Review Article

International Journal of Advanced Computer Research, Vol 8(38)
ISSN (Print): 2249-7277 ISSN (Online): 2277-7970
http://dx.doi.org/10.19101/IJACR.2018.838012

The approaches to quantify web application security scanners quality: a review

Lim Kah Seng[1*], Norafida Ithnin[2] and Syed Zainudeen Mohd Said[3]
PhD Research Scholar, Department of Computer Science, University of Technology, Johor Bahru, Malaysia[1]
Associate Professor, Department of Computer Science, University of Technology, Johor Bahru, Malaysia[2]
Assistant Professor, Department of Computer Science, University of Technology, Johor Bahru, Malaysia[3]

Received: 15-July-2018; Revised: 17-September-2018; Accepted: 20-September-2018

Abstract
The web application security scanner is a computer program that assessed web application security with penetration testing technique. The benefit of automated web application penetration testing is huge, which web application security scanner not only reduced the time, cost, and resource required for web application penetration testing but also eliminate test engineer reliance on human knowledge. Nevertheless, web application security scanners are possessing weaknesses of low test coverage, and the scanners are generating inaccurate test results. Consequently, experimentations are frequently held to quantitatively quantify web application security scanner's quality to investigate the web application security scanner's strengths and limitations. However, there is a discovery that neither a standard methodology nor criterion is available for quantifying the web application security scanner's quality. Hence, in this paper systematic review is conducted and analysed the methodology and criterion used for quantifying web application security scanners' quality. In this survey, the experiment methodologies and criterions that had been used to quantify web application security scanner's quality is classified and review using the preferred reporting items for systematic reviews and meta-analyses (PRISMA) protocol. The objectives are to provide practitioners with the understanding of methodologies and criterions that available for measuring web application security scanners' test coverage, attack coverage, and vulnerability detection rate, while provides the critical hint for development of the next testing framework, model, methodology, or criterions, to measure web application security scanner quality.

Keywords
Web application security scanner, Penetration testing, Quality criteria, PRISMA.

1.Introduction
Automated web application penetration testing is becoming ubiquitous with the development of computer programs that capable of simulating tester activities of web application penetration testing. Computer programs like HTTrack [1] or Maltego [2] were invented to aid penetration tester in intelligent information gathering. The invented web application security scanners like Acunetix [3] scanned web applications for vulnerability assessment. In the meanwhile, exploitation tools like Metasploit and WFuzz are created to compromise web application confidentiality, integrity, and availability.

The web application penetration testing methodology of [4] showed web application security scanner always has a critical role in scanning the web application for vulnerability detection.

*Author for correspondence

Web application security scanner simulates the actions of penetration tester of inspecting the target web application security. Subsequently, penetrating the security of web application attack vectors with selected attack strings. The web application is vulnerable if it responds positively towards the injected attack strings, or otherwise. The texts of [5] and [6] showed automated web application penetration testing is beneficial to pen-testers, which the scanner not only reduced resources, times, labour work, and costs required for conducting a web application penetration testing, the scanner also eliminates pen-tester reliance on human knowledge. Moreover, the web application security scanner preserved the human knowledge of web application penetration testing by converting them into an executable computer program.

The invention of web application security scanners has made automated web application penetration testing a popular research trend. In this research field, practitioners have translated the web application penetration testing's testing techniques into executable programs, to enhance weak algorithms to detect new web application vulnerability, or to address the challenge of scanning modern web application that continuously expanding in both size and complexity.

A computer is merely a dummy machine that performs the calculation based on the written algorithm. Therefore, writing a sophisticated algorithm to achieve the objective of automated web application penetration testing is important in this research field. Unfortunately, humans tend to make mistakes. Moreover, the process of translating web application penetration testing's testing techniques into the executable computer program is tedious and error-prone. Hence, the designed algorithms are not always perfect, and the issue of false positives and false negatives are common for automated web application penetration testing. The false positives are consumed pen-tester extra effort and times to eliminate the fake vulnerability, while the false negatives are impaired pen-tester judgement in deciding an under-test web application security. Consequently, documents such as [6−10] are labelling web application security scanners as inaccurate and untrustworthy. This elaborate the phenomena of why experimentations are often held to quantify the web application security scanner’s quality.

An intriguing discovery is that the methodology and criteria used for measuring web application security scanner's quality are varying in existing publishing manuscripts. Moreover, there is neither a standard nor a technical document by authorized parties that defined the approach for quantifying web application security scanner's quality. Although web application security consortium (WASC) did publish web application security scanner evaluation criteria (WASSEC) [11] in the year of 2009. However, WASSEC has been just a checklist that described the features of the web application security scanner. Moreover, the corresponding checklist has never received any update for the year it was published. The NIST special publication 500-269 [12] published by the NIST SAMATE project is another out-dated technical document that contained the similar content. Therefore, in this paper, preferred reporting items for systematic reviews and meta-analyses (PRISMA) protocol is used to classify and review the experiment methodology of published conference proceedings and journal papers that had the quality of web application security scanner quantified, to convey the compelling approach of measuring web application security scanner's quality.

The remaining part of the survey is consists of following sections. Section 2 defines the web application security scanners. Section 3 elaborates the concept of quantifying the web application security scanner’s quality. Section 4 presents the literature review's methodology. Subsequently, section 5 reviews the published methodologies. Section 6 classifies the manuscripts based on the selected indices. Finally, section 7 concluded the survey paper with the conclusion remarked.

2. Understanding of web application security scanner

Web application security scanner is a computer program that assesses web application security via simulating the pen-tester action of penetrating the web application security selected attack strings [13−16]. *Figure 1* showed the general architecture of the web application security scanner.

The white box, black box, and hybrid web application security scanner are created to automatically assess web application security in either black box or white box testing environment. The black box testing environment is a testing environment that web application codes are not reachable, while white box testing environment has the total opposite meaning. Therefore, white box web application security scanners perform the vulnerability assessment by inspecting the propagation of malicious data on web application codes via a control flow graph (CFG) or data flow graph (DFG) [17−21]. On the other hand, black box web application security scanner assesses web application security by inspecting the web application execution behaviours for anomalies detection [22−26].

Hybrid web application security scanners are unique in such a way that both software static and dynamic testing techniques are used to assess the web application security scanner. The hybrid web application security scanner parsed the code and also examines the web application execution behaviours. According to [27] and [28], the software static and dynamic testing techniques are integrated to improve the test coverage and to reduce the possibility of producing the false positive and false negatives.

Web application security scanners scan a web application security with passive and active scanning. In the passive scanning, the scanner collects information of under-test web application with reconnaissance algorithm. Then, in active scanning, exploitation is performed to compromise web application confidentiality, integrity, or availability using the security penetration algorithm. This include performs the vulnerability detection with information flow analysis or anomaly detection. Therefore, a web application security scanner generally contains a reconnaissance component, security penetration component, and vulnerability detection component [20, 29–31].

