Journal of King Saud University - Computer and Information Sciences

SECURITY TESTING OF WEB APPLICATIONS: A SYSTEMATIC MAPPING OF THE LITERATURE

--Manuscript Draft--

Manuscript Number:	JKSUCIS-D-21-01459
Article Type:	Review Article
Keywords:	Security testing, web application security, systematic mapping; systematic literature mapping; systematic literature review
Abstract:	A large number of testing techniques, tools and frameworks have been proposed by both practitioners and researchers to effectively and efficiently test the security of web applications. As the number of papers increases in the security of web applications and this research area matures, reviewing and getting an overview of this area is getting challenging for a practitioner or a new researcher. Our objective is to summarize the state-of-the-art in web application security testing which could benefit practitioners to potentially utilize that information. We review and structure the body of knowledge related to web application security testing in the form of a systematic literature mapping (SLM). This review paper provides an overview of web application security testing with different focused headings. These headings cover contribution types, web security testing tools and their sub features, specific questions/features to the security testing such as vulnerability types, system under testing (SUT) focused headings and more. The results of this study would benefit researchers and developers working on web application security testing. Thanks to this paper, these researchers could utilize the all results and use them to catch the trend of web application security testing and secure development.

SECURITY TESTING OF WEB APPLICATIONS: A SYSTEMATIC MAPPING OF THE LITERATURE

Murat Aydos
Assoc. Prof., Computer Eng. Dept
Hacettepe University, Turkey
maydos@hacettepe.edu.tr

Çiğdem Aldan
Invicti Security,
US
cigdem.aldan@invicti.com

Evren Coşkun*
Turkish Aerospace
Industries Inc., Turkey
ecoskun@tai.com.tr

Alperen Soydan
Turkish Aerospace
Industries Inc., Turkey
alperen.soydan@tai.com.tr

Abstract:

Context: Web application security is a main component of any web-based business. Web applications are subject to attacks from different locations at various levels of scale and complexity. In this context, a large number of testing techniques, tools and frameworks have been proposed by both practitioners and researchers to effectively and efficiently test the security of web applications.

Objective: As the number of papers increases in the security of web applications and this research area matures, reviewing and getting an overview of this area is getting challenging for a practitioner or a new researcher. Our objective is to summarize the state-of-the-art in web application security testing which could benefit practitioners to potentially utilize that information.

Method: We review and structure the body of knowledge related to web application security testing in the form of a systematic literature mapping (SLM). As part of this study, we pose four sets of research questions, define selection and exclusion criteria, and systematically develop and refine a classification schema. The initial pool consisted of 154 articles. Systematic voting was conducted among the authors regarding the inclusion/exclusion of articles. As a result, there were 80 technical articles in our final pool.

Results: This review paper provides an overview of web application security testing with different focused headings. These headings cover contribution types, web security testing tools and their sub features, specific questions/features to the security testing such as vulnerability types, system under testing (SUT) focused headings and more.

Conclusion: The results of this study would benefit researchers working on web application security testing. Also, it could be useful for developers who discuss application security while they develop web applications. Thanks to this paper, these researchers could utilize the all results and use them to catch the trend of web application security testing and secure development.

Keywords: Security testing, web application security, systematic mapping; systematic literature mapping; systematic literature review

* Corresponding author: Evren Coşkun Turkish Aerospace Industries Inc., Turkey ecoskun@tai.com.tr

CONTENTS

1 INTRODUCTION

Web application is an increasingly important instrument for business. Web applications are used by enterprises and organizations to provide services from an organization to an individual for a wide variety of uses. Commonly used web applications include telecommunication, healthcare, education, finance, e-commerce and information technology domain. Due to the increasing heterogeneity and complexity of web applications, their security is often crucial to make business and organization successful. Thus security testing plays an important role for web applications and it is an essential task.

Web applications have become very popular in recent years and due to extensive use of web applications they contain defects. Some of these defects include vulnerabilities and it increases the occurrence of many attacks that weaken web security and disrupt the integrity of web applications, introducing risks to organizations. Security testing is a process. Its aim is to find security vulnerabilities in web applications and ensure that web application features are secure. It is important to know that web security testing is testing business logic, input validation, output encoding, authentication and authorization issues to avoid common vulnerabilities such as SQL Injections, Cross-site Scripting (XSS) and more.

Security vulnerabilities are listed in the OWASP (Open Web Application Security Project) Top 10 list. OWASP is an open community and organizations use OWASP Top 10 lists to develop secure web applications that are only possible when using a secure SDLC (Software development lifecycle). OWASP Top 10 list as well as thousands of other known vulnerabilities can be detected with web application security testing tools automatically or by security experts manually. Black Box and White Box testing are mostly used in security testing tool development to find the majority of the vulnerabilities and security flaws in web applications. In black testing, the test cases are given to software and outcomes are checked to predict the vulnerabilities or errors while white box testing is performed to examine the faulty or vulnerable code (Pariwish et al., 2019).

To address the above information and to identify needs about security testing of web applications and to fill the gap by reviewing vulnerabilities, web application detection tools and their techniques, applied web applications and their technologies, we conducted a systematic literature review (SLR) on the papers written by academic and industry researchers and we present a review summary in this article. To systematically review and get an overview of studies in a given research area, Systematic Literature Review (SLR) and Systematic (Literature) Mapping (SLM or SM) studies are the established approaches.

Our review pool included 80 technical papers published in conferences and journals. The first paper was published in 2005 and this review includes all the papers until the end of 2020.

The SLR is organized as follows. Background and related work is presented in Section 2. We describe the research method and the planning phase of our review in Section 3. Section 4 presents the results of the literature review. Section 5 summarizes the findings and potential benefits of this review. Finally, in Section 6, we draw conclusions, and suggest areas for further research. In the appendix, we show the list of the primary studies reviewed in this survey.

2 BACKGROUND AND RELATED WORK

This work is a review study in the context of web application security testing. In this section, to understand the rest of the paper we first provide an overview of web application security testing and then we discuss a few of related works which are existing systematic literature review papers in this area.

2.1 AN OVERVIEW OF WEB APPLICATION SECURITY TESTING

Web application security is a main component of any web-based business. Due to the nature of the Internet, web applications are subject to attacks from different locations at various levels of scale and complexity. In this context, a large number of testing techniques, approaches, tools and frameworks have been proposed by both practitioners and researchers over the past few decades to effectively and efficiently test the security of web applications.

Web application is a form of software typically used through web browsers, and it commonly uses a combination of a server-side script and a client-side script. It can be written in all kinds of languages, and run on every kind of operating system. All functionality of web applications is communicated with web servers using Hypertext Transfer Protocol (HTTP) and its results are typically formatted in HTML. Due to the increasing heterogeneity and complexity of web applications, their security is often crucial to make business and organization successful. Thus security testing plays an important role for web applications and it is an essential task. While testing web applications in aspects of security, it's best to use well-known web application testing guides such as the OWASP (Open Web Application Security Project) guide.

The OWASP is a worldwide free and open community focused on improving the security of application software and OWAPS is in conjunction with OWASP top 10, the code review guide, the development guide and tools such as OWASP ZAP and the testing guide. The development guide will show your project how to architect and build a secure application, the code review guide will tell you how to verify the security of your application's source code, and the testing guide will show you how to verify the security of your running application ("OWASP. 4.0 Testing Guide," 2014). In the meanwhile, all of the OWASP tools, documents are open to anyone who is interested in developing secure applications. OWASP Top 10 list as well as thousands of other known vulnerabilities can be detected with web application security testing tools automatically or by security experts manually.

Based on a SLR study conclusion in this area, the mostly occurring risks are SQL following the cross-site scripting and sensitive data exposure (Pariwish et al., 2019). Before we go any further, it is important to know the meaning of vulnerability, exploit and threat. IETF defines these terms as follows (Shirey, 2007).

- *Vulnerability* is a flaw or weakness in a system's design, implementation, or operation and management that could be exploited to violate the system's security policy.
- *Exploit* is a piece of program or data that has been developed to attack an asset by taking advantage of system vulnerability.
- *Threat* is a potential for violation of security, which exists when there is an entity, circumstance, capability, action, or event that could cause harm.

Security testing is a process to test applications in terms of security functionalities. Security functions are related to confidentiality, integrity, availability, authentication, authorization, and non-repudiation. These terms can be defined as follows ("Web Application Security Testing," 2021):

- *Confidentiality,* i.e., the property that information is not disclosed to system entities (e.g., user, process, or device) unless they have been authorized to access the information.
- *Integrity,* i.e., the property whereby an entity has not been modified in an unauthorized manner.
- *Authentication,* i.e., the process of verifying the identity or other attributes claimed by or assumed of an entity (e.g., user, process, or device) or to verify the source and integrity of data.
- *Authorization,* i.e., access privileges granted to a user, program, or process or the act of granting those privileges.
- *Availability,* i.e., ensuring timely and reliable access to and use of information.
- *Non-repudiation,* i.e., the assurance that the sender of information is provided with the proof of delivery and the recipient is provided with the proof of delivery and the recipient is provided with proof of the sender's identity, so neither can later deny having processed the information.

The objective of security testing is to find potential vulnerabilities in applications and ensure that application features are secure from external or internal threats. Software security testing can be divided into security functional testing and security vulnerability testing. Security functional testing ensures whether software security functions are implemented correctly and consistent with security requirements based on security requirement specification. Security vulnerability testing is to discover security vulnerabilities as an attacker (Bhargav, Abhay, 2010).

Web security testing is testing business logic, input validation, output encoding, authentication and authorization issues to avoid common vulnerabilities such as SQL Injections, Cross-site Scripting (XSS) and more. Security testing can be carried out with combination of manual testing through security experts, and web application security testing tools automatically, and they complement each other. In order to perform security testing of web applications manually, security experts should have a good knowledge of HTTP protocol and OWASP Top 10 vulnerabilities as well as other known vulnerabilities. Security experts gather information about applications, test and report back security weaknesses, especially some of them cannot be found to be sufficient by automated security testing tools until the security of application is assessed. Besides, manual security testing is a time-consuming process and requires automated security testing tools. Automated security testing tools are good in finding vulnerabilities and they simulate the actions of security experts.

Black Box and White Box testing are mostly used in security testing tool development to find the majority of the vulnerabilities and security flaws in web applications. In black testing, the test cases are given to software and outcomes are checked to predict the vulnerabilities or errors while white box testing is performed to examine the faulty or vulnerable code (Shirey, 2007). Black box testing is also known as Dynamic Analysis Security Testing (DAST). DAST tools help developers and security experts to find and fix the majority of the vulnerabilities and security flaws in web applications. Similarly, white box testing is also known as Static Analysis Security Testing (SAST) since it has access to internal code and it helps developers write secure source code. The SAST tool scans static code, and reduces the number of vulnerabilities. Both DAST and SAST tools can fit in the development process.

An important part of web application security testing tools is that they are known to be error-prone and report false positives. The problem of false positives and false negatives are common for the automated web application security testing tools. If a vulnerability is reported by a web application security tool but in reality it is not existing, it is called as false positive. If an existing vulnerability is missed by a web application security testing tool, this behavior is called as false negative. Therefore, web application security testing tools should have low value for false positives and the false negatives.

2.2 RELATED WORK: OTHER REVIEW PAPERS IN THE AREA

Focusing on the web application security testing, we were able to find three review papers (secondary) (Bhargav, Abhay, 2010; Mohanty et al., 2019; Seng et al., 2018) which have been reported and we list these papers in Table 1. For each paper, its type, its publication year, number of paper reviewed and focus area were extracted. Based on the publication year, it seems that SLRs have recently started to appear in the web application security testing. As we can see in Table 1, all three papers focused on web application security testing but each paper studied web application security testing in a different perspective.

In our study, apart from these SLRs, our objective is to identify needs about security testing of web applications and to fill the gap by reviewing vulnerabilities, web application security testing tools and their techniques, web applications and their technologies. Therefore, our study is the most up-to-date and comprehensive review in this area by considering all 80 papers, published between 2005-2020.

Table 1: A list of review papers on the web application security testing

Paper title	Year	Num. of paper reviewed	Reference	Focus area
Analysis of Automated Web Application Security Vulnerabilities Testing	2019	30	(Bhargav, Abhay, 2010)	Web application vulnerabilities security testing
Security Testing of Web Applications Using Threat Modeling: A Systematic Review	2019	8	(Mohanty et al., 2019)	The different existing threat modeling techniques for detecting threats used for security testing
The approaches to quantify web application security scanners quality: A review	2018	90	(Seng et al., 2018)	The quality of web application security scanners' quality

3 RESEARCH METHOD

We conducted this study using well-known SLR and SLM guidelines in software engineering (B. Kitchenham, 2007; Kai et al., 2008; Petersen et al., 2015; Wohlin, 2014). The proven SLM process (Garousi et al., 2020, 2018) that lies at the basis of this SLM is outlined in Figure 1, which consists of three phases: Source selection, Classification scheme/map, Systematic literature mapping. We discuss these phases in the following sections.

