	Not specified
S074	An approach that injects comments for legitimate JavaScript code that encode legitimate code features	Attack prevention	Not specified

© COPYRIGHT UPM

	in terms of method definition and call signatures, which makes it difficult to inject legitimate code thereby preventing injection attacks such as XSS.		
S075	An approach that employs mutation-based testing technique to generate adequate test data to test for XSS vulnerabilities in PHP applications.	Vulnerability detection	All
S076	To propose an automated framework to detect cross site scripting attacks at the server side based on boundary injection and policy generation.	Attack detection	Not specified
S077	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	Attack detection	Not specified
S078	A language independent solution to block XSS attacks using the Service-Oriented Architecture approach	Attack prevention	Not specified
S079	A behaviour-based anomaly detection approach that puts a security layer on top of the web application to prevent XSS attacks.	Attack prevention	Not specified
S080	A thread-based solution for efficient process utilization of the web server and to prevent XSS attacks on AJAX applications.	Attack prevention	Not specified
S081	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.	Attack prevention	Not specified
S082	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	Attack prevention	Reflected and stored
S083	To detect and remove the XSS vulnerabilities web applications using static analysis and pattern matching techniques	Vulnerability detection and removal	Reflected and stored
S084	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.	Vulnerability prediction	Not specified
S085	An approach to predicting XSS and SQL injection vulnerabilities using input validation and input sanitization patterns.	Vulnerability prediction	Not specified
S086	An integrated model to prevent reflected XSS and SQL injection attacks in PHP web applications.	Attack prevention	Reflected
S087	The construction of a common XSS vulnerability enumeration that can help security practitioners recognise common developer patterns leading to coding errors in PHP.	Vulnerability Prevention	Not specified
S088	A black-box analysis methodology for public Cloud interfaces that provides countermeasures for XSS attacks.	Attack prevention	Not specified
S089	A framework for the evaluation of web intrusion prevention systems.	Attack prevention	Not specified
S090	An approach to preventing the propagation of XSS worms by monitoring outgoing request that send self-replicating payloads.	Vulnerability detection	Not specified
S091	A model checking method that uses the automatic modelling algorithm for the HTML code to defend against XSS attacks.	Attack prevention	Not specified

S092	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.	Attack prevention	All
S093	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.	Attack prevention	Reflected and stored
S094	Alhambra: A browser-based system for testing enforcing security policies to prevent XSS attacks using taint-tracking engine and browsing history.	Attack prevention	DOM-based
S095	A Webmail XSS fuzzer, which works on a lexical based mutation engine and helps to discover XSS vulnerabilities in webmail applications.	Vulnerability detection	Not specified
S096	A client-side solution that uses step-by-step approach to protect web applications against XSS attacks.	Attack prevention	Not specified
S097	A client-side solution that uses a step by step approach to detect XSS attacks.	Attack prevention	Not specified
S098	An optimum tuning method based on the application firewall that uses keyword filtering and re-treatment to effectively block assaults including XSS attacks.	Attack prevention	Not specified
S099	BIXAN: A browser independent XSS sanitizer that uses a JavaScript tester, a HTML parser, and identification of static tags to prevent XSS attacks.	Attack prevention	Not specified
S100	Noncespaces: A technique that enables web clients to distinguish between trusted and untrusted content to prevent exploitation of XSS vulnerabilities.	Attack prevention	Reflected and stored
S101	FlashOver: A system that automatically scans Rich Internet Applications for XSS vulnerabilities by using a combination of static and dynamic code analysis techniques.	Vulnerability detection	Not specified
S102	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.	Attack prevention	Not specified
S103	A client-side solution that uses dynamic data tainting and static analysis to prevent XSS attacks.	Attack prevention	Not specified
S104	A scheme on how to collect evidence after XSS attacks and strategies to prevent XSS attacks	Attack prevention	Not specified
S105	A static stored XSS detection algorithm integrated with program slicing method to detect stored XSS vulnerabilities.	Vulnerability detection	Stored
S106	A static analysis for finding cross site scripting vulnerabilities that addresses weak or absent input validation by combining tainted information flow with string analysis	Vulnerability detection	Reflected and stored
S107	A study of the security of XSS sanitization abstractions provided by frameworks that shows the gap between the abstractions and the application requirements.	Vulnerability detection	Not specified
S108	SWAP: A server-side solution for detecting and preventing XSS attacks using a reverse proxy that intercepts all HTML responses.	Attack Detection and Attack Prevention	Not specified
S109	A static analysis tool to detect XSS attacks and SQL injection vulnerabilities on ASP programs based on taint analysis.	Vulnerability detection	Not specified
S110	A model-based penetration testing approach for web	Vulnerability	Not specified