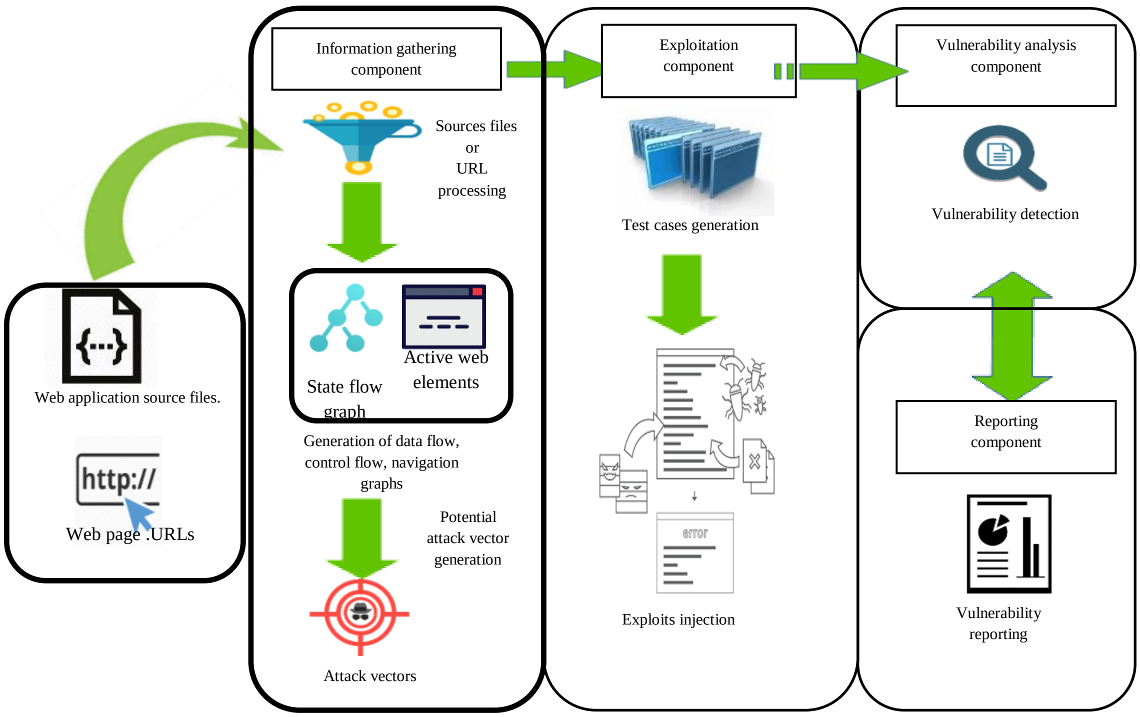

Figure 1 The general architecture of web application security scanner

3.The web application security scanner's quality quantification

The web application security scanner's quality is often quantified to investigate strengths and limitations of existing algorithms, or to evaluate web application security scanner or recently designed algorithm. According to the literature of [9, 32−34], quantification of web application security scanner's quality is achievable by challenging web application security scanner's features with test-beds. The practised experiment methodology usually consists of preparation, execution, and reporting phases. The preparation phase defined the experiment's objectives and scopes. The preparation phase also includes having the selected test-beds and web application security scanners configured and installed. Then, in the execution phase, web application security scanners are configured to scan the test-beds. Lastly, collected experimental results are charted and filed in the reporting phase. The virtualization is common in existing experiment methodologies for reducing both the complexity and cost required, to set up a web application penetration testing lab. The guideline to set up a virtual penetration lab is available in [35].

4.The methodology

The paper conducts the survey based on the PRISMA protocol of [36]. The completeness and transparency of PRISMA protocol have made PRISMA protocol the methodology of this survey. *Figure 2* showed the flow diagram of PRISMA protocol.

PRISMA protocol has an intriguing subject systematically reviewed with these four activities, namely identification, screening, eligibility, and included.

Identification: International conference proceedings and journal papers of six major publishers, which are The Institute of Electrical and Electronic Engineers

(IEEE), Emerald Insight, Association for Computing Machinery (ACM), ScienceDirect, Springer, and Google Scholar, were surveyed in this survey paper. Manuscripts of this area of interest were collected from these publishers using the keywords of 'web application penetration testing', 'automated web application penetration testing', 'web application security scanner', 'web application security testing', 'web vulnerability scanner', and 'web pen-test'. Overall, 290 manuscripts were retrieved with keywords stated above.

Screening: In this screening process, 114 manuscripts were discarded to eliminate the duplication. In the meanwhile, the remaining 144 manuscripts were carried forward for full-text reading.

Eligibility: In this phase, full-text of 144 manuscripts were comprehensively reviewed, to define their eligibility. The process had 54 manuscripts discarded, because of the poorly executed experiment methodology.

Included: Finally, experimental methodologies of elected manuscripts are qualitatively and quantitatively synthesized.