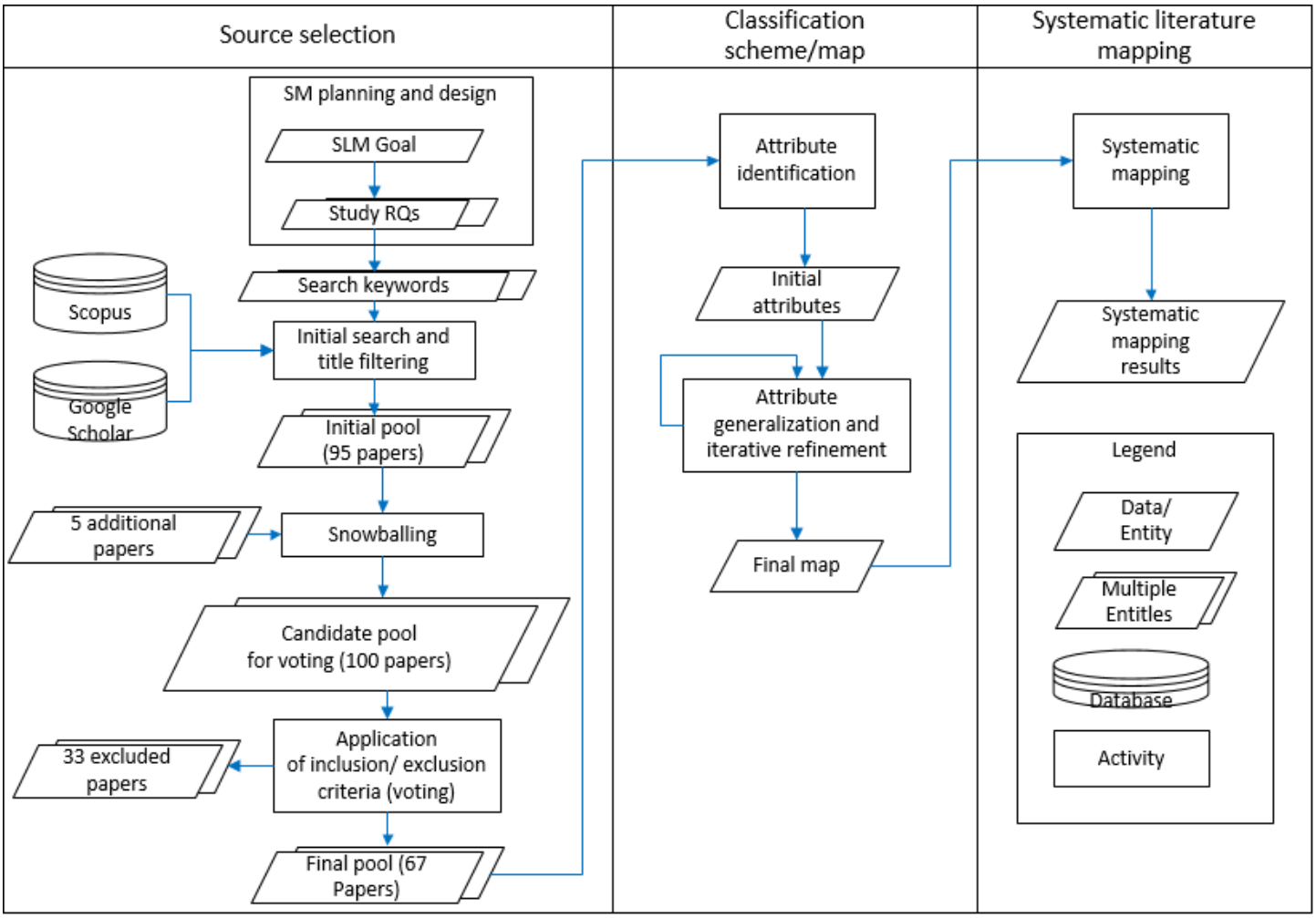


Figure 1: An overview of our SLM process

3.1 GOAL AND RESEARCH QUESTIONS

The goal of this study is to classify, review and analyze the knowledge in the field of web applications security testing to find out the trends and directions. Furthermore, we aim to detect occasions for future research for both researchers and practitioners. We draw a research scope and boundary for our systematic mapping in order to ensure a clear focus. We decided to only include the journal or conference papers on security testing of web applications and exclude all the remotely related papers, e.g., security of web services. Based on the goal that we focused on, we discussed the following research questions RQs grouped as four categories:

Group 1 – Common to all systematic literature mapping studies

RQ1.1-Type of contribution: What are the different types of contributions presented in the papers? What is the main contribution of Web Application Security Testing studies with respect to techniques/methods, tools, processes and measurements?

RQ1.2-Type of research facet: What types of research methods have been used in papers for Web Application Security Testing studies? (Solution Proposal, Weak Empirical Study, Strong Empirical Study, Experience Papers, Philosophical Papers, Opinion Papers, Other)

Group 2 – Specific to the web security testing tool

RQ2.1- Web security testing tools used: Which testing tools are used for security testing? We examined popularity and frequency of usage of tools by years.

RQ2.2-License type: What is the testing tool licensing type? There are lots of choices for tool licensing, we focused on tool's licensing that academic and industrial researchers used.

RQ2.3-Analysis technique: Is static analysis or dynamic analysis used for security testing? Web application security testing mainly focuses on two types of analysis techniques. Thus, we discussed these types as Static Analysis – SAST (Static Application Security Testing), Dynamic Analysis – DAST (Dynamic Application Security Testing) and Other and we tabulated these data.

RQ2.4-Automation level: What is the automation level for testing? We focused on changing the automation level by years. We want to show what is the effect of technological progress on automation level.

RQ2.5-Accuracy focusing: Have the false positives and false negatives problem in web security applications been addressed?

Group 3 – Specific to the security testing

RQ3.1-Vulnerability type: What types of vulnerabilities have been tested and/or detected according to OWASP Top 10 and others? We examined whether the vulnerabilities addressed by years are in these years OWASP Top 10.

RQ3.2-Violation type: What is the type of violations that are encountered? Which types of violations are there and which ones are included in the studies?

RQ3.3-SDLC focusing: Does the paper consider software development life cycle?

Group 4 – Specific to system under testing (SUT)

RQ4.1-Web application type: What is the tested web application type? For web application security testing there are different types of web applications, thus we mainly focused on five types of web application as custom-made web application, educational vulnerable web application, web application security scanner test sites, open-source test application framework, real-world web application and tabulate academic, industrial and collaborative numbers.

RQ4.2-SUT number and name: What is the name of SUT and how many SUTs are included?

RQ4.3-Developing language of SUT: Which languages are used to develop SUT objects?

RQ4.4-Connection type: Which connection types are used for SUT?

3.2 SOURCE SELECTION AND SEARCH QUERY DEFINITION

We selected electronics databases in the search process using the established process for performing SLR study in security testing of web application. Google Scholar and Scopus (www.scopus.com) are widely used in review studies (B. Kitchenham, 2007) and they were used to search. The coverage of google scholar is much higher to be used alone for systematic review but it requires some improvement in the advanced search and "it should not be used alone for systematic literature reviews searches" (Haddaway et al., 2015). Therefore, in addition to Google Scholar, Scopus was used to find some missing papers.

We developed our search strings and we used title based searching over the electronics databases. When we used "Security AND (test OR testing) AND Web), our search results returned relevant results as well as irrelevant results. To ensure maximizing chances of including all relevant studies we used the search string by adding "Web Application OR Web" terms. Our final search string for Google Scholar was: "Security AND (test OR testing) AND (Web Application OR Web App OR Web)".

Additionally, we used the same search string in Scopus and it returned 3 different candidate papers. The scopus search string is "TITLE-ABS-KEY (security AND (test OR testing) AND (Web Application OR Web App OR Web).

During December 2020, January and February 2021, we conducted the search phase and we included papers published until the end of 2020. Data extraction based on inclusion criterias was conducted in the same period.

As our paper search and selection process we applied title based selection at first and then we removed duplicate papers and we excluded venues which are report, book chapter, patent paper, thesis by considering inclusion/exclusion criterias (Section 3.3). Therefore, we balanced precision, rigor and efficiency of papers. Figure 2 shows a screenshot of our search

process using Google Scholar. The green marked paper is a relevant candidate paper while the red marked paper is a candidate of irrelevant paper. We prepared our paper pool by adding potential relevant papers.

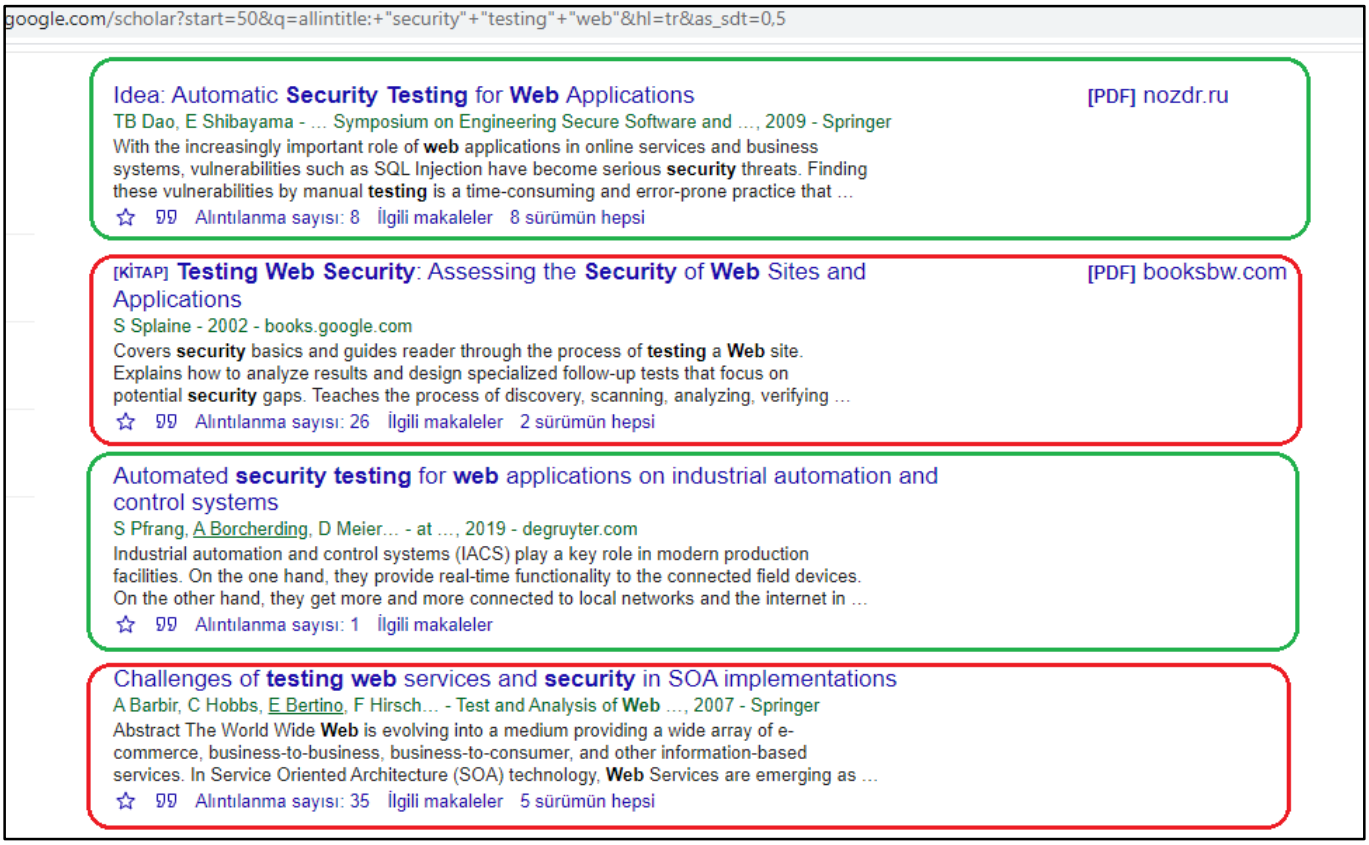


Figure 2: Screenshot of our search activity using Google Scholar

When searching using Google Scholar, we had to limit searching results since Google Scholar provided over 1.5 million documents for the above keyword. It was difficult to go through all of the results. As per our observation when we checked all results by starting from the first page and the relevance of results decreased in the following pages. It was good news to restrict the search results. Therefore, we checked the first n pages and we continued to look further if necessary. Besides, we verified retrieved papers by using the "allintitle" filter of Google Scholar and found some additional papers. At the end of our initial search and title filtering, our initial pool consisted of 148 papers. In order to include all the related papers as much as possible, we use forward and backward snowballing as recommended by systematic review guidelines (Wohlin, 2014) and proved systematic reviews (Garousi et al., 2018, 2019b, 2020). Thanks to this, we included 6 more articles. After compiling an initial pool of 154 "candidate" articles, a systematic voting method was applied among authors according to the defined including/exclusion criteria (as discussed in the next section) to create the final pool of primary studies.

3.3 EXCLUSION AND INCLUSION CRITERIA

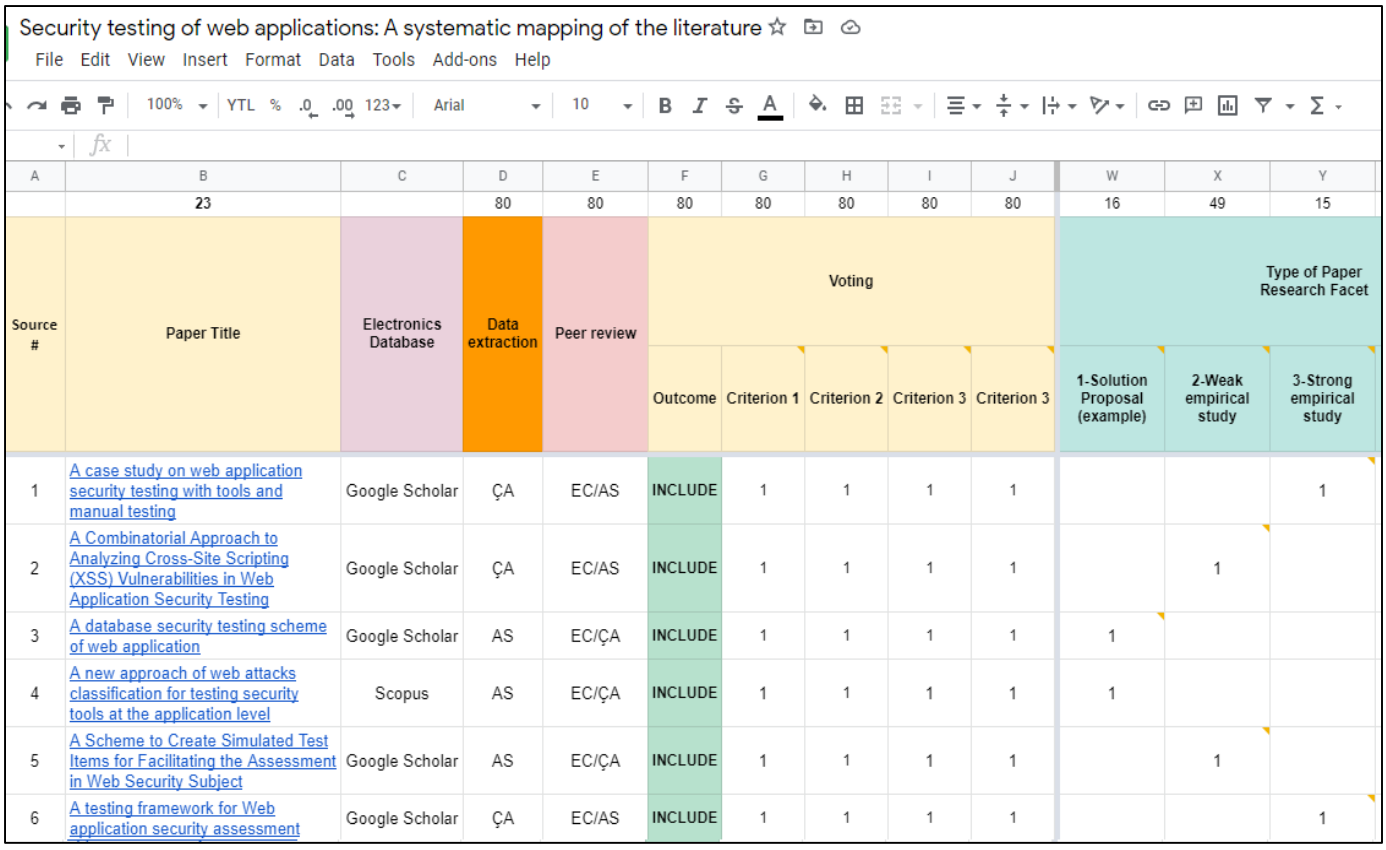
We identified the inclusion and exclusion criteria to provide that all the related papers are included and the ones that are out of coverage are not included. The inclusion criteria were:

1. Is the paper fully accessible on the internet?
2. Is the paper written in English?
3. Does the source focus on Security testing of web applications?
4. Does the source include a relatively sound validation?

