Journal of Computer and Communications, 2015, 3, 28-40
Published Online September 2015 in SciRes. http://www.scirp.org/journal/jcc
http://dx.doi.org/10.4236/jcc.2015.39004

Scientific Research Publishing

Systematic Review of Web Application Security Vulnerabilities Detection Methods

Sajjad Rafique, Mamoona Humayun, Zartasha Gul, Ansar Abbas, Hasan Javed

Department of Computer Science, University Institute of Information Technology, PMAS-Arid Agriculture University, Rawalpindi, Pakistan
Email: sajjad394@gmail.com, mamoona@uaar.edu.pk, zartashagul88@gmail.com, abbas.ansar514@gmail.com, hasan647@gmail.com

Received 28 July 2015; accepted 12 September 2015; published 15 September 2015

Abstract

In recent years, web security has been viewed in the context of securing the web application layer from attacks by unauthorized users. The vulnerabilities existing in the web application layer have been attributed either to using an inappropriate software development model to guide the development process, or the use of a software development model that does not consider security as a key factor. Therefore, this systematic literature review is conducted to investigate the various security vulnerabilities used to secure the web application layer, the security approaches or techniques used in the process, the stages in the software development in which the approaches or techniques are emphasized, and the tools and mechanisms used to detect vulnerabilities. The study extracted 519 publications from respectable scientific sources, *i.e.* the IEEE Computer Society, ACM Digital Library, Science Direct, Springer Link. After detailed review process, only 56 key primary studies were considered for this review based on defined inclusion and exclusion criteria. From the review, it appears that no one software is referred to as a standard or preferred software product for web application development. In our SLR, we have performed a deep analysis on web application security vulnerabilities detection methods which help us to identify the scope of SLR for comprehensively investigation in the future research. Further in this SLR considering OWASP Top 10 web application vulnerabilities discovered in 2012, we will attempt to categories the accessible vulnerabilities. OWASP is major source to construct and validate web security processes and standards.

Keywords

Software Development Lifecycle, Web Applications, Security Vulnerabilities, Systematic Literature Review

How to cite this paper: Rafique, S., Humayun, M., Gul, Z., Abbas, A. and Javed, H. (2015) Systematic Review of Web Application Security Vulnerabilities Detection Methods. *Journal of Computer and Communications*, **3**, 28-40.
http://dx.doi.org/10.4236/jcc.2015.39004

1. Introduction

Due to more customer data going online by adapting to online banking or fund transfer practices, users' accounts and other information have become vulnerable to fraud and other attacks. Also, hackers in recent years are increasingly targeting web applications, since most networks are closely monitored through Intrusion Detection Systems (IDS) and firewalls. Therefore, the web application layer needs to be secured from unauthorized users by building across the software development lifecycle security mechanism [1]. This ensures that it is not an afterthought issue, only considered in the end [2] as in many software development processes, where as a result, attackers continue to explore areas of vulnerability to undermine the integrity of applications. In recognition of this problem, developers have to incorporate security during the development in order to produce vulnerability free software systems, since the existence of flaws at the design or coding phase of the development lifecycle can open web applications to a wide range of attacks [3].

However, many studies have been conducted by both practitioners and researchers on the mechanism of building security in web applications, but few studies have been carried out on security development models that guide the development process [4].

Therefore, there is a need to investigate the available development phases of software lifecycle, as well as the security mechanism, to determine what the most appropriate approach or technique is for securing the web applications layer, and for what vulnerability. This SLR uses [5] guidelines to investigate the different studies available on security development models within the web application layer. Details addressed include the approach or technique used, the vulnerability addressed and the stage in the development life cycle where the approach or technique is emphasized.

The SLR attempts to answer the question of what the most security vulnerability harm the web application. It also addresses the question of under what circumstance the security approach or technique is most efficient in addressing the inherent vulnerabilities. Section 2 describes in detail what a systematic literature review is all about and the procedure followed by this review. Section 3 describes the result obtained from the review and Section 4 discusses the results of Section 3 and makes recommendations.

2. Related Work

Some of the previous systematic literature review such as "Empirical Analysis of System-Level Vulnerability Metrics through Actual Attacks" [6] has highlighted the benefits, shortcomings and strengths of agile methods. It has further explored the implication of the review as imitating a need for better empirical study of agile software development. The review has neither strictly covered the aspect of security surrounding the vulnerability detection method, nor has it addressed the circumstances where the development tool is efficient with regard to security mitigation.

Also, another literature review conducted by [7], they have stressed on security requirement engineering. The review chooses studies that have assimilated security only at the requirement stage of information system construction without paying much attention to security across the development lifecycle.

All the reviews mentioned has not sufficiently address the issue of building security across the entire lifecycle of web application security development.