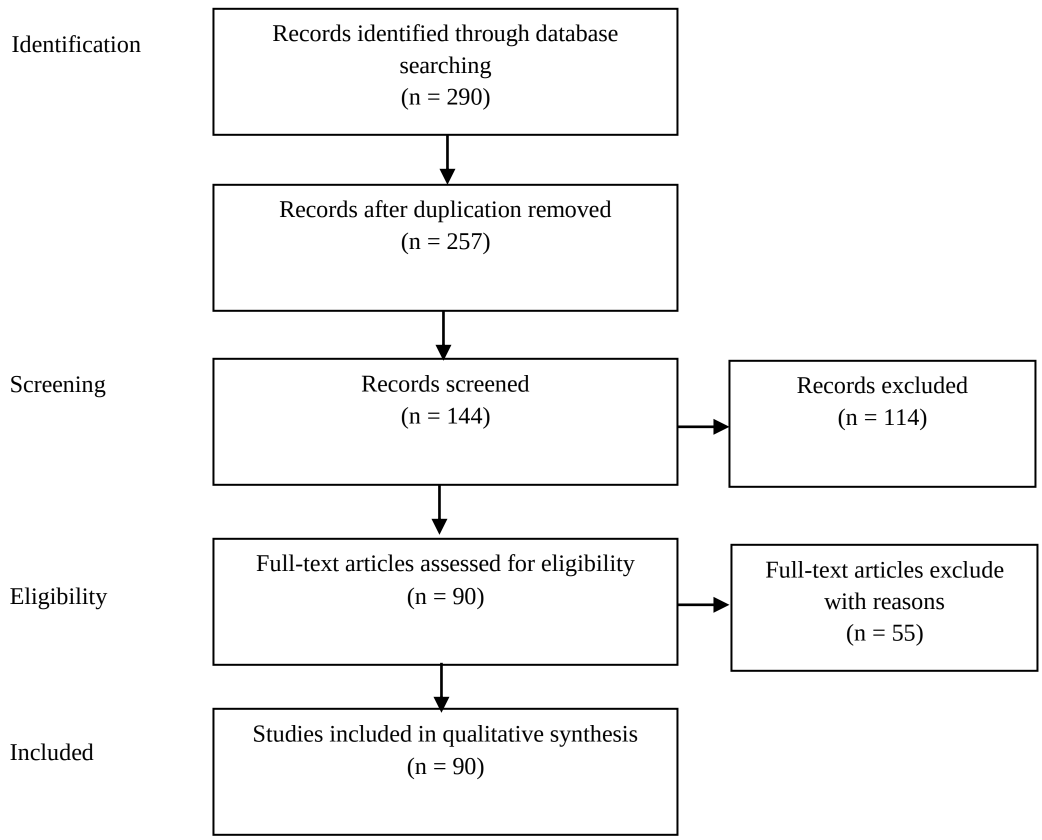

Figure 2 Flow diagram of PRISMA protocol

4.1Data quantification

The final stage of the survey quantitatively synthesizes collected data according to selected indices. The survey categorizes the data based on indices of the year of publication, publisher, web application vulnerability, test-beds, measurement metric, and web application security scanner. This data are quantified to deliver practitioners with the compelling approach of quantifying web application security scanner's quality. In addition to that, it is to provide future works with critical hints of designing next testing framework, measurement metric, test-bed, or model to quantify web application security scanner's quality.

5.Approaches for scanner's quality quantification

The quality of web application security scanner is quantified to achieve these four objectives, which are:

a. To compare the white box and black box web application security scanners' quality.

b. To clarify web application security scanners' strengths and limitations.
c. To benchmark a recently proposed algorithm.
d. To clarify the web application security scanner's test coverage.

In this section, the corresponding experimentations are reviewed to show their findings and methodologies.

5.1White box and black box scanners comparison

The experiment distinguishes the state-of-the-art of white box and black box web application security scanner. It is also to clarify the strengths and limitations of white box and black box web application security scanners in vulnerability detection. The experiments were conducted by having both scanners scanned the same vulnerable web applications.

Subsequently, the obtained experimental results were compared, to clarify their performance. According to experimental results of [9] and [33], white box web application security scanners achieve better test coverage than black box web application security scanners because of the code visibility. Therefore, black box web application security scanner tends to generate the false negatives. However, white box web application security scanners are susceptible to false positives.

5.2Clarifying scanners strengths and limitations

The web application security scanners are quantified to clarify their test coverage, scanning efficiency, attack coverage, and the capability to detect a class of web application vulnerability. The experiment was conducted by configured the scanners to scan selected vulnerable web applications. Summing up experimental results of [6, 7, 18, 22, 23, 26, 30, 34, 37–46], web application security scanners not only tends to generate false alarm, the coverage issue is quite concerning in automated web application penetration testing. Besides this, web application security scanners are exceptionally good in detecting reflected cross-site scripting and SQL injection. Unfortunately, hard work is still required to make web application security scanners detect the advance web application vulnerabilities. Moreover, the coverage issue is yielded, because of the challenge to scan, modern web applications that has rich media.

5.3Benchmarking the algorithms

Experiments are also conducted to validate the recently designed algorithms. The objective is to ensure the algorithm had addressed the targeted research problem. The experiment has the algorithm scan the selected vulnerable web applications. Then, the algorithm is validated by comparing the collected experimental results with those obtained with existing algorithms. [47–49] experimental results showed the proposed code parsing and reverse engineering algorithms are efficient in scraping data entry points (DEPs) and attack vectors from under-test web applications. In the meanwhile, [50–60]'s experimental results showed leveraging of search-based testing technique, mutation testing technique, and genetic algorithm are effective in improving the attack coverage. Moreover, anomaly detection and information flow analysis by [8, 9, 27, 28, 31], and [61–79] are proven effective in detecting the web application vulnerability in either black box or white box testing environment. Besides this, the developed prototypes are validated in [5, 20, 25, 29], and [80−99].

5.4Scanner coverage clarification

These experiments quantify web application security scanners by configured the scanners to crawl selected web applications. Experiment results of [80−82] showed the authors' information knowledge manager (IKM) and topic model manages to increase the number of visited web pages by 28%. In the meanwhile, [100] experimental result showed test coverage is expandable by hooking JavaScript API onto dynamic analysis, to detect registered events, URLs, and web forms.

5.5Related works

Several testing frameworks were proposed by practitioners to quantify web application security scanner's quality. Authors of [101−103] introduced a testing framework that quantifies web application security scanner's quality with fault injection technique. These frameworks defined a web application security scanner's quality by measuring the capability of web application security scanner to detect the faults introduced with fault injection technique. Besides this, [104, 105], and [106] have proposed the alternative measurement metrics to rank web application security scanner's quality. [104] introduced true duplication and false duplication to describe web application security scanner's duplicate results, while [105] proposed the sensitive data flow coverage with an attempt to replace conventional branch coverage and statement coverage. In the meanwhile, [106] introduced the web application security scanner grading system to grade web application security scanner's quality with the fuzzy classifier.

6.Classification of the methodologies

This section classifies methodologies of publishing manuscripts based on selected indices. The selected indices are the type of manuscript, the manuscript's year of publication, the manuscript's publisher, the testing technique of web application security scanner, the web application vulnerability, the test-beds, and measurement metrics used to describe web application security scanner's quality.

6.1The assortment of academic manuscripts

The section classifies collected academic manuscripts to convey publishers that have a high interest in this subject of automated web application penetration testing. Then, this section classifies the manuscripts based on how this area of research is delivered to public. The data showed the relevant area of research were frequently published in six publishers of ScienceDirect, IEEE, ACM, Emerald Insight, Google Scholar, and Springer, which well-known for publishing books, ebooks, and peer-reviewed journals in science, engineering, and computer science. Amongst these publishers, IEEE, Springer, and the ACM have the highest publication rate of 47.8%, 18.9%, and 14.4% respectively. However, only a manuscript is from Emerald Insight, since the publisher is specialist more in fields like business and management, education, and marketing, with only several books series and journals covered the engineering. Unfortunately, relevant researches were frequently published as conference proceedings or symposiums, instead of journal papers with frequencies of 23.3% and 76.7% respectively. The *Figure 3* and *Figure 4* classify manuscripts by publisher and manuscripts' type.

6.2 The assortment of manuscript by publication year

The section classifies related academic manuscripts by their year of publication to convey the research trend of automated web application penetration testing. As depicted in *Figure 5*, this research topic is continuously gaining its popularity, begin from the year of 2000 to 2018. The research topic's popularity is reaching its peak in the year 2014, which 16.7% of the manuscripts were published in that year. Nowadays, the trend of automated web application penetration testing remains attractive with an average of 5 manuscripts were published in the year 2015, 2016, and 2017.