The answers to the questions were in two ways. "Yes" for value=1 and "No" for value=0. The papers that have 1 for every criterion were included and the others were not included. Therefore, 74 papers were excluded based on the criteria mentioned above and the evaluations of these take place on the online spreadsheet (Aydos et al., 2021), and are also listed in the appendix. For instance, the paper entitled as "Enhanced Security Measurement of Web Application Testing by Outsourcing", Kyong-Ho and Lee (Kyong-Ho and Lee, 2015) were not included because the main paper was in Korean, whereas its title and abstract were just in English.

3.4 FINAL POOL OF PAPERS AND THE ONLINE REPOSITORY

The pool was finalized with 80 papers. The references for the final pool of 80 papers can be seen in an online repository (bit.ly/3vnvS5M) and are also listed in the appendix.. The format of "[Si]" is used in this systematic mapping paper in order to refer to the source papers in the pool as listed in the online spreadsheet (see the screenshot in Figure 3).



Security testing of web applications: A systematic mapping of the literature

File Edit View Insert Format Data Tools Add-ons Help

	A	B	C	D	E	F	G	H	I	J	W	X	Y
		23		80	80	80	80	80	80	80	16	49	15
	Source #	Paper Title	Electronics Database	Data extraction	Peer review	Voting					Type of Paper Research Facet		
						Outcome	Criterion 1	Criterion 2	Criterion 3	Criterion 3	1-Solution Proposal (example)	2-Weak empirical study	3-Strong empirical study
1	1	A case study on web application security testing with tools and manual testing	Google Scholar	ÇA	EC/AS	INCLUDE	1	1	1	1			1
2	2	A Combinatorial Approach to Analyzing Cross-Site Scripting (XSS) Vulnerabilities in Web Application Security Testing	Google Scholar	ÇA	EC/AS	INCLUDE	1	1	1	1		1	
3	3	A database security testing scheme of web application	Google Scholar	AS	EC/ÇA	INCLUDE	1	1	1	1	1		
4	4	A new approach of web attacks classification for testing security tools at the application level	Scopus	AS	EC/ÇA	INCLUDE	1	1	1	1	1		
5	5	A Scheme to Create Simulated Test Items for Facilitating the Assessment in Web Security Subject	Google Scholar	AS	EC/ÇA	INCLUDE	1	1	1	1		1	
6	6	A testing framework for Web application security assessment	Google Scholar	ÇA	EC/AS	INCLUDE	1	1	1	1			1

Figure 3: A screenshot from the online spreadsheet of source papers (bit.ly/3vnvS5M)

To visually see the growth of the web application security testing field, the publication years of papers are depicted and compared the trend with data from four other review studies, e.g., a systematic review on unsupervised learning techniques for software defect prediction (Li et al., 2020), a systematic review on time pressure in software engineering (Kuutila et al., 2020), a survey on software testability (Garousi et al., 2019a), and a SLR on machine learning techniques for software maintainability prediction (Hadeel and Roper, 2020). Note that the data for the other four areas are not until year 2018, since the execution and publication timelines of those studies are in 2019 and 2020.

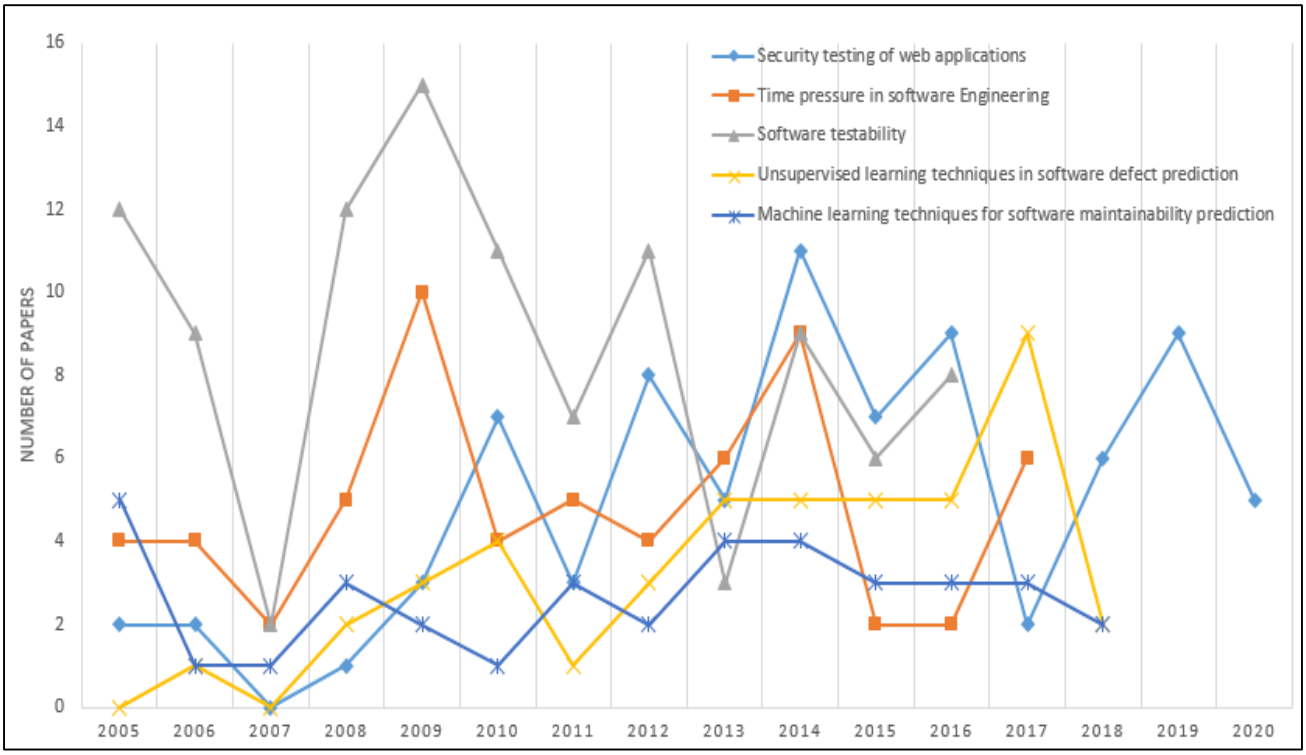


Figure 4: Growth of the security testing of web applications and comparison with data from four other review studies

3.5 CLASSIFICATION SCHEME

To answer all the RQs, a systematic map was developed. Subsequently, the data was extracted from sources in order to categorize them. In this section, progress of a systematic map will be analyzed.

Analyzing the included papers and defining of the beginning attributes were made in order to develop the systematic map. For obtaining the final map, in case of requirement, attribute generalization and iterative refinement were applied, and this was stated in Figure 1.

After defining related works, they were enlisted in a shared spreadsheet for enabling the next analysis. The following aim is to classify the papers for having a comprehensive impact in the study field and respond to RQs. Through iterative approach, wide interests were converted to the systematic map.

The final classification scheme created by using the steps mentioned before is stated in Table 2. Column 2 of this table shows the RQs while column 3 indicates attribute/aspect. Categories/metrics are identified in column 4. The situation of multiple selections for each attribute is stated in column 5. For instance, the value of the last column of RQ 1.2 (research type) is 'S' (Single). This shows that a study could solely be classified by a single research type. On the other hand, the value of the last column of RQ 1.1 (contribution type) is 'M' (Multiple). This shows that a contribution type could have more than one alternative (for example, method, tool, etc.). Categorizing of contribution type and research type in Table 3 was done with the aid of well-known SLR and SLM guidelines (B. Kitchenham, 2007; Kai et al., 2008; Petersen et al., 2015; Wohlin, 2014).

The categories for contribution and research types were mainly provided in Peterson et al.'s guideline paper (Petersen et al., 2015). Approach (method, technique), tool, model, framework, process, empirical results only (if "Empirical results" is selected, other options cannot be selected), "other" can be contribution types. A study can provide more than one contribution type. For example, research could provide a new approach and a tool for automating that approach.

"Solution proposal" is the least strict kind of the research-method types that are obtained from (Petersen et al., 2015). The solution is shown by a simple example or a good line of argumentation. Experimental analyses were classified in two groups: weak empirical studies (validation research) and strong empirical studies (evaluation research). The first type of research does not include any hypotheses, research questions, and does not apply any statistical tests. And if the research

includes these features mentioned in the first one, then it is defined as 'strong'. Peterson et al.'s guideline paper (Kai et al., 2008) includes the descriptions of the research in terms of experience papers, philosophical papers, and opinion papers.

Table 2: Systematic map

Group	RQ	Attribute/Aspect	Categories/metrics	Multiple/Single
Group 1: Common to all systematic literature mapping studies	1.1	Contribution type	Method/Technique/Approach, Tool, Model, Metric, Process, Empirical Study(ONLY), Other	Multiple
	1.2	Research type	Solution Proposal, Weak empirical study, Strong empirical study, Experience Papers, Philosophical Paper, Opinion Papers, Other	Single
Group 2: Specific to the web security testing tool	2.1	Web security testing tools	Paros, WebScarab, Fortify, JBroFuzz, Acunetix, Netsparker, BurpSuite, ATUSA, OWASP ZAP, WebFuzz, WAVES, SQLMap, Other	Multiple
	2.2	License type	Commercial, Commercial Free, Academia, Open Source	Multiple
	2.3	Analysis technique	Static Analysis – SAST, Dynamic Analysis – DAST, Other	Multiple
	2.4	Automation level	Manual, Semi-automatic, Automatic	Multiple
Group 3: Specific to the security testing	3.1	Vulnerability type	Injection, Broken Authentication, Sensitive Data Exposure, XML External Entities (XXE), Broken Access Control, Security Misconfiguration, Cross-Site Scripting, Insecure Deserialization, Using Components with Known Vulnerabilities, Insufficient Logging and Monitoring, Other	Multiple
	3.2	Violation type	Other	Single
	3.3	Accuracy focusing	Focused or not	Single
	3.4	SDLC focusing	Focused or not	Single
Group 4: Specific to systems under testing (SUT)	4.1	Web application type	Custom made web application, Educational vulnerable web application, Web application security scanner test sites, Open-source test application framework, Real-world web application	Multiple
	4.2	SUT number and name	Other	Multiple
	4.3	Developing language of SUT	.NET, PHP, Java, JavaScript, ActiveX, Python, Other	Multiple
	4.4	Connection type	HTTP/HTTPS	Multiple

As for the types of web application security testing vulnerability types used (RQ 3.1), we grouped vulnerability types as OWASP Top 10 and Others which are often implemented in web application security testing researches. Also, we grouped web application type as custom made web application, educational vulnerable web application, web application security scanner test sites, open-source test application framework, real-world web application and this grouping was our approach for applications that we discuss in the papers.

The analysis techniques used for web application security testing presented in the paper was classified as Static Analysis – SAST: If there is a static analysis tool usage or manual analysis of source code, Dynamic Analysis – DAST: If there is a dynamic analysis tool or tester tested web application with DAST techniques, Other: Other than all approaches from SAST and DAST.

As mentioned previously, for obtaining the categories for every attribute in the systematic map mentioned in Table 2, attribute generalization and iterative refinement were used and the classifications were indicated in the study. Every classification seen in several studies was identified as a new classification in the mutual group, on the other hand, they were put in a relevant class defined as 'Other'.

3.6 DATA EXTRACTION AND REVIEW

When the systematic map (classification scheme) was present, every member in our research team extracted data from the subgroup of the studies that were assigned for him or her. Traceability links were provided, and they were attached to the extracted data to relevant info in the studies for providing the justification of the process of every classification.

Figure 5 indicates a photo of our online spreadsheet which is used for facilitating collective endeavor and categorizing the studies through identifiable connections as mentioned. This photo indicates the info for RQ 1.1 (Contribution type). In this, one of the members of our research team has positioned the exact text in the article as the identifiable link for quality assurance of data extractions and enabling peer reviewing.

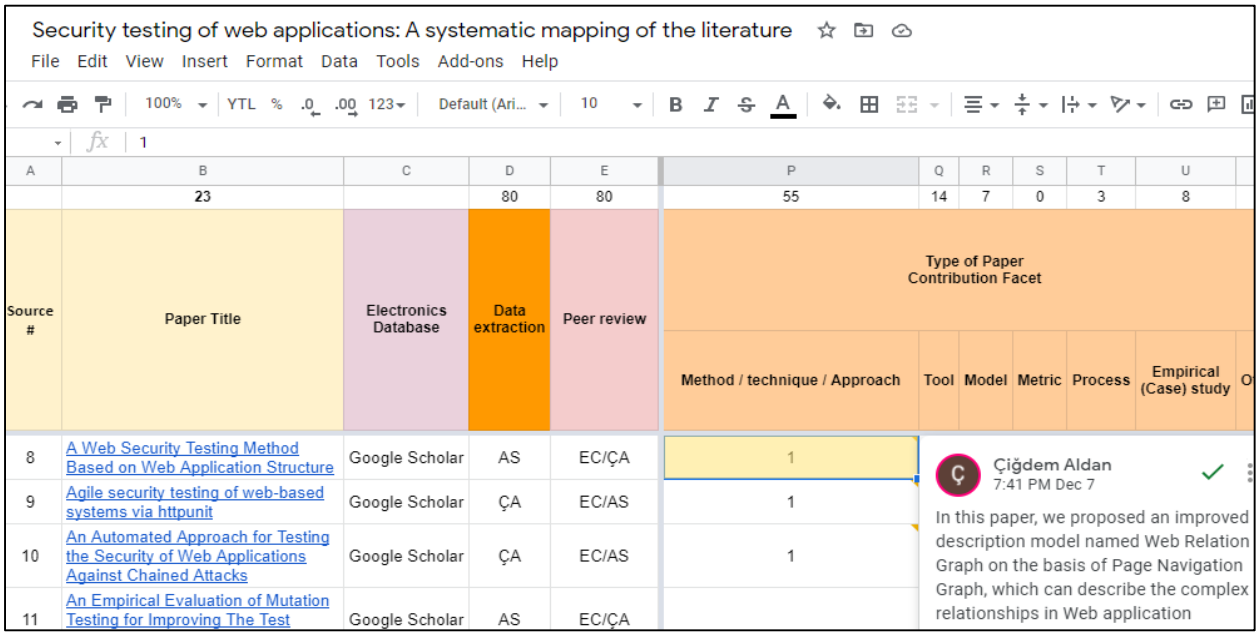


Figure 5- A screenshot from the online spreadsheet (bit.ly/3vnvS5M)

After completing all the data extractions, we applied peer review for the conclusions of the assessments and extractions made by each member by systematic peer reviewing. When disputes occurred, consultations were held for having consent. This action was done for providing a high performance for the extracted data and conclusions. These review operations can be seen from the comment history button in the online repository.