Similarly, there other systematic reviews conducted on other aspects of web applications area which has no association with the current systematic review on building security in development lifecycle. For instance, [8] conducted systematic review to identify and discuss the existing vulnerabilities and tools used by small and medium web companies and to propose new software vulnerability detection methods in order to measure the company's success. Also Blanco *et al*. used systematic view to identify, extract and analyze the main plans for security ontologies. None of these reviews have addressed the question of the most appropriate security tools and techniques for mitigating vulnerability in the web applications.

3. Systematic Review Process

The construction of this review is based on kitchenham guiding principle [9], while also being channeled by other systematic literature review articles in the area of software development, such as [10] and [11]. A wide-ranging search is carried out into a number of respected science sources, *i.e.* the IEEE Computer Society, ACM Digital Library, Google Scholar, Science Direct, and Springer Link. A total of 623 published papers in the do-

main were extracted. After a careful synthesis, and based on inclusion or exclusion criteria, 56 papers that are primarily centered on security in the web application vulnerabilities were considered. Hence a thorough picture of the state of the art in different web application vulnerabilities security approaches is investigated, and other issues surrounding the research question are also brought to light with worthwhile directions for future investigations.

This section shows how the review process was conducted including activities such as question formularization, source selection, study selection process, information extraction and extraction execution.

4. Question Formulization

Based on the (Kitchenham 2007) guidelines for systematic literature reviews, research questions are the most important aspect of review since they guide the entire process by ensuring that primary study selection and aggregation relate directly to the question. Therefore, asking the appropriate questions is critical to finding an appropriate research dimension in SLR. Hence this review has drawn up the research question thus: "Researchers have employed which methods to detect security vulnerabilities of web application".

The research question follows the PICO paradigm mentioned in the (Kitchenham 2007) guideline for SLR.

Population: It is consist on set of studies and articles indicating the empirical studies for detection of the security vulnerabilities of web applications.

Intervention: Solutions mentioned in the studies for detecting security vulnerabilities of web applications.

Outcome: Measure and form of evidence linked to the detection of security vulnerabilities in web applications.

Context: Covering the domain of web applications focusing on empirical studies.

The above question is further refined to the following important questions:

RQ1. *What are the methods adopted by researchers and practitioners in order to detect web application security vulnerabilities*.

The research question RQ1. Is enhanced into following sub questions?

1) *Enlist and elaborate the stage*(*s*) *of web application in which the vulnerability detection methods* (*VDMs*) *are functional.*

The aim of this question is to highlight the phases of web development process in which VDMs are applied in number of times, what kind of web security vulnerabilities that are extracted during the web development stages are analyzed, and in what way these methods are combined during the process of web development.

2) *Which web application vulnerability*(*es*) *have been frequently detected during empirical analysis*?

During the analysis different security vulnerabilities are described which affect the web application severely and occur frequently while some are not very common and not occur so many times. The aim of this research is to detect most severe and dangerous vulnerabilities in web applications.

3) *Enlist data characteristics used for web application security vulnerabilities detection*?

This question will answer about single, cross-company and student/commercial projects data sets used for evaluation.

RQ2. *Can OWASP Top* 10 *help to categories these vulnerabilities*?

The OWASP Top 10 is widely used by the researchers in case of web application vulnerabilities detection. There is broad consensus regarding OWASP Top 10 for detection of most critical web application vulnerabilities. It is consulted in all over the world by the security experts to contribute in OWASP Top 10 [5]-[7] [9], and [12].

The research question RQ1 is expected to identify the different techniques used by researchers or practitioners when developing secure web applications. These tools and techniques could be well-known software engineering tools with built-in security, or a new security paradigm purposely designed for web application security development.

Similarly, research question RQ2 is expected to identify the most of the vulnerabilities that are detected within the security development lifecycle, such as Cross site scripting, SQL Injection and Insecure direct object.

5. Source Selection

In the way to identify as many primary studies as possible that are related to these research questions, the study carried a pilot search on some trustworthy database sources. A pilot search on these sources has discovered that

some similar publications are indexed in two different sources, and therefore, the search selection is limited to the following database sources:

- IEEE Explorer;
- ACM Digital Library;
- Springer Link;
- Science Direct.

At first the search keywords were extracted from **Population**, **Intervention**, **Context** and **Outcome** paradigm (PICO), as claimed in the earlier section. After a steer assessment, the recommended keywords from the sources were combined to the list of keywords during live-run searches.

Table 1 shows the primary strongly matched search keywords based on the following derivation:

- Population: web application;
- Intervention: security vulnerability;
- Context: domain of web applications;
- Outcome: quantity and type appropriate vulnerability.