6.3 The assortment of web application security scanners

The section classifies web application security scanners involved in the experiments with their testing technique and licensing to convey web application security scanners that available for automatically assessing web application security, while to deliver those most accessible for benchmarking purpose. Overall, the experiments had quantified 93 web application security scanners, which 87 of them are black box web application security scanners, while 8 of them are white box web application security scanners. The 87 black box web application security scanners showed 29 of them are open-source, 10 of them are closed software while remaining 48 web application security scanners are from academia. On the other hand, the manuscripts only had eight white box web application security scanners' quality quantified, which 5 of them are open-source, and 3 of them are developed in academia. *Table 1* showed the tested web application security scanners.

Figure 6 showed quantification of black box web application security scanners' quality were more often than white box web application security scanners. The reason is collected manuscripts are often describe automated web application penetration testing as a kind of black box software testing technique. In fact, manuscripts have a term called a static analyser to describe the automated white box web application testing. Consequently, the quality of black box web application security scanners called an Acunetix web vulnerability scanner, WebInspect and AppScan are often quantified with frequencies of 11.5%, 6.3%, and 9.9% respectively. In addition to that, 17.8% of manuscripts were tested web application security scanners developed in academia.

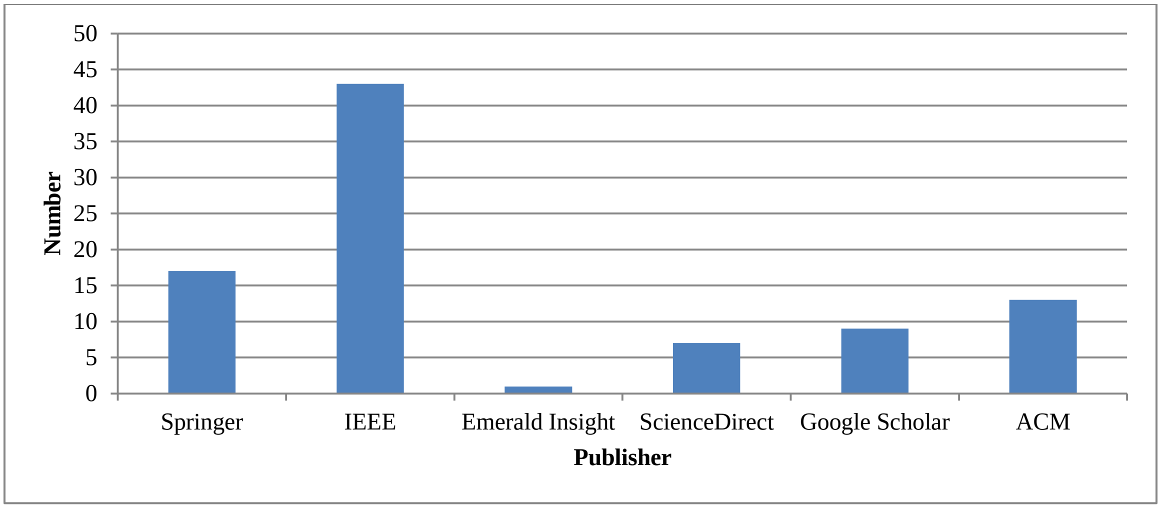

Figure 3 The division of academic manuscripts by publisher

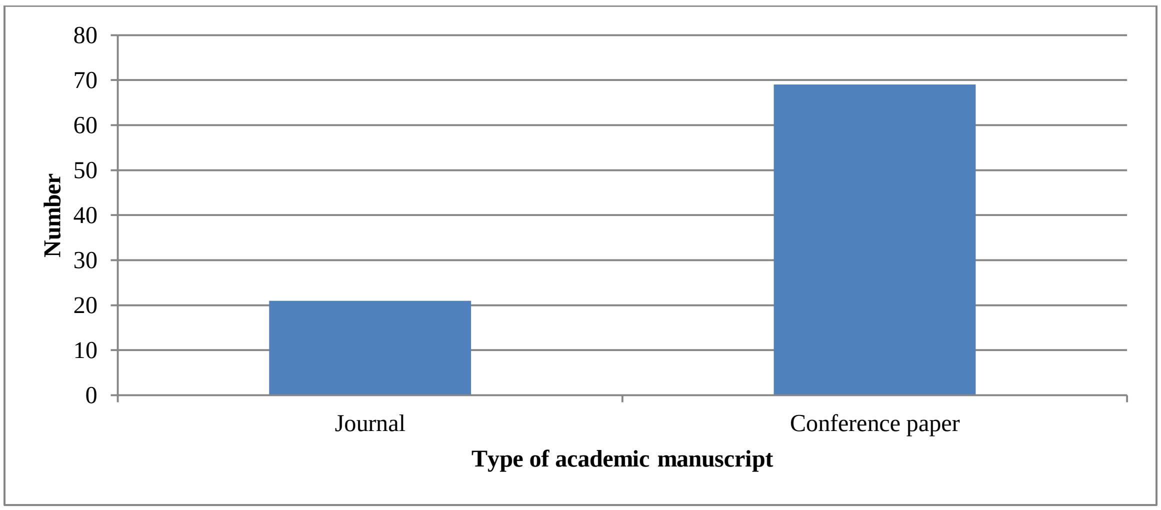

Figure 4 The division of academic manuscripts by the type of document

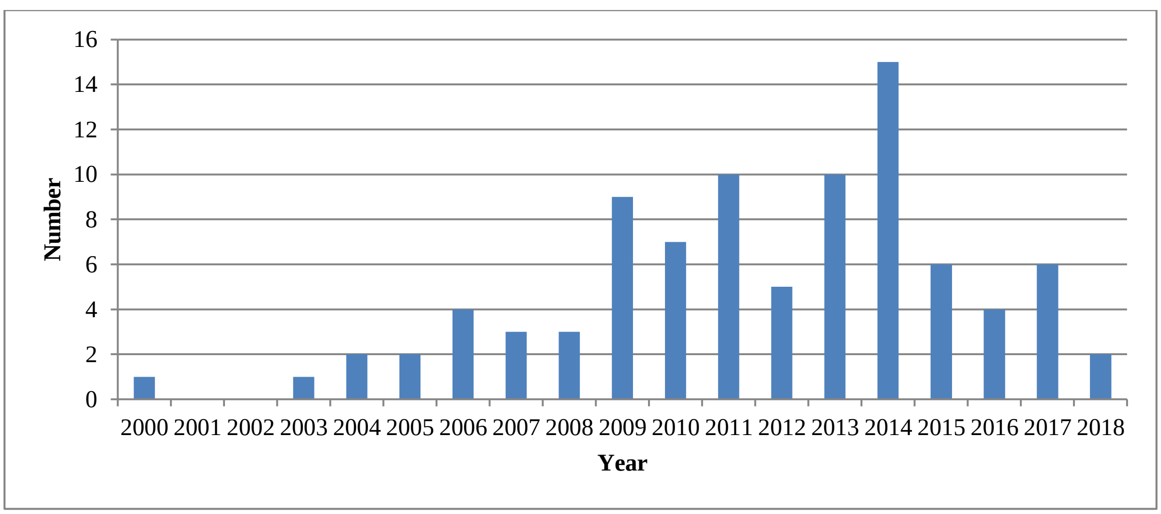

Figure 5 The division of academic manuscript by publication year

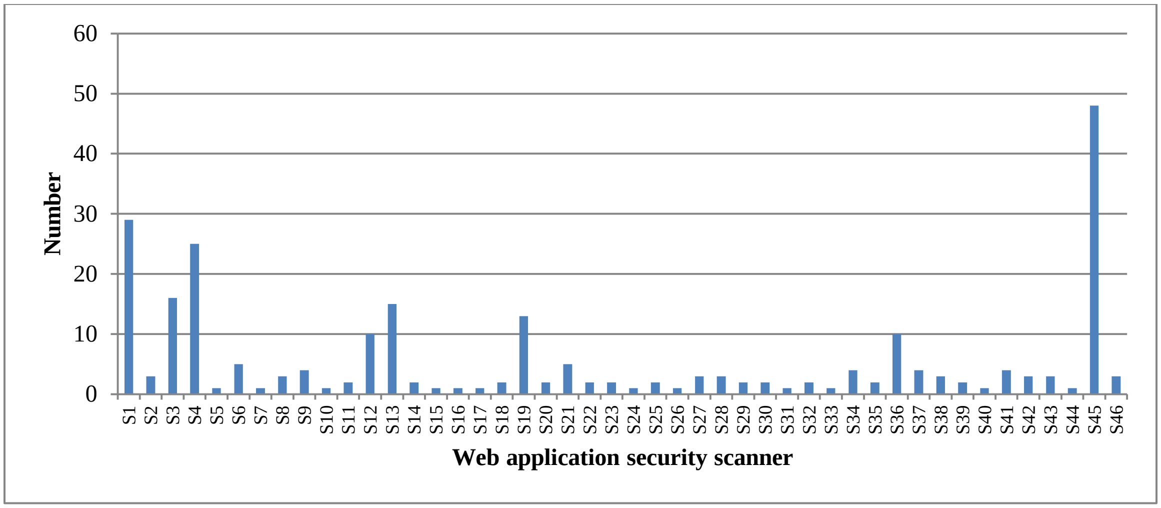

Figure 6 Frequencies "web application security scanner" was evaluated

Table 1 The assortment of web application security scanners by the licensing and testing approach

Licencing	Testing approach	Items	The web application security scanner/ web spider/ parser
Commercialized	Black box	S1	Acunetix Web Vulnerability Scanner
		S2	HailStorm
		S3	WebInspect
		S4	Appscan
		S5	McAfee SECURE
		S6	Qualysguard
		S7	NeXPose
		S8	BurpSuite
		S9	N-Sparker
		S10	Retina
Open source	Black box	S11	Teleport
		S12	Wapiti
		S13	W3af
		S14	WebCruiser
		S15	Wasapy
		S16	PowerFuzz
		S17	WebXSSDetector
		S18	wget
		S19	Skipfish
		S20	Harvest
		S21	Vega
		S22	PownMe
		S23	N-Stalker
		S24	Mikito
		S25	WebScarab
		S26	WebRavor
		S27	WebSPHNIX
		S28	Larbin
		S29	Websecurity
		S30	Web-Glimpse
		S31	SQLfast
		S32	SQLmap
		S33	ARDILLA
		S34	Arachni
		S35	NTOSpider
		S36	ZAP