4 RESULTS

4.1 GROUP 1: COMMON TO ALL SYSTEMATIC LITERATURE MAPPING STUDIES

4.1.1 RQ 1.1: Types of papers by contribution facet

Figure 6 indicates the categorization of research through contribution facets. Most of the research in this field provided methods/approaches, apart from just a few different ones as contribution types (i.e. models or processes). Notice that it was mentioned in the structure of the systematic map (Section 3.5 & Table 2), because every research might have different contribution facets, a research could be appointed with more than one category in Figure 6.

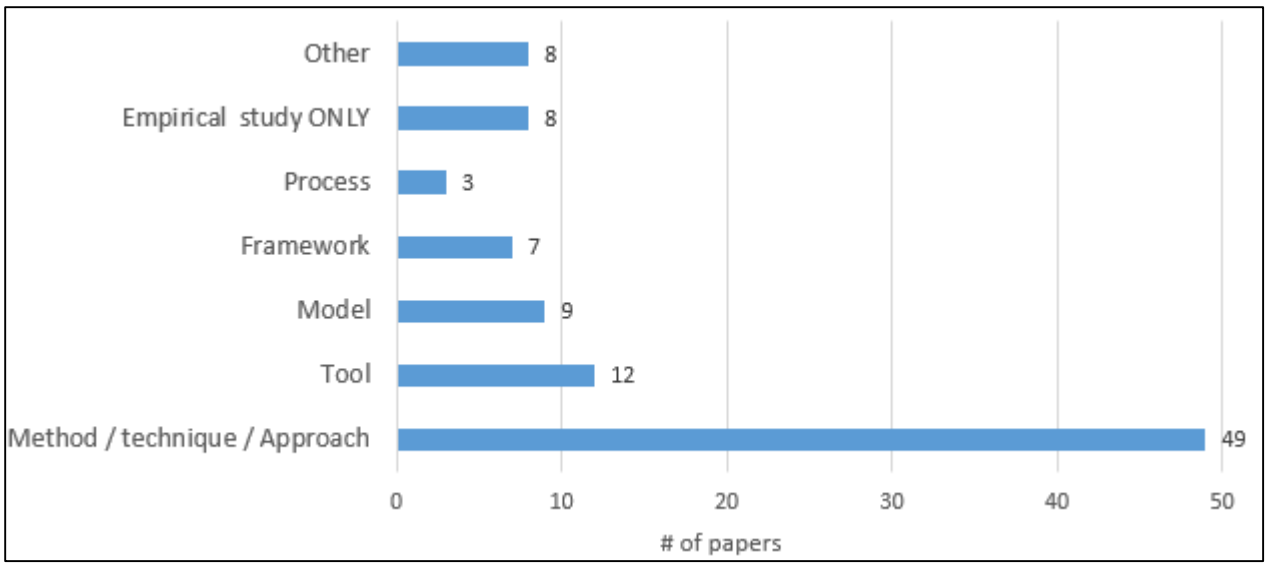


Figure 6: Categorization of the papers by contribution types

Approaches, methods, and techniques were set in one type, because they have similar concepts. For implementing a security testing of web applications, all of them suggested a "way". 49 of the 80 studies in the pool provided security testing approaches/methods/techniques for web applications. As it is seen in Figure 6, this set is the greatest type in the pool for contribution facets.

12 of the 80 studies (15%) in the pool provided tools to make the presented approaches for easier implementation. Different viewpoints of web application security testing tools will be detailed in Section 4.2.

8 of the 80 studies in the pool, [S1, S11, S21, S22, S46, S73, S76, S79], were empirical results only. Firstly, [S1] illustrates a case study on conducting security testing on Tunestore which was tested using some tools such as Paros, WebScarab, JBroFuzz, Acunetix, and Fortify. [S11] focuses on an empirical evaluation of mutation testing for Improving the test quality of web application's security. [S21] presents a questionnaire which reviews the benefits of Web Applications Security Testing for Sri Lankan SMEs. [S22] demonstrates challenges & problems in security testing of Web applications at software companies in India. [S46] is a case study on a recently introduced combinatorial testing methodology used for web application security. [S73] presents the results of the experiment about the security level of the top 10 popular free web-mail. [S76] uses open source software to identify, implement and test non-functional requirements of web applications which are important to users. Lastly, [S79] exposes the growing importance of adequate training at higher education institutions for web application security testing in industry. Therefore, in order to provide meaningful data that may be used by higher education institutions, a 12-question survey was distributed to software quality analyst professionals.

9 papers (~11% of the pool), [S6, S8, S31, S48, S50, S51, S67, S68, S77], provide models to support web application security testing. [S6] proposes "Topic Model", the knowledge base selects the best injection patterns according to experiences. [S8] proposes an improved description model named Web Relation Graph. [S31] improves model-based testing by combinatorial testing. [S48] presents ontology-driven security testing of web applications. This model represents an attack ontology that serves as the first step in a test generation process. P[S50] focuses on planning-based security testing of web applications with attack grammars. This model improves a high degree of extendibility and configurability and as well deals with limits of traditional graphical representations. [S51] proposes the model in the planning domain definition language (PDDL). The proposed model serves as an input for the planner. [S67] presents a software security testing model. It extends the traditional security testing process in order to defects behavior analysis. [S68] proposes an improved test model for Web controls' vulnerability based on fuzzing. Similarly, [s77] proposes a model for the running project in the financial institutions websites.

7 of the 80 studies (~9%) in the pool, [S6, S8, S16, S32, S43, S59, S77], provided frameworks to improve web application security testing. [S6] describes a black-box testing framework for Web application security assessment. [S8] proposes a

new security testing framework on the basis of vulnerability-related paths. [S16] focuses on a framework for testing and detecting SQL injection vulnerabilities in web applications. [S32] proposes an overall security testing framework for web-based applications. [S43] introduces a formal and flexible model-based security testing framework. [S59] proposes the framework for integrating the security testing procedures so that inexperienced software testers are able to produce better reports. Lastly, [S77] focuses on Vulnerability Assessment and Penetration Testing.

The contributions of 3 papers [S23, S39, S57] were related to process. [S23], proposes the continuous security testing procedure which is using test cases reusability to increase security test efficiency. [S39] focusses on threat assessment which is a controlled process in a group to assess the risk masqueraded. [S57] proposes a guidance that can be used by software analysts though the process of security requirements elaboration.

8 papers (10% of the pool), [S3, S5, S6, S15, S25, S58, S69, S71], present "Other" types of contribution. For instance, [S15] provides a taxonomy of known types of web application vulnerabilities. Different schemas for web applications security were presented in [S3, S5]. Entitled as "Security Sensitive Data Flow Coverage Criterion for Automatic Security Testing of Web Applications", [S58] presents a coverage criterion called "security sensitive data flow coverage".

4.1.2 RQ 1.2: Types of papers by research facet

The cumulative trend of mapping of studies by research facet are shown in Figure 7. Weak empirical studies (validation research), which have not yet been implemented in practice but in the lab environment, have the largest share in the pool with 63.7% (49 of 80). 16 of the 80 studies in the pool are solution proposals which present applicability of the solution by a small example without detailed empirical studies. 15 papers in the pool are strong empirical studies.

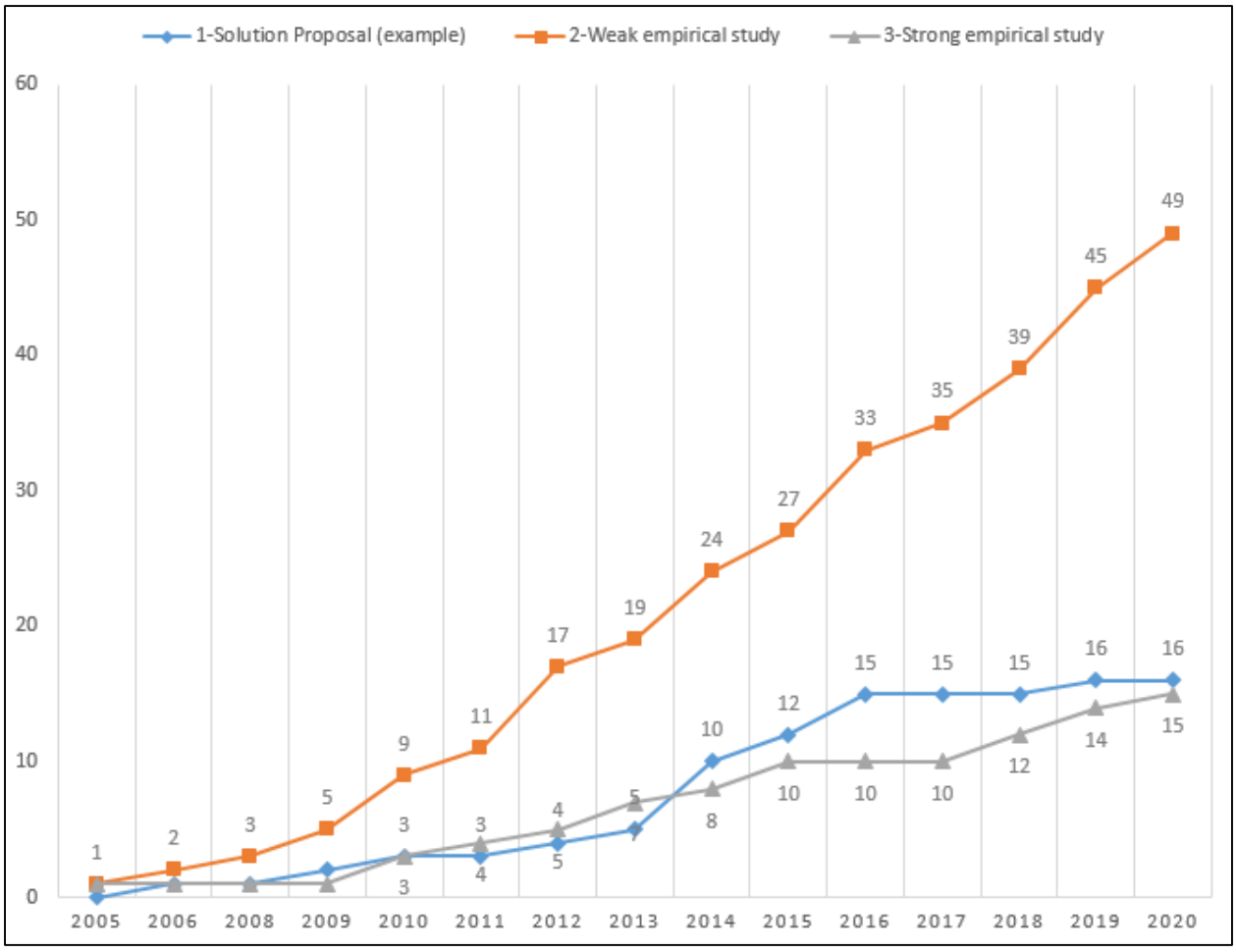


Figure 7: Cumulative trend of web application security studies by research facet

A few examples of "strong" empirical studies are provided because those sources are the most attentive ones. Entitled as "Search-based security testing of web applications", [S55] proposes a technique to automatically detect SQL injections with prototype which names BIOFUZZ and then presents an empirical study with two research questions:

- RQ1: "Is BIOFUZZ effective in detecting and exploiting SQLI vulnerabilities?"
- RQ2: "How does BIOFUZZ compare with other similar white-box and black-box techniques?"

[S40] introduces a new method for knowledge-based security testing by logic programming and the related tool implementation for model-based non-functional security testing of web applications and then a comprehensive empirical study including rigorous measurement.

Entitled as "Semi-Automatic Security Testing of Web Applications from a Secure Model", [S66], raises and addresses three research questions:

- RQ1: "Can we successfully exploit a vulnerability at the concrete level?"
- RQ2: "How many times do test experts have to be guided?"
- RQ3: "What is the advantage of modeling the system at the abstract message level instead of the protocol level?"

4.2 GROUP 2: SPECIFIC TO THE WEB SECURITY TESTING TOOL

4.2.1 RQ 2.1: Types of papers by contribution facet

Popularity of tools are shown in Figure 8. We have classified the commonly used and common ones in the studies as a separate title and the others as a separate title. After the data extraction, we examined the popularity of testing tools usage.

48 of the 80 articles mentioned tools which are Burp Suite [S2, S14, S39, S43, S45, S46, S61], OWASP ZAP [S15, S43, S46, S59, S69, S76, S77], Acunetix [S1, S33, S52, S60], Paros [S1, S37, S43], Vega [S15, S52, S76], Volcano [S25, S37, S58], WebScarab [S1, S10, S24], Appscan [S20, S41], ARDILLA [S55, S78], Fortify [S1, S19], JBroFuzz [S1, S33], Nikto [S15, S69], Pixy [S62, S63], PMD [S60, S75], RIPS [S59, S75], SQLMap [S55, S74] with the main headings. Under the other headings, there are 7 nameless tools and 44 different tools that could be listed as ACTS [S14], Apache Cactus [S65], Arachni [S43], AutoInspect [S20], Ben [S2], BIOFUZZ [S55], Commix [S59], FindBugs [S75], Google Crawler [S6], Harvest [S6], HTTPUnit [S9], IMATT [S12], Junit [S65], JWAST [S75], Larbin [S6], NESSUS [S77], Netcraft S69], Netsparker [S45], Nmap [S69], PoC.py [S45], SAFELI [S75], ScanDo [S20], SeaMonster [S60], Selenium framework [S47], SideJacking [S73], simTI-WS [S5], Skipfsh [S15], Sonarcube [S59], Sparrow [S33], Sparta [S69], TAMPER DATA [S24], Teleport [S6], Wapiti [S15], WATIR [S65], WAVES [S6], Web Spider [S40], Webfuzz [S70], Web-Glimpse [S6], WebInspect [S20], WebSphinx [S6], WebTester [S8], Wireshark [S59], XSS Analyzer [S35], XSS Test Driver [S29], XSSER [S59].

As seen in Figure 8, the most common tools on web application security testing are Burp Suite and OWASP ZAP. When it was examined, the specific tools for papers were least common tools because these were used only for one study. Also, Vega and Acunetix are becoming more popular because of technological improvement on these tools.