Therefore, keywords are concatenated using Boolean "AND" and "OR" to come up with various query strings such as those shown in **Table 2**.

In the search keyword combination using "AND" and "OR" even though a sole query string might construct higher results in a particular data source than other sources, the query string that created the on the whole best result was:

"Web AND security AND web development OR web development vulnerabilities OR development procedure OR software lifecycle".

The search results were purified based on applicability to the search keyword since the date of publication is irrelevant to the research questions.

6. Study Selection

Before the extraction of articles from the identified sources was done, the method of inclusion and exclusion of articles based on the research questions was explained. The following are the criteria used in excluding or including a publication:

Initial selection for inclusion was based on whether the article title, abstract or introduction has a clear connection with the study.

The second stage excluded all articles that do not consider security issues in the software development, since the major concern of this study is security vulnerabilities.

The third stage excluded those articles that, although security related, do not directly involve the web application layer. The articles in this case might be related to security on web services such as browsers, mash ups, and other services.

The fourth and final stage included all primary studies related to the two research questions.

The first author of this study applied the inclusion/exclusion criteria to the publications regained using the search keywords. The second author used a quantitative valuation.

Checklist to assess the quality of the individual publication.

6.1. Study Assessment Checklist

The study assessment checklist is designed mostly to avoid publication bias that can lead to bias in systematic reviews. The checklist provide for the selected publications to be assessed using an unbiased strategy based on whether they actually supposed to be included. It minimizes bias and maximizes validity by including questions aimed at assessing the degree to which articles have addressed bias and validity.

The scoring modalities for the qualitative assessment checklist were done as follows: The possible answer to quality assessment questions are: "Very related (+1)", "Related (0)", and "Not so related (−1)".

The checklist is shown in **Table 3**.

6.2. Quality Evaluation Results

In **Table 4** shows the number of studies selected in each year.

6.3. Selection Execution

A total of 579 papers were extracted based on the first step inclusion and exclusion criteria.

The second step of the process excluded 71 papers from the total number, while the third step excluded 309 papers. Therefore, from the 415 extracted studies, only a total of 56 publications were taken in this systematic review after removing the duplicates.

6.4. Information Extraction

The data extraction form as shown in **Table 5** included information about the primary study itself as well as the information required to address the research questions. In order to gather the required information to address the objective of the study a data extraction form is designed. The full paper was read to collect the required data, and

Table 1. Closely matched keywords.

Keywords	Initially Matched Keywords
Web	web, internet, world wide web, net-centric, web hypermedia, web-enabled application, e-commerce, e-banking, e-business, e-transaction-trade
Application	web application, web service, internet application, web-based application, software, system
Security	security, secure, insecurity, vulnerabilities, robust, safety, secure
Vulnerability	vulnerability, threat, attack, risk
Method	methods, processes, techniques, system, practice, procedures, models

Table 2. The search keyword combination using "AND" and "OR".

S. No	Combination using AND/OR
1	"web AND security AND development vulnerability OR software lifecycle"
2	"Internet AND security AND development vulnerability OR development lifecycle"
3	"www AND security AND development vulnerability OR development lifecycle"
4	"online AND security AND development vulnerability OR development lifecycle"
5	"world wide web AND security AND development vulnerability OR development lifecycle"
6	"e-commerce AND security AND development vulnerability OR development lifecycle"
7	"e-commerce AND security AND development vulnerability OR development lifecycle"
8	"e-banking AND security AND development vulnerability OR development lifecycle"
9	"e-business AND security AND development vulnerability OR development lifecycle"
10	"e-transaction AND security AND development vulnerability OR development lifecycle"
11	"e-trade AND security AND development vulnerability OR development lifecycle"
12	"electronic banking AND security AND development vulnerability OR development lifecycle"
13	"electronic web AND security AND development vulnerability OR development lifecycle"
14	"electronic business AND security AND development vulnerability OR development lifecycle"
15	"electronic transaction AND security AND development vulnerability OR development lifecycle"
16	"electronic trade AND security AND development vulnerability OR development lifecycle"

Table 3. Quality evaluation checklist.

No.	Question	Possible Answers		
Q.1	VDMs are described in detail in the study?	+1	0	−1
Q.2	The guidelines are provided in the given study for the application of VDMs?	+1	0	−1
Q.3	The clear results are obtained after the application of VDM?	+1	0	−1
Q.4	The study under observation has been published in a relevant journal/conference?	+1	0	−1
Q.5	The other authors also cited the given study?	+1	0	−1

Table 4. Studies with percentage according to year.