Licencing	Testing approach	Items	The web application security scanner/ web spider/ parser
		S37	Nikto
		S38	Wikto
		S39	Paros
	White box	S40	Grep
		S41	FindBugs
		S42	Yasca
		S43	IntellJIDEA
		S44	PHPMinerII
Academia	Black box	S45	WAVES, Saner, VS. WS., CIVS-WS, WebSSARI, Andromeda, Multi-agent scanner, Attack injection tool, RWSS, Wasapy, WASC, PAPAS, PIUIVT, Sania, Secubat, ARDILLA, MUBOT, MUSIC, MUTEC, MUFORMAT, XSS analyser, Sign-WS, WS-Attacker, Vulnerability & Injection Tool, WASAPY, Confleagle, SOA-scanner, SQLIVDT, LigRE, ETSSDetector, NVS, WebGuardia, SQLfast, Idea, Volcano, ANOVA, PMVT, THAPS, XquerryFuzzer, JÄK, WAPTT, BIOFUZZ, KamaleonFuzz, CRS, XSSPeeker, Inferential, XiParam, DetLogic
	White box	S46	ITS4, Pixy, WAP

6.4 The assortment of web application vulnerability

The section delivers web application vulnerabilities that detectable with automated web application penetration testing. *Table 2* showed 54 web application vulnerabilities that detectable with automated web application penetration testing. *Table 2* grouped relevant web application vulnerabilities, according to open web application security project (OWASP) Top 10 [107], with brief descriptions are provided to elaborate each class of web application vulnerability. Web application vulnerabilities are left unclassified if it doesn't fit the OWASP top 10.

Table 2 Classification of web application vulnerabilities by OWASP Top 10

OWASP Top 10	Items	Vulnerability	Description (Deriving from OWASP)
Injection attacks	V1	SQL injection	Insertion of SQL queries to modify integrity, availability, confidentiality of database data.
	V2	XPath injection	Compromising of integrity, availability, or confidentiality of data in XML.
	V3	OS command injection	Execution of arbitrary commands in the host operating system through the vulnerable web application.
	V4	Code injection	Execution/interpretation of injected code in the web application.
	V5	Command injection	Execution of command on the host system through a vulnerable web application.
	V6	Script injection	Arbitrary scripts execution.
	V7	XQuery injection	Incorporation of malicious data into XQuery pattern to alter the XQuery logic.
	V8	SSI injection	Manipulation of the file system and process of web server process.
File inclusion	V9	Remote file inclusion	The remote inclusion of file that could bring harm to the target application.
	V10	Local file inclusion	Inclusion local harmful files to the target web application.
	V11	Arbitrary file upload	Upload of the malicious file that can bring harm to the target application.
	V12	Arbitrary file inclusion	The inclusion of malicious file that can bring harm to the target application.
Session related vulnerability	V13	Session fixation	The hijacking of the valid user session.
	V14	Session prediction	Prediction of the session ID values.
	V15	Session hijacking	Exploitation of web session control mechanism.
Broken authentication	V16	Authentication bypass	Bypass web application's authentication scheme.