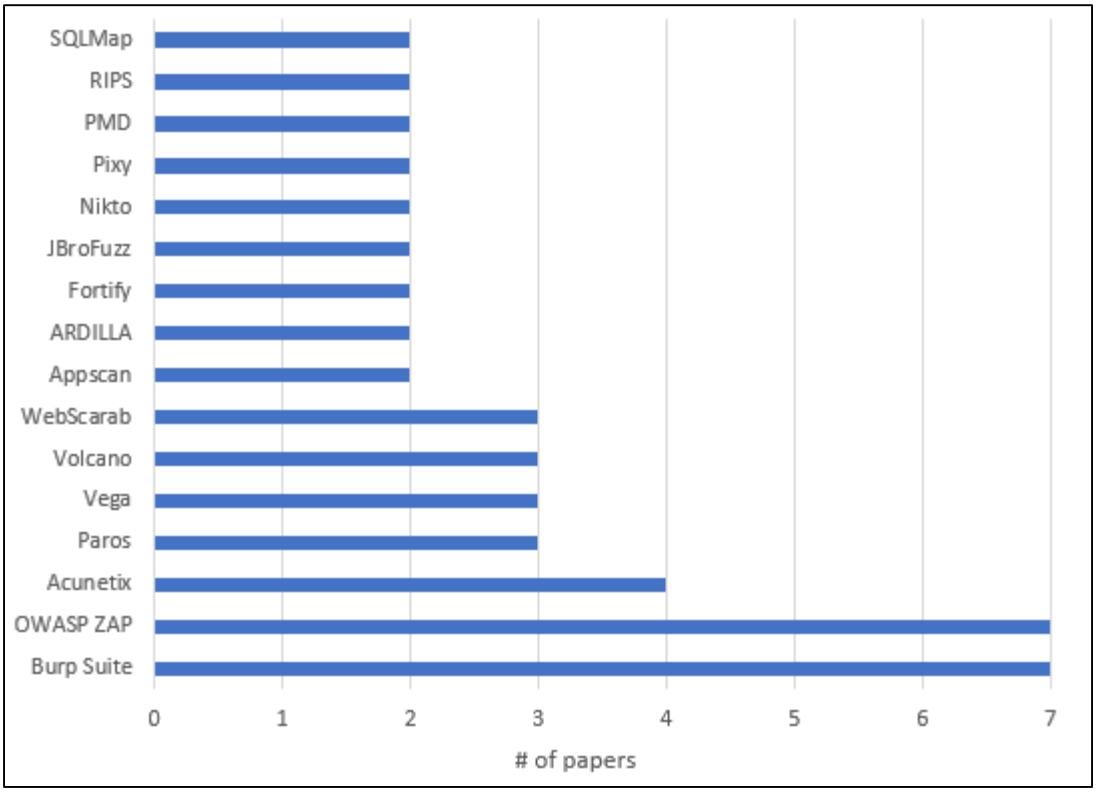


Figure 8: Security testing tools of web applications

4.2.2 RQ 2.2: License type

There are lots of choices for tool licensing, we focused on tool's licensing that academic and industrial researchers used as illustrated in Figure 9. We discussed affiliation under 3 groups: Academy, Industry and Collaboration. Each affiliation group used different types of licensed tools, thus we examined each affiliation group and their licensed tool choices as Commercial, Commercial Free, Academia and Open Source.

While Academy used 8 Commercial, 2 Commercial Free, 11 Academia and 27 Open Source tools, Industry didn't use any Academia and Commercial Free tool, they focused on 3 Commercial and 15 Open Source tools. Since A is the combination of these two, the proportions also look like 3 Commercial, 2 Commercial Free, 3 Academia and 6 Open Source.

Academies generally used their own tools that were improved for their studies. Because of this, a number of academia tools came behind the open source tools. Also, for each group, open source tools were the most used tools because they are free and changeable.

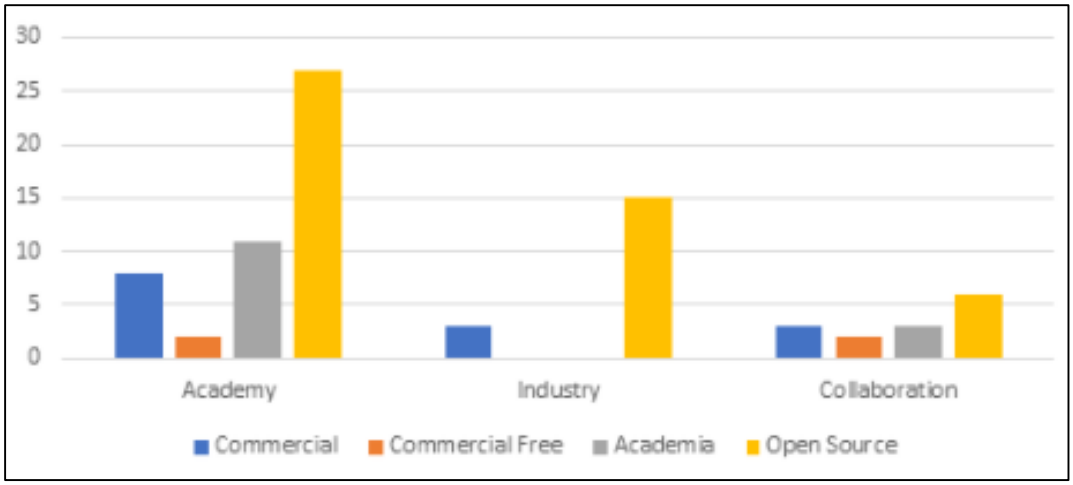


Figure 9: Tool licensing according to affiliation

4.2.3 RQ 2.3: Analysis technique

Is static analysis or dynamic analysis used for security testing? Web application security testing mainly focuses on two types of analysis techniques. Thus, we discussed these types as Static Analysis – SAST (Static Application Security Testing), Dynamic Analysis – DAST (Dynamic Application Security Testing) and others, and we tabulated these data.

Web Application Security Testing groups analysis technique under 3 categories; SAST (Static Application Security Testing), Dynamic Analysis – DAST (Dynamic Application Security Testing) and IAST (Interactive Application Security Testing). The studies that we examined focus on SAST and DAST (as 22 SAST and 49 DAST), there is no study that focuses on IAST. Also, only one study focused on Grey-Box Testing as an analysis technique.

According to our reviews, the analysis technique being SAST or DAST is related to the tools. As we can see in Table 3, we examined tools as static analysis or dynamic analysis tools.

Table 3: Tools associated with analysis technique

SAST	DAST
SonarCube, PMD, RIPS, Volcano, Sparrow, Fortify, IMATT, CPPcheck, Pixy, JWAST, FindBugs	Paros, WebScarab, JBroFuzz, Acunetix, HTTPUnit, TamperData, IMATT, BioFuzz, Ardilla, JWAST, WAVES, Teleport, WebSphinx, Larbin, Web Glimpse, Harvest, AppScan, ScanDo, WebInspect, AutoInspect, WebFuzz, XSSAnalyzer, OWASP ZAP, SQLMap, XSS Test Driver, Vega, Arachni, XSSER, Commix, Netsparker, Nikto, Skipfish, Wapiti, WebSpider, Nessus, Burp Suite, Sparta, Nmap

4.2.4 RQ 2.4: Automation level

Automation Level shows automation type of web application security testing. We grouped automation levels under three groups that were Manual, Semi-Automatic and Automatic and examined each level of testing by years.

In Figure 10, we have also graphed all the articles in the pool by years so that you can see the trend of the articles that did not mention automation level. After the comparison of these trends, we deduced that automation level is increasing year by year. According to this result, we could say technological improvement and tool automation level increasing were affected automation level of security testing.

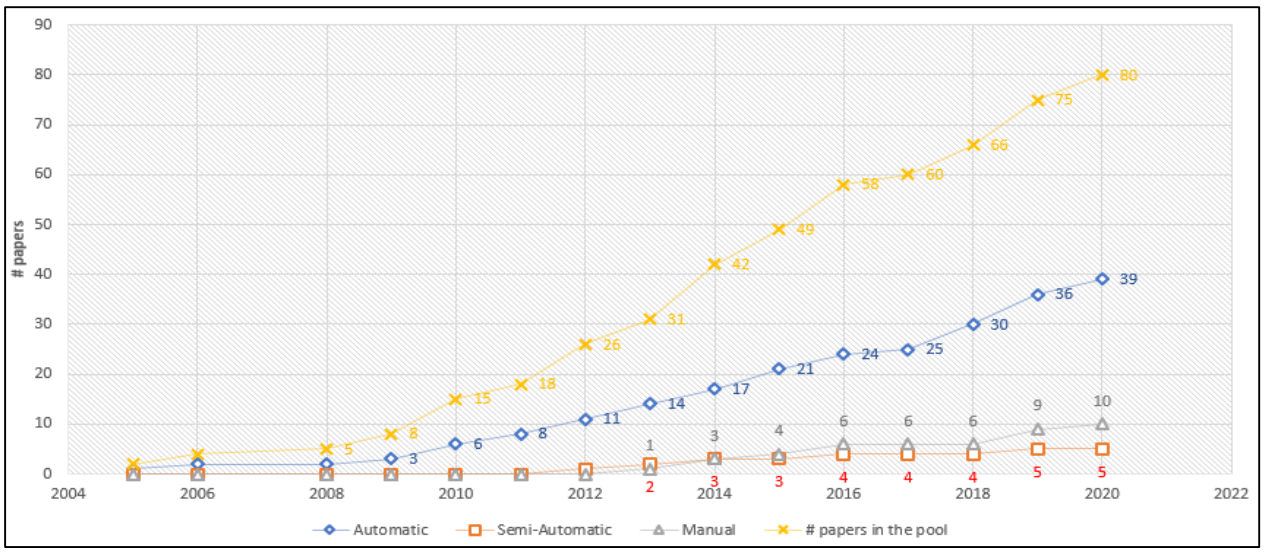


Figure 10: Cumulative trend of automation level

4.2.5 RQ 2.5: Accuracy focusing

Another important part of web application security testing tools is that they are known to be error-prone and report false positives. The problem of false positives and false negatives are common for the automated web application security testing tools. If a vulnerability is reported by a web application security tool but in reality it is not existing, it is called as false positive. If an existing vulnerability is missed by a web application security testing tool, this behavior is called as false negative. 19 of the 80 studies mentioned the false positives and/or false negatives in their approach. In total, only 6 of the 19 papers (we provide their list in Table 4) explained the problem of false positives and/or false negatives with going into detail and they used a method or tool to eliminate them, while large ratio of papers (13 of 19) only explained the problem in their study in a single sentence.

Table 4: Method or Tool which is used to eliminate the problem of false positives and/or false negatives

No	Source	Method-Tool	Truth values
1	[S6]	Negative response extraction (NRE) Mechanism	False Negative
2	[S34]	The Hybrid Vulnerability Analyzer	False Positive and False Negative
3	[S40]	Knowledge-based security testing of web applications founded on logic programming and model-based testing	False Positive and False Negative
4	[S54]	White and black box combination analysis method	False Positive
5	[S60]	The EAST methodology	False Positive
6	[S74]	SQLTest	False Positive and False Negative

4.3 Group 3: Specific to the security testing

4.3.1 RQ 3.1: Vulnerability type

The Open Web Application Security Project is an online community that produces freely-available articles, methodologies, documentation, tools, and technologies in the field of web application security. OWASP has published the most common 10 vulnerabilities every 3 years.

We handled all vulnerabilities in studies and grouped these vulnerabilities according to OWASP Top 10 2017 as OWASP and Others. It is necessary to evaluate the studies by years to show whether the studies address the vulnerabilities according to the OWASP. Because of this, we created the table below.

We created the Vulnerability Type Matrix as Vulnerability/Year and that matrix contained the papers that referred to these vulnerabilities (Table 5). Study numbers and the stars at the end of some studies show normal ones that show vulnerabilities that were in OWASP Top 10 for related years. One star at the end show vulnerabilities that pointed to OWASP Top 10 but not for related years. Two stars at the end show vulnerabilities that weren't pointed to OWASP Top 10 for any year. Years and related OWASP Top 10 are as follows: OWASP Top 10 2004 for 2005-2006, OWASP Top 10 2007 for 2008-2010, OWASP Top 10 2010 for 2011-2013, OWASP Top 10 2013 for 2014-2017, OWASP Top 10 2017 for 2018-2020.

According to a grouping note about normal, one star at the end and two stars at the end, we made some inference: All years except 2018 handled Injection Vulnerabilities. After the Injection Vulnerabilities, Cross-Site Scripting is the most common handle vulnerability. Especially last year's OWASP (2018-2020), there are lots of vulnerabilities that weren't listed in OWASP Top 10 List.

Table 5: Vulnerability types

Vulnerability/Year	2005	2006	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020
Cross-Site Scripting	6	20, 65		32	53, 60	63	7, 13, 33, 36, 62, 66	1, 24, 35, 47, 70	12, 18, 29, 38, 46	4, 8, 1,4, 31,2, 75	19, 39, 76	22	43, 51, 59, 78	15, 27, 40, 45, 77	42, 48, 50, 69
Injection	6, 9	20, 65		3, 32, 37	25, 53, 60	58	7, 13, 33, 62	1, 24, 70	12, 54, 55	4, 8, 16, 75	10, 39, 76	22, 74	43, 49, 51, 59, 78, 80	15, 27, 40, 52, 64, 77	42, 48, 50, 69
Cross-Site Request Forgery					60			1		4	39, 76		59*	5*, 15*, 52*	69*
Broken Access Control				32*			7*, 66*	47*							42
Broken Authentication		65		32	60			1		4					42
Directory Traversal							7**	70**		75**			43**, 59**	15**	
Sensitive Data Exposure				32*					21	4	76				42
Insecure Direct Object Ref.					60			1		4				15*	
Security Misconfiguration				32*						4				15	42
Using Components with Vuln...							13*, 26*							45	42
Buffer Overflow					53*						39**, 76**		59**	15**	
Brute Force													43**	15**	
Clickjacking													49**	77**	69**
HTTP Response Splitting										75**				15**	
Information Disclosure							7**				39**, 76**			77**	
Information Leakage					60										
Remote File Inclusion											39**, 76**			15**	
Session Hijacking			73**											15**	
Unvalidated Redirects														15**	
Arbitrary File											39**				
Cleartext Password														15**, 52**	
Code Execution											39**		43**		
Cookie Validation Error							7**								
Default Credential													49**	52**	
Denial of Service											39**				
Directory Browsing														45**	
Directory Listing														15**	
Error Disclosure														15**	69**
Failure to Restrict URL Access					60										
File Uploading					53**										
Forced Browsing													43**		
Full Path Disclosure														15**	
HTTP Trace Option														15**	
HTTP-Only Flag														15**	69**
Input Validation		65													
Insecure Cryptographic Storage													59*		
Insecure Deserialization															42
Insufficient Logging															42**
Insufficient Transport Lay Pro...													59*		
Integrity Checks														45**	
Invalid Redirects and Forwards										4			59*		
Local File Inclusion											39**				
Log Spoofing													43**		
Malicious File Execution					60										
Memory Corruption											39**				
Memory Leak													59**		

Memory out of bound				59**		
Missing Checks				43**		
Missing Function Level Access		4				
Multiple Logins Overload					15**	
Network Stack Vulnerabilities					15**	
Outdated Components					15**	
Path Disclosure			39**			
Plain Text Cookie					52**	
Restricted URL Access		4**				
Skip Stages				43**		
Standard Password					15**	
Transport Credential				49**	52**	
URL Jump	7**					
Web Parameter Tampering					15**	
X-Frame Options Header					15**, 77**	69**
XML External Entities (XXE)						42**
XSRF			10**			

4.3.2 RQ 3.2: Violation type

The violation in the web application security refers to structures that may cause weakness outside the normal structure of the system. There are lots of security violations in web application security. They can be listed as XML Scheme Violation, Web Server Enforcement Violation, Web Access Policy Violation, URL Length Violation, URL Detection – Content Violation, URL Access Violation, HTTP Protocol Violation, SQL Signature Violation, SSL Protection Violation, SSL Enforcement Violation, Parameter Violation, File Upload Violation, Content Protection Violation, StartURL Violation, Cookie Violation etc.