Years	Relevant Studies	Selected Studies	Percentage
2002	4	1	1.8
2003	5	2	3.5
2004	3	2	3.5
2005	11	3	8.9
2006	3	1	1.8
2007	7	4	7.1
2008	6	4	7.1
2009	22	4	7.1
2010	32	7	17.8
2011	14	4	7.1
2012	23	5	8.9
2013	18	6	10.7
2014	14	6	10.7
2015	12	7	17.8
Total	162	56	

Table 5. Form for data extraction.

Paper Title			
Authors		**Publication Year**	
Source		**Evaluator**	
Assessment of Quality	**1**	**0**	**−1**
WAS risk evaluation is provided in detail in the study?			
The clear guidelines are provided in the study to apply the WAS risk evaluation risk method?			
The clear results are provided after application of the WAS risk evaluation methods?			
The study under observation has been published in a relevant journal/conference?			
The other authors also cited the study?			

Extraction of Data for Questions	Answers
Which journals/conferences include papers on web application security?	Journal/Conf Name
What risks in web application security are addressed?	Risk Name
Which risk from OWASP Top 10 web application security risks is addressed?	From OWASP Top 10
Which solutions of web application security have been proposed for web application development?	
Which type WAS method employed?	1. Testing 2. Inspection 3.Inquiry Analytical 4. Modeling Simulation
Which type of evaluation is performed by the WAS risk evaluation methods employed?	1.Automated 2. Manual
WAS risk evaluation methods is applied in which phase(s) and web artifacts?	1. Requirement 2. Design 3. implementation
Any feedback provided by the WAS evaluation methods?	1. Yes 2. No
There is empirical validation of web application security issue(s)?	1. Case study 2. Experiment 3. Survey 4. No

the following information have been extracted from each paper: source, authors, title, publication year and research question answers; and information required to classify the study using used facets.

Also important in the data extraction form are:

- Application used to build security;
- Security approach or technique;
- Stage in the lifecycle where security is incorporated;
- Vulnerability it addresses;
- Tool used to identify vulnerabilities;
- Mechanism being adopted.

To answer research questions the data characteristics are also described, so that the validity of data can be ensured and detailed information can be provided. These data characteristics are highlighted in **Table 6**. The two main sources of data includes from academia and industry which indicate high percentage as compared to other sources.

7. Results

There are three major categories of primary studies identified from the extracted publications:

- Studies involving validation methods;
- Studies involving development stages or a lifecycle where security is emphasized;
- Studies involving security tools and mechanisms for detecting vulnerabilities.

Table 7 shows the various methods and strategies used to validate the given studies. The graphical representation of **Table 7** is also shown in **Figure 1**. Similarly, **Table 8** shows the category of studies that consider security at various development phases [13].

Some of the studies consider placing security checks during requirements and design, while others consider the security checks through implementation and testing phase.

Table 9 depicts the security vulnerabilities identified by the OWASP Top 10 during software development. These Top 10 vulnerabilities are documented in the list of OWASP Top 10. **Figure 2** is also representing the percentage of listed vulnerabilities. **Table 10** indicates the data characteristics from different fields, the tools used to assess vulnerabilities and the mechanism used such as code vulnerability analysis, run-time check and others

Table 6. Data characteristics.

Data Field	Responses	Resp. %age
Academia	21	52
Mixed	5	12
Industrial	15	36
Government	2	5
Others	10	23
Mean: 3.268	Std. Deviation: 1.672	Satisfaction Rate: 38.208

Table 7. Methods used for validation.

Validation Method	Responses	Response %age
Case study	10	24
Experiment	25	62
Survey	2	5
Experience Report	4	12
Observational Study	3	7
Action Research	1	2
Others	3	8
Mean: 3.122	Satisfaction Rate: 26.871	Std. Deviation: 1.716
Variance: 2.946	Std. Error: 0.245	

Table 8. Research type and software development phases.

Phases	Responses	Response %age
Requirement	4	7
Design	6	17
Implementation	26	66
Testing	20	51
Maintenance	0	0
Mean: 4.439		St. Deviation: 1.537
Satisfaction Rate: 53.448	Variance: 2.363	Std. Error: 0.202

Table 9. Detection of security vulnerabilities from OWASP Top 10.