	applications that uses TTCN-3 technique for test case generation related to XSS	detection	
S111	STRANGER: An automata-based string analysis tool for finding and eliminating string-related vulnerabilities including XSS in PHP applications.	Vulnerability detection	Not specified
S112	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.	Vulnerability Detection and Attack Detection	Not specified
S113	D-WAV: A web application vulnerability detection tool that uses characteristics of web forms to detect vulnerabilities including XSS.	Vulnerability detection	Not specified
S114	An execution-flow analysis technique is proposed that analyses the execution flow of the client-side JavaScript before the requested arrives at the browser.	Attack detection	Not specified
S115	MBDS: A model-based, client-side system that automatically detects XSS vulnerabilities using both primitive and advanced models.	Vulnerability detection	Not specified

© COPYRIGHT IPM

Appendix E. Sample XSS Attack Vectors

XSS Attack Vectors
';alert(String.fromCharCode(88,83,83))//\';alert(String.fromCharCode(88,83,83))//";alert(String.fromCharCode(88,83,83))//\";-></SCRIPT>">'><SCRIPT>alert(String.fromCharCode(88,83,83))</SCRIPT>
'';!--"<XSS>=&{()}
<IMG SRC="javascript:alert('XSS');"
<IMG SRC=javascript:alert('XSS')
<IMG """><SCRIPT>alert("XSS")</SCRIPT>">
<IMG SRC=javascr&#
<IMG SRC=&#0000106&#0000097&#0000118&#0000097&#0000115&#0000099&amp
<IMG SRC="javascript:alert('XSS');"
perl -e 'print "<IMG SRC=java\0script:alert(\"XSS\")>";' >ou
<BODY onload!#$%&()*~+-_.,:;?@[/|\]^`=alert("XSS")
"><iframe src=google.de></iframe>
<BODY BACKGROUND="javascript:alert('XSS')">
<FRAMESET><FRAME SRC="javascript:alert('XSS');"></FRAMESET
<IFRAME SRC="javascript:alert('XSS');"></IFRAME>
“><script >alert(document.cookie)</script
%253cscript%253ealert(document.cookie)%253c/script%253
“><s”%2b”cript>alert(document.cookie)</script
%22/%3E%3CBODY%20onload=’document.write(%22%3Cs%22%2b%22cript%20src=http://my.box.com/xss.js%3E%3C/script%3E%22)’
<imgsrc=asdfonerror=alert(document.cookie)>

© COPYRIGHT IBM

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.

© COPYRIGHT UPM

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**)

An Approach for Cross-Site Scripting Detection and Removal Based on Genetic Algorithms. In Proceedings of the International Conference on Software Engineering Advances (ICSEA) 2014, October 12 – 16, Nice, France

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)

Removing Cross-Site Scripting Vulnerabilities Using the OWASP Security Guidelines. In Proceedings of the 4th International Conference on Computer Science and Computational Mathematics (ICCSCM) 2015, May 7 – 8, Langkawi, Malaysia

An Enhanced Approach for Detecting and Removing Cross-Site Scripting Vulnerabilities in Web Applications (submitted to journal)

© COPYRIGHT UPM

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 (√)

☐	**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)
New IC No/ Passport No.:

Date :

_____
(Signature of Chairman of Supervisory Committee)
Name:

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.]