In our study, there is only one paper which is [S6], "A testing framework for Web application security assessment", handled Cookie Privacy Violation and no other studies have focused on violations.

4.3.3 RQ 3.3: SDLC Focusing

In general, most organizations and enterprises usually perform the security testing at the end of the software development life cycle(SDLC), thus the vulnerabilities might be found in the last phase of the SDLC. That is why security testing activities should shift from the last phase to the early phase of SDLC and it should continue throughout the SDLC. In this RQ, we aimed to gather information about security testing in terms of the SDLC by asking the question: "Does the study consider SDLC and Why?".

7 of the 80 studies specifically mentioned the SDLC. The common point of most studies [S60, S65, S17, S57, S43,S22] is the elicitation and specification of security requirements, and that security testing should be addressed during all phases so that developers and QA engineers will be more aware of security issues. They will be guided how to test, and they will test the code well, and produce more elaborated and precise results. Security Testing activity thus will start as early as possible in the SDLC. The studies [S65, S67, S17, S76] are specially examined agile software development methodologies and it is suggested to be preferred. Especially in the [S17] reference of the study, it is suggested to use a agile testing practice which is Behavior Driven Development (BDD) following the guidelines by the selected security framework. We can see in Table 6, agile testing methodologies have been conducted to produce web applications that are more robust, easier to maintain, and more secure since 2006.

Table 6: Reference of the studies considers SDLC by years

Reference of the studies considers SDLC	Year
[S65]	2006
[S57]	2010
[S60]	2010
[S17]	2015
[S76]	2016

[S22]	2017
[S43]	2018

4.4 GROUP 4: SPECIFIC TO SYSTEM UNDER TESTING (SUT)

4.4.1 RQ 4.1: Web application type

For web application security testing there are different types of web applications, thus we mainly focused four types of web application as custom-made web application, web application security scanner test sites, open-source web application, and real-world web application.

The 49 of 80 studies in the final pool were classified according to the type of web application and the results obtained are shown in the following figure. A web application can have more than one type, as well as the total number of web application types is more than the number of studies. As can be seen in Figure 11, the real-world web application has been applied the most in the studies. The details of web application classification based on type is given under "Web application and their types" sheet in our online spreadsheet.

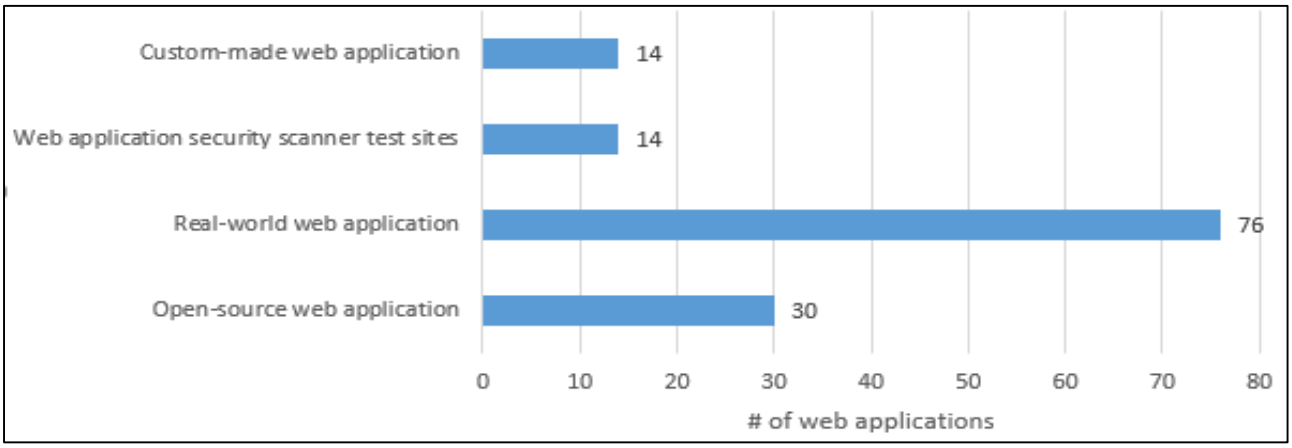


Figure 11: Web Application Types

4.4.2 RQ 4.2: SUT number and name

We examined that 49 of 80 studies used 261 web-based applications in revealing security vulnerabilities in web applications. As shown in Figure 12, the most commonly used web applications are WebGoat, Mutillidae, DVWA, Gruyere, BodgeIt, Github, EasyMoblog, phpaacms, Jobhut, testfire and Yapig. WebGoat, Mutillidae, DVWA, Gruyere, and BodgeIt are the most popular since they are free, open source, and deliberately vulnerable web-application. Github, EasyMoblog, phpaacms, Jobhut and testfire were used in at least 2 studies to reveal vulnerability. The details of web application classification based on studies is given under "Web application and their types'' sheet in our online spreadsheet. The rest of web applications were mentioned in only one study. We have classified web applications as Tunestore (S[1]), Tag2HtmlPageScope(S[2]), Tag2TagStructure(S[2]), Event2TagScope (S[2]), Event2DoubleQuotePropertyScope(S[2]), NAI(S[6]), Lucent(S[6]), Trend Micro(S[6]), Palm(S[6]), Olympic(S[6]), Apache(S[6]), Verisign(S[6]), Ulead(S[6]), Cert(S[6]), Maxtor(S[6]), Mazda(S[6]), Linux Journal(S[6]), Cadillac(S[6]), Web500(S[6]), osCommerce(S[7]), AccountMSystem(S[8]), Agent Based Web Application(S[12]), XVWA(S[15]), Bitweaver(S[14]), WackoPicko(S[16]), iTrust(S[19]),18 apps(S[20]),Mycrocms(S[25]), pastelcms(S[25]), JEUS Web Server(S[26]), Oracle WebLogic Server(S[26]), WebSphere(S[26]), Sun One Web Server(S[26]), Online Shopping Application(S[28]), Chrome 11.0.696.68(S[29]), IE 8.0.6001.19048(S[29]), Opera Mobile 11(S[29]), Opera 11.11 rev2019(S[29]), IE Mobile(S[29]), Safari Mac OSX(S[29]), iPhone 3GS(S[29]), Android 2.2(S[29]), Firefox 5 Android(S[29]), Firefox 8.0a1(S[29]), IE 6.0.2900.2180(S[29]), Firefox 2.0.0.2(S[29]), Netscape 4.8(S[29]), IE 4.0.1(S[29]), Wecoo(S[32]), BugTrack(S[33]), en.faname(S[37]), shnews(S[37]), ajchat(S[37]), Neuronnews(S[37]), PHPEchoCMS(S[37]),

taskfreak(S[37]), GestDown(S[37]), tutorialCMS(S[37]), An enterprise application(S[38]), Hale Sports Club (HSC)(S[39]), Facebook(S[41]), Fandango(S[41]), Netflix(S[41]), OpsLead(S[41]), User Site(S[41]), elFİnder(S[41]), EDLAH2(S[42]), Jenkins(S[42]), OnlineShop(S[43]), SAML-based SSO services for Google Apps(S[44]), Sample Application(S[45]), davidriha.cz(S[45]), SIGARRA(S[49]), jigsawplanet(S[49]), TdinApp(S[49]), Acunetix(S[49]), WebChess(S[55]), Schoolmate(S[55]), FaqForge(S[55]), geccBBlite(S[55]), phpMyAddressbook(S[55]), Elemata(S[55]), A smart home application(S[60]), PHP QR Code(S[60]), Moodle(S[60]), Social Network Visualized (SNV)(S[60]), phpNuke(S[62]), 3 real-world web applications(s[64]), M3TR(S[67]), STORM player V2.9(S[68]), 100 apps(S[69]), A simple ASP.NET login procedure(S[70]), Hotel Management Information System(S[71]), Sample Asp app(S[72]), Gmail(S[74]), AOL Mail(S[74]), GMZ Mail(S[74]), Yahoo Mail(S[74]), Inbox Mail(S[74]), Gawab Mail(S[74]), Yahoo Mail Classic(S[74]), Zenbe Mail(S[74]), Fast Mail(S[74]), Hotmail(S[74]), An application(S[76]), A financial web application(S[77]), A prototype of the banking application(S[78]).

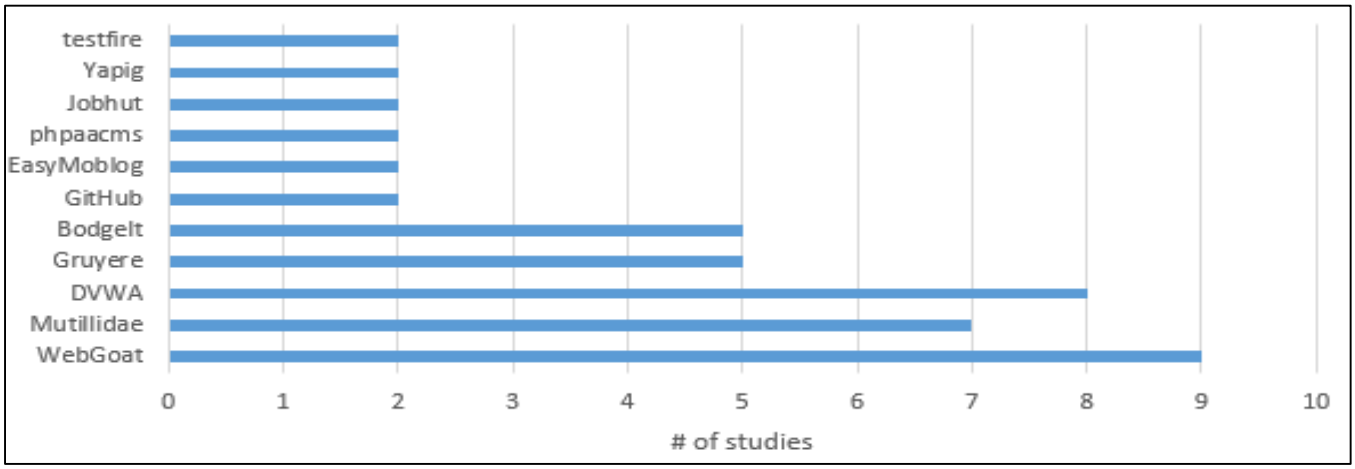


Figure 12: Popular web applications

4.4.3 RQ 4.3: Developing language of SUT

There are lots of languages and frameworks to develop web applications. Each language and framework can contain its own vulnerabilities or can be prone about some type of vulnerabilities. Because of this situation, we focused on technology of tested web applications.

As shown in Figure 13, the most common technology is PHP with 44 tested web applications in 25 papers out of 80. After this, 29 tested web applications in 24 papers with JAVA, 42 tested web applications in 19 papers with JavaScript, 6 papers .NET and 5 papers Python. Otherwise, 20 tested web applications in 4 papers used real-world web applications with complex technologies and 26 papers with no information about used technology.

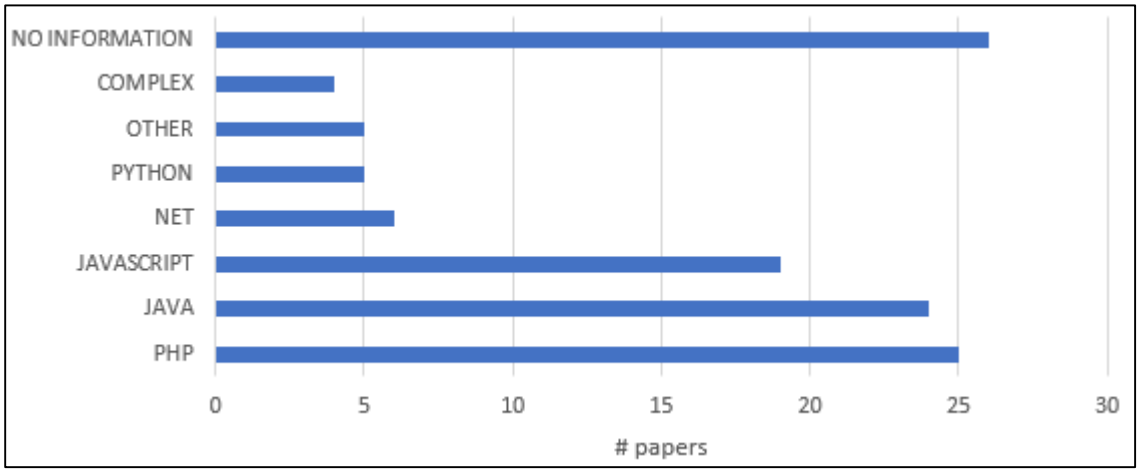


Figure 13: Development language of web applications

4.4.4 RQ 4.4: Connection type

Nowadays web applications use two main protocols as a web communication protocol, HTTP and HTTPS. The main difference between HTTPS and HTTP is that HTTPS is more secure because of usage of encryption and cryptographic algorithms. Basically HTTPS is a more advanced version of HTTP.

Because of this common usage of these protocols, we focused on HTTP and HTTPS emphasis of our SLM studies. We mainly focused on protocol usage of tested web applications and tabulated this information.

As shown in Figure14, according to our results, majority of the papers did not handle protocol usage (64 papers out of 80). While the web applications tested in 16 studies used HTTP, only 2 studies were found to use HTTPS. It should also be noted that the 2 studies that used HTTPS also used HTTP.