OWASP Top 10	Items	Vulnerability	Description (Deriving from OWASP)
	V17	Insufficient authentication	Usage of weak passwords or poorly protected application.
Broken authorization	V18	Broken access control	Weakly enforced restrictions for authenticated users.
	V19	Insufficient password discovery	Bypass password authentication schemes with weak password recovery mechanism.
	V20	Insufficient authorization	Authorized users have loosely configured restriction.
Security misconfiguration	V21	SSL misconfiguration	Misconfiguration of the server to force the usage of cryptographic options.
	V22	Insecure temporary file	Creation and usage of insecure temporary files that lead to compromising of application security.
	V23	Predictable resource location	Uncover hidden web content and functionality of target application.
	V24	misconfiguration	Misconfigured application stack.
Using component with known vulnerabilities	V25	Input sanitization	Inappropriate input sanitization functions.
Sensitive data exposure	V26	Path traversal	Accessing files and directories that stored outside the web root directory.
	V27	Error message disclosure	Accidentally reveals of error codes.
	V28	Username/ password disclosure	Reveals of username or password.
	V29	Server path disclosure	Reveals of server's path.
	V30	Information leakage	Reveals of the internal state of the application.
	V31	Insecure object reference	Direct access to protected objects by the user's supplied input.
Insecure deserialization	V32	Code vulnerability	Leveraged of insecure codes.
	V33	Code execution	Execution of injection code by the application.
HTTP manipulation	V34	HTTP response splitting	The inclusion of malicious characters in HTTP response header without being validated.
	V35	Parameter tampering	Manipulation of the value of HTTP parameter.
	V36	Parameter pollution	Supplying of HTTP parameter with the similar name to alter the way application is interpreting the parameter value.
	V37	HTTP request smuggling	Tamper HTTP requests or responses with malformed HTTP requests.
Spoofing	V38	Content spoofing	Defacement of the web application with text injection.
	V39	SOAP spoofing	Defacement of HTTP header element known as SOAPAction.
	V40	WS-addressing spoofing	Adding of routing information to the SOAP header to allow asynchronous communication.
Poisoning	V41	Cache poisoning	Duplicate headers in a single header field.
	V42	Cookie poisoning	Filling in the cookie attribute to make browser send the cookie within the cross-site request.
Uncategorized	V43	Abuse use of functionality	Misused of application functions and features.
	V44	Cross-site scripting	Injection and sending of malicious scripts to the other end user.
	V45	Clickjacking	Transparent or opaque layer for malicious web browsing.
	V46	Buffer overflow	Submission of malicious data to corrupt web application execution stack.
	V47	Cross-site request forgery	Force execution of malicious actions by the web application.
	V48	SOAP/ AJAX attack	Injection of malicious data to alter XMLHttpRequest logic.
	V49	Denial of service	Making resources of web application unavailable.
	V50	Hidden field manipulation	Disabling resources of a web application.
	V51	Drive-by download	Injection of a legitimate web page with malicious code to infect legitimate web page.

OWASP Top 10	Items	Vulnerability	Description (Deriving from OWASP)
	V52	Format string bug	Injection of the input string for evaluating as a command by the web application.
	V53	Unvalidated redirect	Injection of malicious input to trigger malicious URL redirect.
	V54	Insufficient process validation	Failure in enforcing application business logic.
	V55	Logic vulnerabilities	Fault in application logic.

In summary, the manuscripts had quantified web application security scanners' capability in detecting eight injection-based attacks, four file inclusion, three session related vulnerabilities, two broken authentication, three broken authorization, four security misconfiguration, one usage of component with known vulnerability, six data exposure vulnerability, two insecure deserialization, four HTTP manipulation, three spoofing, and eight uncategorized web application vulnerability. In existing manuscripts, the study of SQL injection and cross-site scripting are the most common with frequencies of 32.6% and 22.4% respectively, while 34.8% of the academic manuscripts covered both SQL injection and cross-site scripting. Unfortunately, evaluation of web application security scanners' quality for others web application vulnerabilities is rare as elaborated in *Figure 7*.

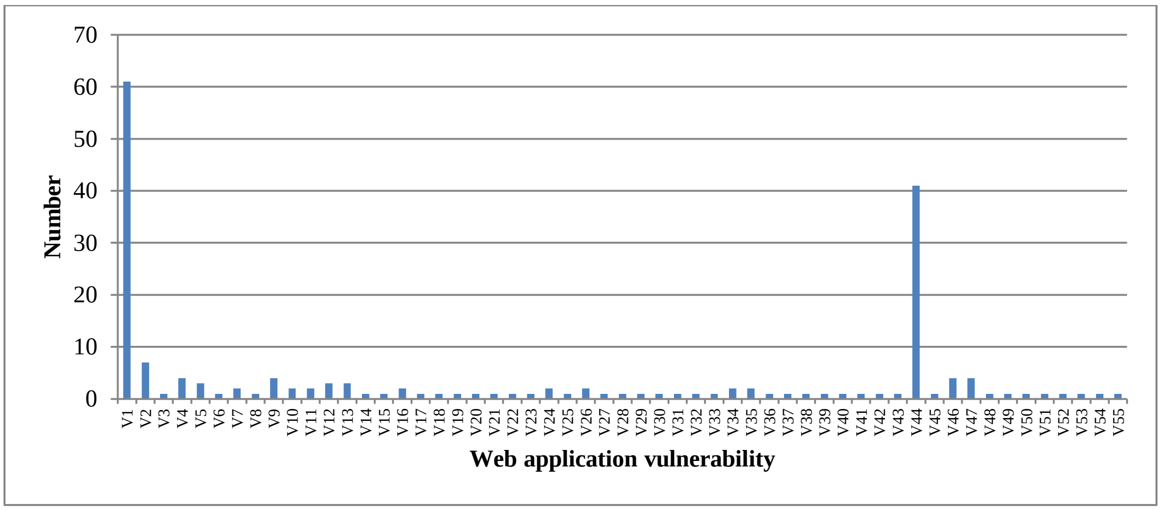

Figure 7 Frequencies "web application vulnerability" was evaluated

6.5 The assortment of test-bed

The section delivers the test-beds that available for benchmarking web application security scanner's quality and their pros and cons. Test-bed is a very vulnerable web application that contained a finite number of vulnerabilities or challenges [26, 37, 41, 103]. The test-bed is having a critical role in benchmarking web application security scanner's quality. Existing experiment methodologies often quantify web application security scanners' quality by configuring the scanners to scan selected test-beds. Then, web application security scanners' vulnerability detection rate or what that most relevant are measured to define their quality. *Table 3* showed four test-beds are available to benchmark web application security scanner quality.