Vulnerability Name	Responses	Resp. %age
Injection vulnerability	27	68
Cross site scripting (xss)	21	49
Broken authentication and session management	4	10
Insecure direct object references	1	2
Cross site request forgery (csrf)	5	12
Security misconfiguration	1	2
Failure to restrict url access	2	4
Invalidated redirects and forwards	1	2
Insecure cryptographic storage	2	5
Insufficient transport layer protection	0	0
Others	22	54
Mean: 9.31	Std. Deviation: 6.33	Satisfaction Rate: 35.47

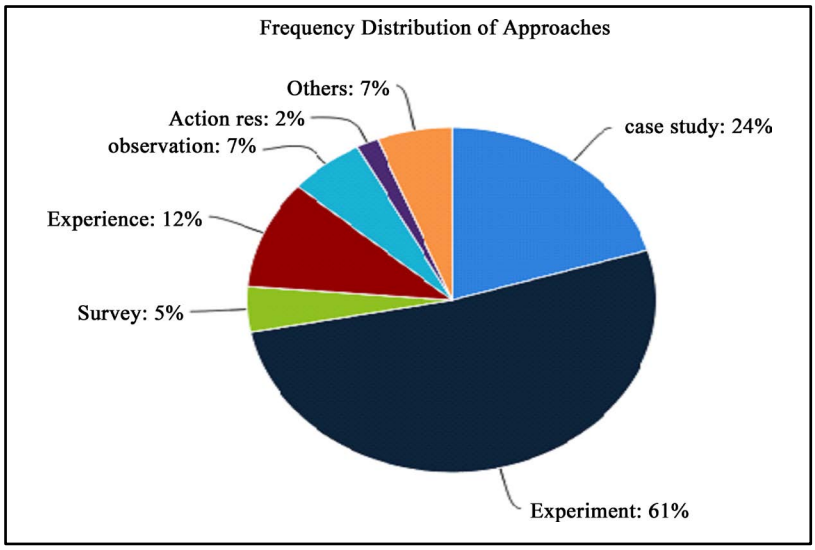

Figure 1. Frequency distribution of approaches.

are depicted in **Table 10**.

8. Discussion

This section is intended to discuss and analyze the result presented in the previous section and provide significant suggestions that may lead to an in-depth understanding of the domain.

Therefore, our discussion is based on the following result.

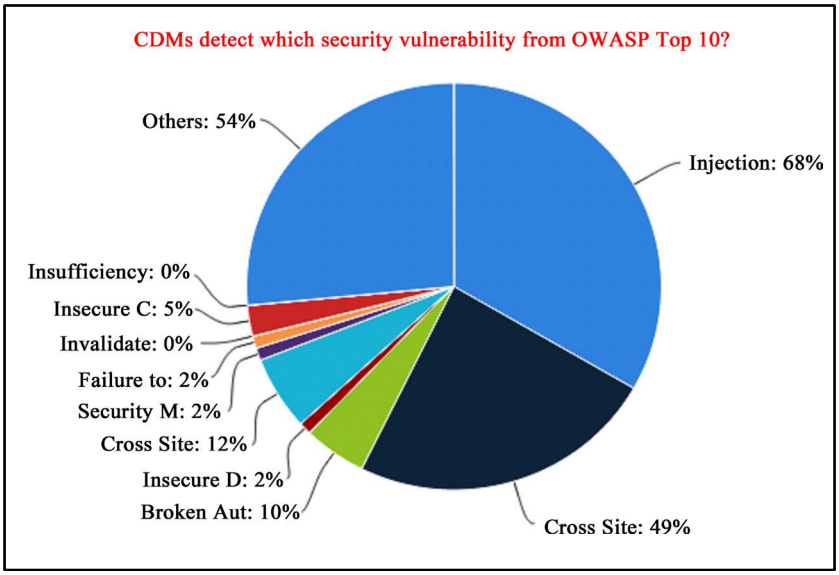

Figure 2. Vulnerability detected by VDM.

Table 10. Studies with tools and security mechanism.

Author	Year of publications	Methodology for evaluating security	Mechanism for assessing security vulnerability	Phases
P. Salini1 *et al*.	2012	Model-Oriented Security Requirement Engineering (MOSRE)	E-voting(Run time)	Requirement
Anshika Pandey *et al*.	2014	H2S Application framework	Run time	Implementation
Mattia Monga *et al*.	2008	Phan (PHP Hybrid Analyzer)	Run time	Testing
Romaric Ludinard *et al*.	2012	RRABIDS (Ruby on Rails Anomaly Based Intrusion Detection System)	Code analysis	Requirement
Theodoor Scholte *et al*.	2013	IPAAS (Input Parameter Analysis System)	Code analysis	Testing
Lwin Khin Shar *et al*.	2012	Safer XSS	Run time	Testing
Li *et al*.	2010	Perturbation based Interactive UIV Testing (PIUIVT)	Code analysis	Implementation
Mei Junjin	2009	SQL Injection Gen	Run time checks	Testing
Davide Balzarotti *et al*.	2008	Saner	Run time	Implementation
Abdelkader Lahmadi*et al*.	2012	Sec SIP framework	Code analysis	Testing
Michael D. Bond *et al*.	2010	PECAN (Precise, Efficient, Context-sensitive Anomaly detection)	Code analysis	Implementation
Viktoria Felmetsger *et al*.	2010	Waler's architecture.	Run time	Testing
Santa Barbara	2007	MiMoSA	Code analysis	Implementation
Yao-Wen Huang *et al*.	2004	Web SSARI (Web Application Security by Static Analysis and Runtime Inspection)	Code analysis	Testing
Ibéria Medeiros *et al*.	2013	Web Application Protection (WAP) tool	Taint analysis	Testing
Yao-Wen Huang *et al*.	2003	Web Application Vulnerability and Error Scanner (WAVES)	Code analysis	Testing
Trevor Jim *et al*.	2007	Browser-Enforced Embedded Policies (BEEP)	Run time	Implementation
Prithvi Bisht	2010	NOTAMPER tool	Code analysis	Testing