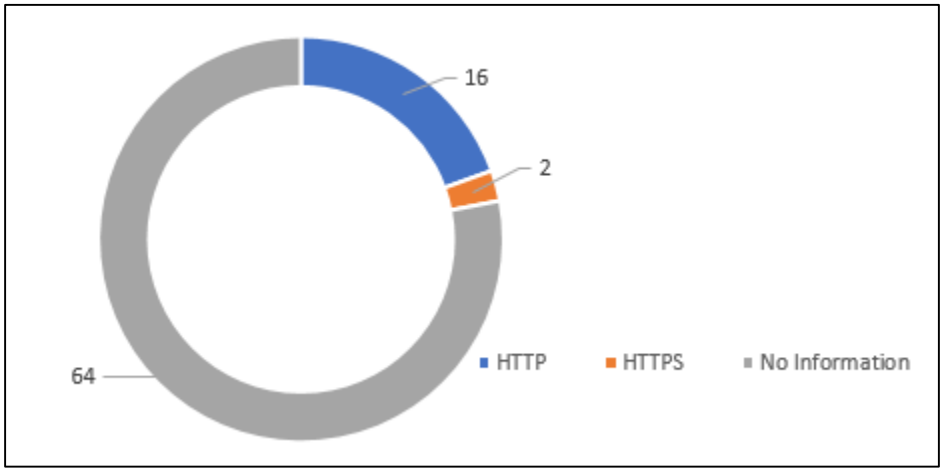


Figure 14: Web communication protocols

5 DISCUSSIONS

5.1 BENEFITS OF THIS STUDY

Our study, web application security testing systematic literature review, set sight on software developers, web application security testers and researchers. Software developers can take advantage of this study while they develop their web applications. Our study focused on security testing tools, web security analysis techniques, the most encountered vulnerabilities, web application types and developed languages etc. Software developers can use that information and can develop more secure web applications against attackers.

Additionally, web application security testers can use all these information and results to test web applications and their vulnerabilities. They can improve their testing process with all this information. Also, researchers can extend our study with their research. They can add/remove and extend all headings and exhibit new research.

5.2 LIMITATIONS AND POTENTIAL THREATS TO VALIDITY

The most important topics related to threats for validation of this review study are inaccuracy of data extraction, missing studies in the final pool due to incomplete search terms, selection of databases and author attention related with exclusion/inclusion criteria. In this part, these threats are discussed based on a standard checklist for validity threats presented in (Claes et al., 2012).

Conclusion validity: This kind of validity of a literature review study is related with the obtaining of proper conclusions via strict and repeatable analysis. Every paper in the pool was evaluated via minimum two authors to decrease error for data extraction. Differences of opinion between the authors were resolved with consensus. Pursuing a defined process provided not only repeatability of this research but also assured that findings of similar research will not have significant divergences from our results.

Internal validity: The systematic approach that has been used for source selection is defined in Section 3. For assuring that this review is repeatable, search terms/engines and inclusion/exclusion criteria are delicately described and recorded. Troublesome topics in the selection process are missing search engines/terms, and carelessly applying exclusion/inclusion criteria.

Missing search terms/engines can cause an incomplete pool of primary studies. Different authors used different search terms to find all related papers and then sources were reviewed by one of the authors. Systematic searching using defined keywords has been done followed by manual search in references of the pool to mitigate risk of incomplete relevant papers. As mentioned before, according to well-known proved SLR guidelines, Google Scholar and Scopus are widely used in review studies (B. Kitchenham, 2007). These two search engines were used in order to minimize the chances of missing papers in the initial pool.

Applying inclusion/exclusion criteria can be different via the perception and background of authors. In order to avoid the mentioned threat and provide the high quality of the final pool and the extracted data, comprehensive peer review was applied.

Construct validity: Construct validity is used to decide how well a test measures what it is supposed to measure. To put it more clearly, is the test built to successfully test what it claims to test? In review studies, such construct validity risks are the suitability and classification scheme of RQs used for data extraction (Garousi et al., 2020). For reducing these risks, the authors discussed in detail the RQs together. Moreover, the proved classification schemes were used or the categorization schemes were improved from these proved schemas.

External validity: External validity of this study can be described as the validity of applying the conclusions of the study outside the context of the study. We extracted data from all industrial and academic papers in the related work, which provided sufficient information for concluding findings useful for both academia and industry. Moreover, notice that our results in this research are only in the area of security testing of web applications. We did not have any tendency for making our findings generalize apart from this topic.

6 Conclusion and future work

In this article, we conducted a systematic literature mapping in order to identify the state of the art in the security testing of web applications and detect what we know about this area. Our aim is to benefit practitioners and researchers by providing the most extensive research in the field of web application security testing.

This paper mainly investigates types of testing topics studied, types of testing tools, vulnerability and violation types, and the types of system under testing, in security testing of web applications on a systematic literature mapping including 80 papers. Web application security testing is a growing field of research for both academia and companies especially working on internet technologies. Most studies for security testing of web applications focus on Cross-Site Scripting and SQL injection vulnerability. We recommend companies, which aim to improve their security level, consider the collected information from this comprehensive review study into account. While we conducted in this review work a basic level of analysis, a more thorough systematic literature review is needed to objectively evaluate and compare the strengths and limitations of cross-site scripting and SQL injection vulnerability that we have worked on in this initial SLM. Such a systematic literature review can be enhanced with additional new empirical studies in which a few Systems Under Test (SUTs) are chosen, to which the security testing frameworks of web applications are applied.

Our direction of future work is to enhance the industry-academia collaborative projects using the results of this review study. Moreover, our future work includes to refine and further evaluate the available evidence on security testing of web applications and empirical evaluation of effectiveness and efficiency of available web application security testing approaches.

7 References

[Dataset]Aydos, M., Aldan, C., Coskun, E., Soydan, A., 2021. Dataset for the study: Security testing of web applications: A systematic mapping of the literature [WWW Document]. URL bit.ly/3vnvS5M

B. Kitchenham, S.C., 2007. Guidelines for performing Systematic Literature Reviews in Software Engineering.

Bhargav, Abhay, and B.V.K., 2010. Secure Java: for web application development. CRC Press.

Claes, W., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A., 2012. Experimentation in software engineering. Springer Science & Business Media.

Garousi, V., Bauer, S., Felderer, M., 2020. NLP-assisted software testing: A systematic mapping of the literature. Inf. Softw. Technol. 106321.

Garousi, V., Felderer, M., Karapıçak, Ç.M., Yılmaz, U., 2018. Testing embedded software: A survey of the literature. Inf. Softw. Technol. 104, 14–45.

Garousi, V., Felderer, M., Kılıçaslan, F.N., 2019a. A survey on software testability. Inf. Softw. Technol. 108, 35–64. https://doi.org/10.1016/j.infsof.2018.12.003

Garousi, V., Giray, G., Tüzün, E., Catal, C., Felderer, M., 2019b. Aligning software engineering education with industrial needs: A meta-analysis. J. Syst. Softw. 156, 65–83. https://doi.org/10.1016/j.jss.2019.06.044

Haddaway, N.R., Collins, A.M., Coughlin, D., Kirk, S., 2015. The role of Google Scholar in evidence reviews and its applicability to grey literature searching. PLoS One 10.

Hadeel, A., Roper, M., 2020. A systematic literature review of machine learning techniques for software maintainability prediction. Inf. Softw. Technol. 119.

Kai, P., Feldt, R., Mujtaba, S., Mattsson, M., 2008. Systematic mapping studies in software engineering., in: 12th International Conference on Evaluation and Assessment in Software Engineering (EASE). pp. 1–10.

Kuutila, M., Mäntylä, M., Farooq, U., Claes, M., 2020. Time pressure in software engineering: A systematic review. Inf. Softw. Technol. 121. https://doi.org/10.1016/j.infsof.2020.106257

Kyong-Ho, C., Lee, D., 2015. Enhanced Security Measurement of Web Application Testing by Outsourcing. Converg. Secur. J. 15, 3–9.

Li, N., Shepperd, M., Guo, Y., 2020. A systematic review of unsupervised learning techniques for software defect prediction. Inf. Softw. Technol. 122. https://doi.org/10.1016/j.infsof.2020.106287

Mohanty, S., Acharya, A.A., Mishra, D.B., Panda, N., 2019. Security Testing of Web Applications Using Threat Modeling : A Systematic Review. Int. J. Comput. Sci. Mob. Comput. 8, 50–57.

OWASP. 4.0 Testing Guide [WWW Document], 2014. . OWASP Found. URL https://www.owasp.org/images/1/19/OTGv4.pdf

Pariwish, T., Alam, K.A., Jamil, A., Tauseef, H., Ajmal, S., Asif, R., Rehman, B., Mustafa, S., 2019. Analysis of automated web application security vulnerabilities testing, in: The 3rd International Conference on Future Networks and Distributed Systems. pp. 1–8.

Petersen, K., Vakkalanka, S., Kuzniarz, L., 2015. Guidelines for conducting systematic mapping studies in software engineering: An update. Inf. Softw. Technol. 64, 1–18.

Seng, L.K., Ithnin, N., Mohd Said, S.Z., 2018. The approaches to quantify web application security scanners quality: A review. Int. J. Adv. Comput. Res. 8, 285–312. https://doi.org/10.19101/IJACR.2018.838012

Shirey, R., 2007. Internet Security Glossary, Version 2 [WWW Document]. URL https://tools.ietf.org/html/rfc4949

Web Application Security Testing [WWW Document], 2021. URL https://www.youtube.com/watch?v=5ADO1E-NAG4

Wohlin, C., 2014. Guidelines for snowballing in systematic literature studies and a replication in software engineering. ACM Int. Conf. Proceeding Ser. 1–10. https://doi.org/10.1145/2601248.2601268

APPENDIX: LIST OF THE PRIMARY STUDIES

[S1] Dukes, L., Yuan, X., & Akowuah, F. (2013, April). A case study on web application security testing with tools and manual testing. In 2013 Proceedings of IEEE Southeastcon (pp. 1-6). IEEE.

[S2] Simos, D. E., Kleine, K., Ghandehari, L. S. G., Garn, B., & Lei, Y. (2016, October). A combinatorial approach to analyzing cross-site scripting (XSS) vulnerabilities in web application security testing. In IFIP International Conference on Testing Software and Systems (pp. 70-85). Springer, Cham.

[S3] Haixia, Y., & Zhihong, N. (2009, July). A database security testing scheme of web application. In 2009 4th International Conference on Computer Science & Education (pp. 953-955). IEEE.

[S4] Abouelmehdi, K., Bentajer, A., Dali, L.,& Sefiani, N., (2015, April). A New Approach Of Web Attacks Classification For Testing Security Tools At The Application Level, in Journal of theoretical and applied information technology(JATIT), pp. 347-354.

[S5] Su, J. M., Cheng, M. H., Wang, X. J., & Tseng, S. S. (2019, August). A Scheme to Create Simulated Test Items for Facilitating the Assessment in Web Security Subject. In 2019 Twelfth International Conference on Ubi-Media Computing (Ubi-Media) (pp. 306-309). IEEE.

[S6] Huang, Y. W., Tsai, C. H., Lin, T. P., Huang, S. K., Lee, D. T., & Kuo, S. Y. (2005). A testing framework for Web application security assessment. Computer Networks, 48(5), 739-761.

[S7] Yan, B., Li, X., & Du, Z. (2012, August). A Threat Model-Driven Security Testing Approach for Web Application. In International Conference on E-business Technology and Strategy (pp. 158-168). Springer, Berlin, Heidelberg.

[S8] Yu, X., & Jiang, G. (2015, August). A Web Security Testing Method Based on Web Application Structure. In International Conference on Cloud Computing and Security (pp. 244-258). Springer, Cham.

[S9] Tappenden, A., Beatty, P., Miller, J., Geras, A., & Smith, M. (2005, July). Agile security testing of web-based systems via httpunit. In Agile Development Conference (ADC'05) (pp. 29-38). IEEE.

[S10] Calvi, A., & Viganò, L. (2016, April). An automated approach for testing the security of web applications against chained attacks. In Proceedings of the 31st Annual ACM Symposium on Applied Computing (pp. 2095-2102).

[S11] Vikas, S., & Gupta, S. An Empirical Evaluation of Mutation Testing for Improving The Test Quality of Web Application's Security.

[S12] Eassa, F. E., Zaki, M., Eassa, A. M., & Aljehani, T. (2014). An Integrated Multi-Agent Testing Tool for Security Checking of Agent-Based Web Applications. WSEAS Transactions on computers, 13.

[S13] Xu, W., Deng, L., & Zheng, Q. (2012, April). Annotating Resources in Sequence Diagrams for Testing Web Security. In 2012 Ninth International Conference on Information Technology-New Generations (pp. 873-874). IEEE.

[S14] Bozic, J., Garn, B., Kapsalis, I., Simos, D., Winkler, S., & Wotawa, F. (2015, August). Attack pattern-based combinatorial testing with constraints for web security testing. In 2015 IEEE International Conference on Software Quality, Reliability and Security (pp. 207-212). IEEE.

[S15] Pfrang, S., Borcherding, A., Meier, D., & Beyerer, J. (2019). Automated security testing for web applications on industrial automation and control systems. at-Automatisierungstechnik, 67(5), 383-401.

[S16] Awang, N. F., & Abd Manaf, A. (2015, September). Automated Security Testing Framework for Detecting SQL Injection Vulnerability in Web Application. In International Conference on Global Security, Safety, and Sustainability (pp. 160-171). Springer, Cham.

[S17] Rathod, P., Julkunen, V., Kaisti, T., & Nissilä, J. (2015, September). Automatic acceptance testing of the web application security with ITU-T X. 805 framework. In 2015 Second International Conference on Computer Science, Computer Engineering, and Social Media (CSCESM) (pp. 103-108). IEEE.

[S18] Hossen, K., Groz, R., Oriat, C., & Richier, J. L. (2014, March). Automatic model inference of web applications for security testing. In 2014 IEEE Seventh International Conference on Software Testing, Verification and Validation Workshops (pp. 22-23). IEEE.

[S19] Mohammadi, M., Chu, B., Lipford, H. R., & Murphy-Hill, E. (2016, May). Automatic web security unit testing: XSS vulnerability detection. In 2016 IEEE/ACM 11th International Workshop in Automation of Software Test (AST) (pp. 78-84). IEEE.

[S20] Choi, K. C., & Lee, G. H. (2006, May). Automatic test approach of web application for security (autoinspect). In International Conference on Computational Science and Its Applications (pp. 659-668). Springer, Berlin, Heidelberg.