Figure 8 showed 45.6% of experiment methodologies benchmark web application security scanner's quality with open-source web application framework, while 17.3% and 16.3% of experimental methodologies evaluate web application security scanner's quality with educational vulnerable web applications and web application security scanner test sites respectively. Only 5.4% experiment methodologies use custom-made web application developed by students or teaching assistant due to their validity is questionable.

Table 3 Test-beds for benchmarking web application security scanner's quality

Items	Category	Description	Pros	Cons	Test-beds
W1	Custom-made web application	Practitioners developed vulnerable web applications.	•No concern for committing the cybercrime.	•Never validated. •Manual testing is required to validate results validity. •Not well documented.	Custom web applications developed by a group of teaching assistants, researchers, or students
W2	Educational vulnerable web application	Very vulnerable web applications that for educational purpose.	•Web application vulnerabilities are known. •Well documented. •No concern for committing the cybercrime. •Manual testing is not required.	•Limited web application vulnerabilities. •Limited challenges. •Only well-known web application vulnerabilities are testable.	Damn Vulnerable Web Application (DVWA), online bookstore, WebGoat, Gryyere, P0wnMe!, Multillidae, YAVWA, WIVET
W3	Web application security scanner test sites	Test-site that specifically for benchmarking web application security scanner.	•Web applications vulnerability is known. •Well documented. •No concern for committing the cybercrime. •Manual testing is not required.	•Limited web application vulnerabilities. •Limited challenges. •Only well-known web application vulnerabilities are testable.	WackoPicko, PCI, MatchIt, W-VST, Scan-bed
W4	Open-source web application framework	The open-access framework that supports web application development.	•No concern for committing the cybercrime.	• Not well documented. • Manual testing is required to validate result validity.	Drupal, phpBB, WordPress, Django, SatchMo, Vanilla, Gallery, SCARF, Reference, PHPFusion, PHPBlog, PHPNuke, PHPMyAdmin, TikiWiki, PHP Gallery, MyBB, Moodle, TestLink, SquirrelMail, Elgg, FeedSearch, RssReader, LampCMS, Joomla, PhpNN, MediaWiki, OwnCloud, Tidio, Nibbleblog, Modx-CMS
W5	Real-world web application	Web application live on the World Wide Web.	•Web application security scanner's capability can genuinely reveal.	•Cybercrime may be conducted. •Not well documented. •Manual testing is required to validate result validity.	Alexa top ranking sites.

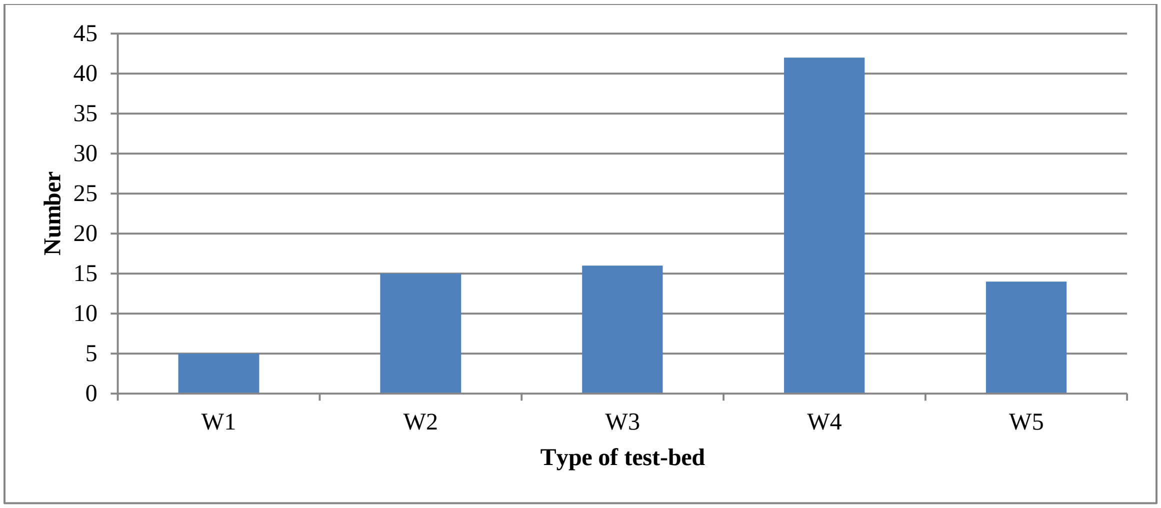

Figure 8 Frequencies the test-bed was leveraged for benchmarking purpose

6.6 The assortment of measurement metrics

The section delivers measurement metrics available for quantifying web application security scanner's quality. Overall, practitioners have 13 measurement metrics to scale web application security scanner's test coverage; 7 measurement metrics to compute web application security scanner's attack coverage; 18 measurement metrics to measure web application security scanner's vulnerability detection rate; 5 measurement metrics to measure web application security scanner's scanning efficiency. The test coverage described the part of a web application that successfully scanned by a web application security scanner. In the meanwhile, attack coverage explains DEP that had been penetrated with attack payloads, while vulnerability detection rate elaborate web application vulnerability that successfully detected by a web application security scanner. On the other hand, scanning efficiency elaborates the time required to complete a vulnerability scanning session. *Table 4* showed the stated measurement metrics used to quantify web application security scanner's quality.

Table 4 Measurement metrics to quantify web application security scanner quality

Criteria	Items	Metrics	Description (By the authors)
Test coverage	M1	Number of URLs	The number of URLs that a web application security scanner had visited.
	M2	Number of networks generated	The number of networks produced by a web application security scanner in a vulnerability scanning session.
	M3	Number of web pages visited	The number of web pages visited by a web application security scanner in a vulnerability scanning session.
	M4	Code coverage	The degree of web application source code that is tested by a web application security scanner.
	M5	Test coverage	The degree of a web application that had been successfully tested by a web application security scanner.
	M6	Number of links	The number of links that a web application security scanner had successfully retrieved.
	M7	Surface coverage	Surface and sink coverage retrieved by a web application security scanner.
	M8	Testing level	Description of the testing approach, either in a black box or white box manner.
	M9	Number of data extracted	The number of data that successfully extracted.
	M10	Capability to bypass authentication scheme	Description of the ability of a web application security scanner in provides an authentication scheme with valid data.
	M11	Reachability scores	The faction of retrieved entry points over the entry points of a web application.
	M12	Number of forms retrieved	The number of web forms that a web application security scanner manages to retrieve.