8.1. Security Approach Used

Security approaches are mechanism or procedures that are integrated during the development of a secure web application using some systematic and well defined methods such as OWASP Top 10 that is fused in web application development [14]. There are some degrees of similarity among the different studies in terms of detection of vulnerabilities.

OWASP Top 10 is aimed at analyzing the security of a system by identifying the vulnerabilities of web applications. It complements security review process by building security at the start of software development life cycle. **Table 8** shows the security vulnerabilities from OWASP Top 10. According to OWASP Top 10 Injection vulnerability and Cross site scripting are most common with high frequency of 27 and 21 respectively. Beside this, the vulnerabilities falling in other category are also yielding frequency of 22. The one ought to be conscious about the nature stored data or information, its position and its entrance control strategy. Similarly Cross site request forgery (CSRF) (frequency = 5) is damaging vulnerability in web applications. Broken Authentication and Session management (frequency = 4) is security flaw. Insecure Cryptographic Storage is detected with (frequency = 2) security gap. While some of the vulnerabilities like Insecure Direct Object References, Security Misconfiguration, and Failure to Restrict URL Access are shown with (frequency = 1) security threats. From the list of OWASP Top 10 the vulnerabilities such as Invalidated Redirects and Forwards and Insufficient Transport Layer Protection are not addressed in any study under our observations.

As reflected in the data shown in **Table 8**, we found many single vulnerability that effects the web application in more times.

8.2. Lifecycle Stage

The stage in the development lifecycle where a security approach or technique is emphasized varies with different studies. **Table 8** shows the distribution of research type of the selected studies. The results of this classification are also represented by **Figure 3**. We also classified the studies on the basis of different stages of software development. Specifically, we assembled the software development stages into: requirements, design, implementation, testing, and maintenance. The breakdown of the classification of the selected studies is given in **Table 8**. The majority of selected primary studies addressed the (51%), while some of the studies were classified under the Requirement and Design phase respectively (7% & 17%). We did not find any study related to software maintenance stage.

It may be noted that vulnerabilities detected in requirement phase are very less in percentage. Which may create more problems in next stages of development? Similarly the design phase However, applying security checks across the entire lifecycle has received less attention. Similarly, there has not been an empirical study to the best of our knowledge that assesses whether concentrating security around maintenance is sufficient or not.

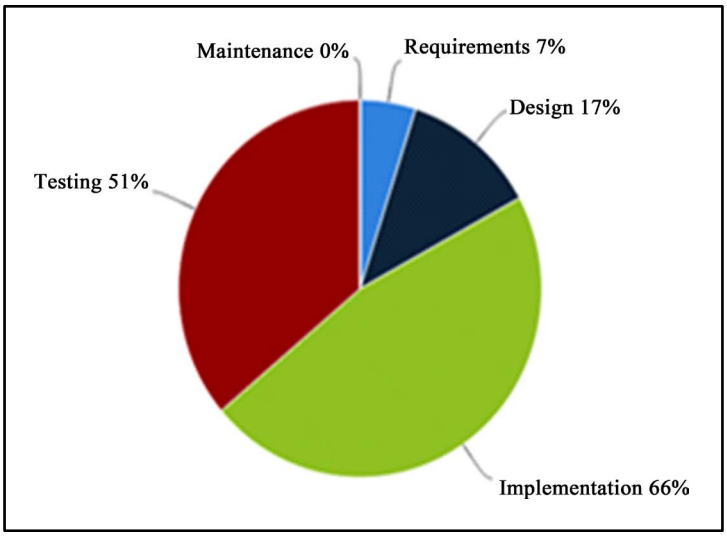

Figure 3. Phases of web applications in which VDMs are applied.

However, putting security checks across the entire lifecycle, which also includes all phases, can guarantee more assurances than if they are only introduced during the testing phase.