[S21] Kavindya, M. D. A., Jayasundera, O. M. B., Guruge, T. L., Senevirathne, D. W., & Manawadu, C. D. (2014). Benefits of Web Applications Security Testing for on Sri Lankan SMEs. COMPUSOFT: An International Journal of Advanced Computer Technology, 3(10).

[S22] Patil, D., & Phil, M. (2017, February). Challenges & Problems in Security Testing of Web based Applications: A study of software companies in Pune city. In National Conference on Innovations in IT and Management, Chennai India. Retrieved (Vol. 16).

[S23] Lai, S. T. Combining reusable test cases and continuous security testing for reducing web apps security risks.

[S24] Qian, L., Wan, J., Chen, L., & Chen, X. (2013, December). Complete Web Security Testing Methods and Recommendations. In 2013 International Conference on Computer Sciences and Applications (pp. 86-89). IEEE.

[S25] Dao, T. B., & Shibayama, E. (2010, December). Coverage criteria for automatic security testing of web applications. In International Conference on Information Systems Security (pp. 111-124). Springer, Berlin, Heidelberg.

[S26] Lee, T., Won, G., Cho, S., Park, N., & Won, D. (2012, September). Detection and mitigation of web application vulnerabilities based on security testing. In IFIP International Conference on Network and Parallel Computing (pp. 138-144). Springer, Berlin, Heidelberg.

[S27] Mohanty, S., & Acharya, A. A. (2021). Detection of XSS Vulnerabilities of Web Application Using Security Testing Approaches. In Intelligent and Cloud Computing (pp. 267-275). Springer, Singapore.

[S28] Garima, S., & Manju, K. (2016). Dual Security Testing Model for Web Applications. International journal of advanced computer science and applications, 7(2).

[S29] Abgrall, E., Le Traon, Y., Gombault, S., & Monperrus, M. (2014, March). Empirical investigation of the Web browser attack surface under cross-site scripting: An urgent need for systematic security regression testing. In 2014 IEEE Seventh International Conference on Software Testing, Verification and Validation Workshops (pp. 34-41). Ieee.

[S30] Sun, Y., Liang, D. Y., & Wang, W. J. (2014). Enhancement of test platform for web application security. In Applied Mechanics and Materials (Vol. 511, pp. 1205-1210). Trans Tech Publications Ltd.

[S31] Bozic, J., Garn, B., Simos, D. E., & Wotawa, F. (2015, April). Evaluation of the IPO-family algorithms for test case generation in web security testing. In 2015 IEEE Eighth International Conference on Software Testing, Verification and Validation Workshops (ICSTW) (pp. 1-10). IEEE.

[S32] Mao, C. (2009, November). Experiences in security testing for web-based applications. In Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human (pp. 326-330).

[S33] Lee, T., Won, G., Cho, S., Park, N., & Won, D. (2012). Experimentation and Validation of Web Application's Vulnerability Using Security Testing Method. In Computer Science and its Applications (pp. 723-731). Springer, Dordrecht.

[S34] Huang, Y. Y., Chen, K., & Chiang, S. L. (2012). Finding security vulnerabilities in Java Web applications with test generation and dynamic taint analysis. In Proceedings of the 2011 2nd International Congress on Computer Applications and Computational Science (pp. 133-138). Springer, Berlin, Heidelberg.

[S35] Tripp, O., Weisman, O., & Guy, L. (2013, July). Finding your way in the testing jungle: a learning approach to web security testing. In Proceedings of the 2013 International Symposium on Software Testing and Analysis (pp. 347-357).

[S36] Avancini, A., & Ceccato, M. (2012, June). Grammar based oracle for security testing of web applications. In 2012 7th International Workshop on Automation of Software Test (AST) (pp. 15-21). IEEE.

[S37] Dao, T. B., & Shibayama, E. (2009, February). Idea: Automatic Security Testing for Web Applications. In International Symposium on Engineering Secure Software and Systems (pp. 180-184). Springer, Berlin, Heidelberg.

[S38] SriNithi, D., Elavarasi, G., Raj, T. M., & Sivaprakasam, P. (2014). Improving Web Application Security Using

Penetration Testing. Research Journal of Applied Sciences, Engineering and Technology, 8(5), 658-663.

[S39] Gohel, H., & Sharma, P. (2016). Intelligent Web Security Testing with Threat Assessment and Client Server Penetration. In Proceedings of International Conference on ICT for Sustainable Development (pp. 555-568). Springer, Singapore.

[S40] Zech, P., Felderer, M., & Breu, R. (2019). Knowledge-based security testing of web applications by logic programming. International Journal on Software Tools for Technology Transfer, 21(2), 221-246.

[S41] Ben-Bassat, I., & Rokah, E. (2020). Locality-sensitive hashing for efficient web application security testing. arXiv preprint arXiv:2001.01128.

[S42] Mai, P. X., Pastore, F., Goknil, A., & Briand, L. (2020, October). Metamorphic security testing for web systems. In 2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST) (pp. 186-197). IEEE.

[S43] Peroli, M., De Meo, F., Viganò, L., & Guardini, D. (2018). MobSTer: A model- based security testing framework for web applications. Software Testing, Verification and Reliability, 28(8), e1685.

[S44] Armando, A., Carbone, R., Compagna, L., Li, K., & Pellegrino, G. (2010, April). Model-checking driven security testing of web-based applications. In 2010 Third International Conference on Software Testing, Verification, and Validation Workshops (pp. 361-370). IEEE.

[S45] Dušek, D. Non-destructive Security Testing for Web Apps.

[S46] Garn, B., Kapsalis, I., Simos, D. E., & Winkler, S. (2014, July). On the applicability of combinatorial testing to web application security testing: a case study. In Proceedings of the 2014 Workshop on Joining AcadeMiA and Industry Contributions to Test Automation and Model-Based Testing (pp. 16-21).

[S47] Shanthakumar, M., Sukumaran, S., & Janarthanam, S. (2013). Online Automatic Security Testing on Web Applications with User Sessions.

[S48] Bozic, J., Li, Y., & Wotawa, F. (2020, August). Ontology-driven Security Testing of Web Applications. In 2020 IEEE International Conference On Artificial Intelligence Testing (AITest) (pp. 115-122). IEEE.

[S49] Araújo, P. J., & Paiva, A. C. (2018). Pattern based Web Security Testing. In 6th International Conference on Model-Driven Engineering and Software Development (MODELSWARD).

[S50] Bozic, J., & Wotawa, F. (2020). Planning-based security testing of web applications with attack grammars. Software Quality Journal, 1-28.

[S51] Bozic, J., & Wotawa, F. (2018, May). Planning-based security testing of web applications. In 2018 IEEE/ACM 13th International Workshop on Automation of Software Test (AST) (pp. 20-26). IEEE.

[S52] Kachhwaha, R., & Purohit, R. (2019). Relating vulnerability and security service points for web application through penetration testing. In Progress in Advanced Computing and Intelligent Engineering (pp. 41-51). Springer, Singapore.

[S53] Wang, R., Xu, Y., & Xiang, Y. (2010, December). Research and Realization of WEB Security Auto-Testing Tool Based on AHP. In 2010 International Conference on Computational Intelligence and Software Engineering (pp. 1-4). IEEE.

[S54] Fan, J., Gao, P., Shi, C. C., & Li, N. G. (2014). Research on combine White-box testing and Black-box testing of Web Applications security. In Advanced Materials Research (Vol. 989, pp. 4542-4546). Trans Tech Publications Ltd.

[S55] Thomé, J., Gorla, A., & Zeller, A. (2014, June). Search-based security testing of web applications. In Proceedings of the 7th International Workshop on Search-Based Software Testing (pp. 5-14).

[S56] Singh, K., & Singh, G. (2014). Security Analysis of Web Application using Genetic Algorithms in Test Augmentation Technique. Advances in Computer Science and Information Technology (ACSIT), 61.

[S57] Assad, R. E., Katter, T., Ferraz, F. S., Ferreira, L. P., & Meira, S. R. L. (2010, August). Security quality assurance on web-based application through security requirements tests: Elaboration, execution and automation. In 2010 Fifth International Conference on Software Engineering Advances (pp. 272-277). IEEE.

[S58] Dao, T. B., & Shibayama, E. (2011, February). Security sensitive data flow coverage criterion for automatic security testing of web applications. In International Symposium on Engineering Secure Software and Systems (pp. 101-113). Springer, Berlin, Heidelberg.

[S59] Alrawais, L. M., Alenezi, M., & Akour, M. (2018). Security Testing Framework for Web Applications. International Journal of Software Innovation (IJSI), 6(3), 93-117.

[S60] Erdogan, G., Meland, P. H., & Mathieson, D. (2010, June). Security testing in agile web application development-a case study using the EAST methodology. In International Conference on Agile Software Development (pp. 14-27). Springer, Berlin, Heidelberg.

[S61] Akmaluddin, M., & Pramadiv, Y. R. (2019, April). Security Testing of a User-Participating Authentication Scheme Implementation on Web-Based Login System. In IOP Conference Series: Materials Science and Engineering (Vol. 508, No. 1, p. 012136). IOP Publishing.

[S62] Avancini, A. (2012, June). Security testing of web applications: A research plan. In 2012 34th International Conference on Software Engineering (ICSE) (pp. 1491-1494). IEEE.

[S63] Avancini, A., & Ceccato, M. (2011, September). 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.

[S64] Liu, M., Li, K., & Chen, T. (2019, July). Security testing of web applications: a search-based approach for detecting SQL injection vulnerabilities. In Proceedings of the Genetic and Evolutionary Computation Conference Companion (pp. 417-418).

[S65] De Vries, S. (2006). Security Testing Web Applications throughout Automated Software Tests. In OWASP Europe Conference (Vol. 1, No. 1, pp. 1-13).

[S66] Büchler, M., Oudinet, J., & Pretschner, A. (2012, June). Semi-automatic security testing of web applications from a secure model. In 2012 IEEE Sixth International Conference on Software Security and Reliability (pp. 253-262). IEEE.

[S67] Hui, Z., & Huang, S. (2010, August). Software security testing of web applications based on SSD. In International Conference on Intelligent Computing (pp. 527-534). Springer, Berlin, Heidelberg.

[S68] Yao, G., Guan, Q., & Ni, K. (2012). Test Model for Security Vulnerability in Web Controls based on Fuzzing. JSW, 7(4), 773-778.

[S69] Devi, R. S., & Kumar, M. M. (2020, June). Testing for Security Weakness of Web Applications using Ethical Hacking. In 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184) (pp. 354-361). IEEE.

[S70] Li, L., Dong, Q., Liu, D., & Zhu, L. (2013, November). The application of fuzzing in web software security vulnerabilities test. In 2013 International Conference on Information Technology and Applications (pp. 130-133). IEEE.

[S71] Zhai, H., Shi, H., & Zhai, R. (2014). The Application of Software Testing Technology on Security in Web Application System. In Applied Mechanics and Materials (Vol. 556, pp. 6159-6161). Trans Tech Publications Ltd.

[S72] Wang, S. Y., Sun, J. Z., & Zhang, J. (2016, August). The Method of Generating Web Link Security Testing Scenario Based on UML Diagram. In 2016 IEEE Trustcom/BigDataSE/ISPA (pp. 1831-1838). IEEE.The Method of Generating Web Link Security Testing Scenario Based on UML Diagram

[S73] Noiumkar, P., & Chomsiri, T. (2008, November). Top 10 free web-mail security test using session Hijacking. In 2008 Third International Conference on Convergence and Hybrid Information Technology (Vol. 2, pp. 486-490). IEEE.

[S74] Stephanow, P., & Khajehmoogahi, K. (2017, March). Towards continuous security certification of software-as-a-service applications using web application testing techniques. In 2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA) (pp. 931-938). IEEE.

[S75] Imran, M., Eassa, F., & Jambi, K. Using Agent Technology for Security Testing of WEB Based Applications. SEDE–2015, 3-10.

[S76] Živković, K., Milenković, I., & Simić, D. (2016). Using open source software for web application security testing. JITA-JOURNAL OF INFORMATION TECHNOLOGY AND APLICATIONS, 12(2).

[S77] Goutam, A., & Tiwari, V. (2019, November). Vulnerability Assessment and Penetration Testing to Enhance the Security of Web Application. In 2019 4th International Conference on Information Systems and Computer Networks (ISCON) (pp. 601-605). IEEE.

[S78] Murthy, P. V. R., & Shilpa, R. G. (2018, September). Vulnerability coverage criteria for security testing of web applications. In 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI) (pp. 489-494). IEEE.

[S79] Scarpino, J. J. (2010). Web Application Security Testing: an Industry Perspective on How Its Education Is Perceived. vol. XI, (1), 142-153.

[S80] Iskandar, A., Tuasamu, M. R. F., Syamsu, S., Mansyur, M., Listyorini, T., Sallu, S., ... & Rahim, R. (2018, September). Web based testing application security system using semantic comparison method. In IOP Conference Series: Materials Science and Engineering (Vol. 420, No. 1, p. 012122). IOP Publishing.

BIOGRAPHICAL NOTES

Assoc. Prof. Dr. Murat AYDOS received the B.Sc. degree from Yildiz Technical University (Turkey) in 1991, and M.S. degree from Electrical and Computer Engineering Department, Oklahoma State University (USA), in 1996. He completed his Ph.D. study in Oregon State University, Electrical Engineering and Computer Science Department in June 2001. Dr. Aydos joined Informatics Institute @ Hacettepe University in April 2013. He is the Head of Information Security Division at the Informatics Institute. Dr. Aydos is the author/co-author of more than 30 technical publications focusing on the applications of Cryptographic Primitives, Information; Data Security Mechanisms.

MSc. Çigdem ALDAN received the BSc degree from the Computer Education and Instructional Technology Department, Middle East Technical University (Turkey) in 2013. She, then completed her MSc degree in Hacettepe University, Software Engineering Department in 2021. She has started working as a test engineer in Ankara. Up to now she has experienced all level of software testing in various domain (defense, telecommunication, finance, stock market and software cybersecurity systems). Currently she works as Quality Assurance Engineer in Invicti Security (Netsparker and Acunetix).

MSc. Evren COSKUN received the B.Sc. degree from Cankaya University Computer Engineering Department (Turkey) in 2004 and M.Sc. degree from Software Engineering Department, Chalmers University of Technology (Sweden), in 2012. He has over 15 years of industry experience in various roles in different companies and countries. Currently, he works as Chief of IT Governance at Turkish Aerospace. At the same time, he is Computer Engineering Ph.D. Student at Hacettepe University.

Alperen SOYDAN received the B.Sc. degree from Eskisehir Technical University Computer Engineering Department (Turkey) in 2019. He is currently Cyber Security MSc. Student at TOBB University of Economics and Technology. At the same time, he is currently employee of Turkish Aerospace as Cyber Security Engineer.