8.3. Assessment Mechanism

It is clearly represented in **Table 10** that most approaches focus on code vulnerabilities using code vulnerability assessment mechanisms to uncover gapes that attackers can exploit [15]-[18]. However, some works use the runtime checks (8 out of 18 papers) or testing (10 out of 18 papers) as a mechanism for detecting vulnerability. This does not prove that they are not useful, but it could be that it is more efficient to assess any errors during the coding which may include those vulnerabilities that revealed during requirement and design. Perhaps, coupling two or more would intensify the assessment and could yield a better result.

8.4. Tools

Different tools are deployed to ease and automate the task of catching errors or vulnerabilities. Some of these tools are meant to test leakages in coding such as the Auto Inspect tool for black box testing, web application vulnerability and error scanner (WAVES) and so on, other tools such as web SSARI [19], MIMOSA and BEEP [20] are meant to discover vulnerabilities [21]. There is no tool that can be considered as a standard and each tool has different functionalities.

8.5. Conclusion and Recommendation

After discussing the results of this systematic literature review, the review has benefited the practitioners, researchers and policy makers of web application development projects. The unavailability of a standardized security techniques to guide the development of secure web applications suggests that more research needs to be done to determine what the appropriate development technique is, and what process is required. Similarly, the consistent reference to the OWASP Top 10 or risk assessment in the different studies under this review suggests to both practitioners and researchers that tightening security around the development lifecycle using different tools and techniques can be effective or easier to adopt than other methodology. However, other approaches such as security patterns and digital signatures are also crucial. Furthermore, policy makers and practitioners need to institutionalize, in their various projects, the culture of considering security at early phases and throughout the entire development lifecycle, with emphasis on requirement vulnerabilities. Therefore, this may suggest the need for further research on whether or not building security checks around requirement and coding is adequate. It is also important for developers and practitioners to be aware of the fact that automated tools for scanning vulnerabilities during development are not enough to have a bullet proof product, since errors sometimes are not perceived simply because they appear at certain times. Therefore, the grouping of manual and automated tools is necessary, especially requirement vulnerability analysis, runtime checks, and other tools. Therefore, this SLR has been conducted based on systematic literature review (Kitchenham 2007) guidelines. The review aims to determine what development tools are available for security mitigation in web applications, what approach or technique is used, and what security problems these approaches or techniques have addressed. Based on these objectives, we extract 623 papers from 4 reputable scientific sources and then exclude 567 papers based on an initially defined inclusion and exclusion criteria. Hence, we consider 56 key papers for in-depth review. The review results suggest that different papers use different security development techniques to develop secure web applications. For this reason, it shows that there is no standard or preferred development technique for web application security. However, OWASP Top 10 seems to have gained more attention, probably due to the involvement of multiple stakeholders when discussing security viewpoints. This ensures proper understanding of security requirements rather than enforcing it on a few members of the development team. Similarly, many papers studied in this review use different techniques to tighten security during the phases of development. This may suggest the universality, effectiveness or ease of use of tools and techniques in dealing with different kinds of vulnerabilities in the web application. Furthermore, applying security checks across the entire life cycle of the development process has gotten little attention, even though it is the right thing to do when it comes to security assurances on the web. This points to the need for more work that will consider security across the whole process of software development. Finally, the result of this review has shown that many studies emphasize vulnerability assessment using various kinds of tools to detect gapes that attackers can exploit when an application

is subjected to internet. There is a lack of study which represents current state-of-the-art of empirically supported work in this area. To tie this gap, this paper presents the plan for conducting a systematic literature review in order to present the current position of the field, possible gaps and directions for future research. This study will help researchers and practitioners in the area of web applications to find out more established practices and techniques, and to know the problems that need more empirical assessment.

Acknowledgements

We want to pay special thanks to Ms. Mamoona Humayun for her supporting role in conducting this research work and spent her precious time to help as a supervisor. We also thank anonymous reviewers who gave us very valuable feedback to improve this work.

References

[1] Ge, X., Paige, R.F., Polack, F.A., Chivers, H. and Brooke, P.J. (2006) Agile Development of Secure Web Applications. *Proceedings of the* 6*th International Conference on Web Engineering*. Palo Alto, 11-14 July 2006, 305-312.

[2] Norwawi, N.M. and Selamat, M.H. (2011) Secure E-Commerce Web Development Framework. *Information Technology Journal*, **10**, 769-778.

[3] McGraw, G. and Viega, J. (2002) Building Secure Software. In RTO/NATO Real-Time Intrusion Detection Symp.

[4] Mouratidis, H., Jürjens, J. and Fox, J. (2006) Towards a Comprehensive Framework for Secure Systems Development. Advanced Information Systems Engineering. Springer, Berlin Heidelberg, 48-62. http://dx.doi.org/10.1007/11767138_5

[5] Keele, S. (2007) Guidelines for Performing Systematic Literature Reviews in Software Engineering. Technical Report, EBSE Technical Report EBSE-2007-01, 1-57.

[6] Cachia, E. and Micallef, M. (2007) A Multi-Tier, Multi-Role Security Framework for E-Commerce Systems. 14*th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems*, Tucson, 26-29 March 2007, 422-432.

[7] Lipner, S. (2004) The Trustworthy Computing Security Development Lifecycle. 20*th Annual Computer Security Applications Conference*, Washington, 6-10 December 2004, 2-13. http://dx.doi.org/10.1109/csac.2004.41

[8] Sulayman, M. and Mendes, E. (2009) A Systematic Literature Review of Software Process Improvement in Small and Medium Web Companies. Advances in Software Engineering. Springer, Berlin Heidelberg, 1-8. http://dx.doi.org/10.1007/978-3-642-10619-4_1

[9] Shar, L.K. and Tan, H.B.K. (2012) Automated Removal of Cross Site Scripting Vulnerabilities in Web Applications. *Information and Software Technology*, **54**, 467-478. http://dx.doi.org/10.1016/j.infsof.2011.12.006

[10] Avancini, A. and Ceccato, M. (2013) Comparison and Integration of Genetic Algorithms and Dynamic Symbolic Execution for Security Testing of Cross-Site Scripting Vulnerabilities. *Information and Software Technology*, **55**, 2209-2222. http://dx.doi.org/10.1016/j.infsof.2013.08.001

[11] Jang, Y.S. and Choi, J.Y. (2014) Detecting SQL Injection Attacks Using Query Result Size. *Computers & Security*, **44**, 104-118. http://dx.doi.org/10.1016/j.cose.2014.04.007

[12] Goseva-Popstojanova, K., Anastasovski, G., Dimitrijevikj, A., Pantev, R. and Miller, B. (2014) Characterization and Classification of Malicious Web Traffic. *Computers & Security*, **42**, 92-115. http://dx.doi.org/10.1016/j.cose.2014.01.006

[13] Shahriar, H., Weldemariam, K., Zulkernine, M. and Lutellier, T. (2014) Effective Detection of Vulnerable and Malicious Browser Extensions. *Computers & Security*, **47**, 66-84. http://dx.doi.org/10.1016/j.cose.2014.06.005

[14] Scholte, T., Balzarotti, D. and Kirda, E. (2012) Have Things Changed Now? An Empirical Study on Input Validation Vulnerabilities in Web Applications. *Computers & Security*, **31**, 344-356. http://dx.doi.org/10.1016/j.cose.2011.12.013

[15] Woo, S.W., Joh, H., Alhazmi, O.H. and Malaiya, Y.K. (2011) Modeling Vulnerability Discovery Process in Apache and IIS HTTP Servers. *Computers & Security*, **30**, 50-62. http://dx.doi.org/10.1016/j.cose.2010.10.007

[16] Awoleye, O.M., Ojuloge, B. and Ilori, M.O. (2014) Web Application Vulnerability Assessment and Policy Direction towards a Secure Smart Government. *Government Information Quarterly*, **31**, S118-S125. http://dx.doi.org/10.1016/j.giq.2014.01.012

[17] Buja, G., Bin Abd Jalil, K., Bt Hj Mohd Ali, F. and Rahman, T.F.A. (2014) Detection Model for SQL Injection Attack: An Approach for Preventing a Web Application from the SQL Injection Attack. *Proceedings of the* 2014 *IEEE Symposium on Computer Applications and Industrial Electronics* (*ISCAIE*), Penang, 7-8 April 2014, 60-64.

[18] Salas, M.I.P. and Martins, E. (2014) Security Testing Methodology for Vulnerabilities Detection of XSS in Web Services and WS-Security. *Electronic Notes in Theoretical Computer Science*, **302**, 133-154. http://dx.doi.org/10.1016/j.entcs.2014.01.024

[19] Shar, L.K. and Tan, H.B.K. (2013) Predicting SQL Injection and Cross Site Scripting Vulnerabilities through Mining Input Sanitization Patterns. *Information and Software Technology*, **55**, 1767-1780. http://dx.doi.org/10.1016/j.infsof.2013.04.002

[20] Katkar Anjali, S. and Kulkarni Raj, B. (2012) Web Vulnerability Detection and Security Mechanism. *International Journal of Soft Computing and Engineering* (*IJSCE*), **2**, 237-241.

[21] Wang, S., Gong, Y., Chen, G., Sun, Q. and Yang, F. (2013) Service Vulnerability Scanning Based on Service-Oriented Architecture in Web Service Environments. *Journal of Systems Architecture*, **59**, 731-739. http://dx.doi.org/10.1016/j.sysarc.2013.01.002