Criteria	Items	Metrics	Description (By the authors)
Attack coverage	M13	Number of injection point	The number of entry points that retrievable by a web application security scanner.
	M14	Number of vector	The number of inputs used to test a web application.
	M15	Granularity of test case	Description of the object that constitutes a test case.
	M16	Source of test case	Description of artefacts used to generate the test case.
	M17	Test case generation method	Description of approach that converts the source of test cases into a set of test cases.
	M18	Number of attack vector	The number of retrievable paths.
	M19	Number of test case generated	Amount of test cases produced by a web application security scanner in a scanning session.
Vulnerability detection	M20	Number of vulnerability	The number of vulnerability produced by a web application security scanner.
	M21	Number of false positive	The number of unreal vulnerability produced by a web application security scanner.
	M22	Number of false negative	The number of vulnerability missed by a web application security scanner.
	M23	Number of true positive	The number of benign vulnerabilities reported by a web application security scanner.
	M24	Number of true negative	The number of benign vulnerabilities that not reported by a web application security scanner.
	M25	F-measure	Harmonic means of recall and precision.
	M26	Recall	The probability to produce a benign vulnerability.
	M27	Precision	The fraction of benign vulnerability from vulnerabilities reported.
	M28	Detection score	The fraction of vulnerability detected over vulnerabilities that possessed by a test-bed.
	M29	Number of true vulnerability	The number of benign vulnerabilities.
	M30	Number of false alarm	The number of false positives and false negatives
	M31	Detection rate	The ratio of the found vulnerabilities.
	M32	Vulnerability coverage	Amount of vulnerability that detectable.
	M33	Detection coverage	Percentage of detectable vulnerabilities.
	M34	Number of true duplication	The number of duplicate true positives.
	M35	Number of false duplication	The number of duplicate false positives.
	M36	Coverage	The number of vulnerabilities detected.
	M37	Fitness	The number of vulnerability covered by a test case.
Efficiency	M38	Scanning time	Amount of time required to complete a vulnerability scanning.
	M39	Parsing time	Amount of time required to complete parsing a set of codes.
	M40	Automation level	The capability to complete a scanning session without tester involvement.
	M41	Processing overhead	Amount of extra time required to complete a scanning session.
	M42	Productivity	Average time requited to generate a test case.

In existing academic manuscripts, it is common that web application security scanners were quantified to measure their vulnerability detection rate. Therefore, *Figure 9* showed measurement metrics like the number vulnerability, the number of false positives and the number of false negatives are the three most common measurement metrics used to measure web application security scanner's quality with frequencies of 24.2%, 20.8%, and 5.6% respectively. However, [33] had defined F-measure as the most suitable measurement metrics to measure web application security scanner's quality.

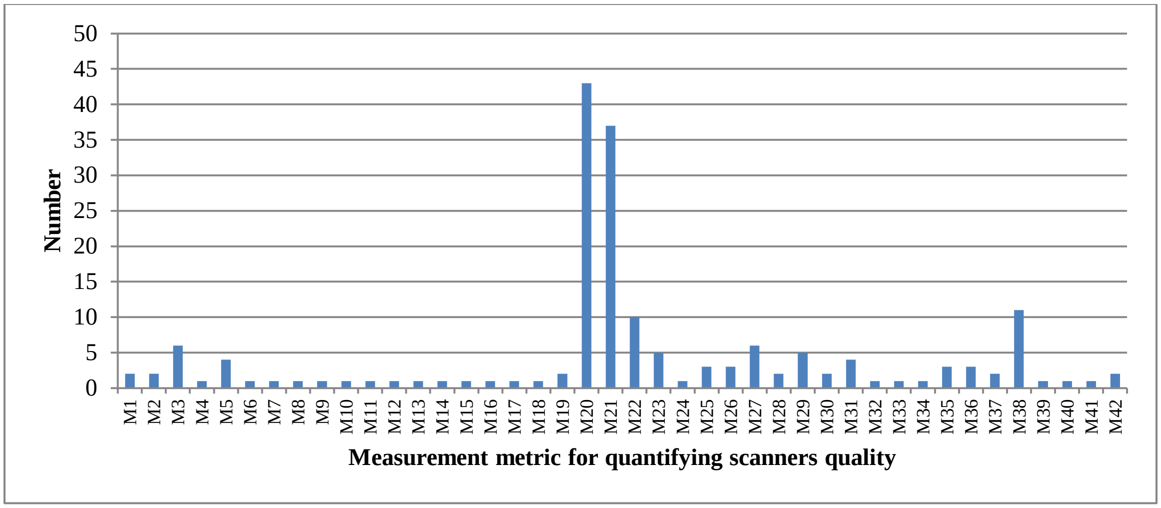

Figure 9 Frequencies "measurement metric" is used to measure web application security scanner quality

7.Conclusion and future work

Quantifying a web application security scanner's quality with sophisticated methodology is essential for the following reasons. Firstly, a sophisticated methodology help in accurately defined web application security scanner's strengths and limitations, especially in locating weakly designed algorithms. Secondly, to deliver a platform to allow researchers scientifically and precisely present their concept, idea, algorithm, or achievement in the field of automated web application penetration testing to the public. Third and the last, precise methodology are significantly effected advancement of this research field. Unfortunately, existing 90 academic manuscripts, neither have a standard methodology nor measurement metric to quantify web application security scanner's quality, although the relevant study is common. There is only a common practice that web application security scanner's quality was quantified by configuring the scanner to scan selected test-beds. Then, practitioner quantified a scanner's quality by calculating the number of vulnerabilities detected. Consequently, the survey showed practitioners use the diverse set of methodologies, test-beds, web application security scanners, and measurement metrics to quantify web application security scanner's quality.

Although the survey has presented the compelling approach to quantify the quality of web application security scanners' quality, as well as exhibit the test-beds, web application security scanners, and measurement metrics to measure web application security scanner's quality. However, the survey delivers more research questions, instead of giving the answer of providing the sophisticated methodology to quantify web application security scanner's quality. For instance, the suitable amount of test-beds or web application security scanners to benchmark a web application security scanner or algorithm is unknown. In existing academic manuscripts, it showed the number of web application security scanners and test-beds used to benchmark a web application security scanner is ranging from the minimum number of zero to the maximum number of a thousand. Besides this, fittest measurement metrics to describe a web application security scanner's test coverage, attack coverage, vulnerability detection rate, and scanning efficiency are also unknown. The survey showed practitioners had quantified web application security scanner's quality with less meaningful and redundant measurement metrics. Practitioners had measured web application security scanner's capability for vulnerability detection with measurement metrics of vulnerability detection rate and the number of vulnerabilities, which carries the same definition. In the meanwhile, measurement metrics of surface coverage and the number of links are too ambiguous to define web application security scanner's test coverage. Since the scope of surface coverage is difficult to define, meantime the number of links cannot represent a web application's coverage because modern web applications not only consist of links but also other web elements that critical for vulnerability assessment. Therefore, there is an assuring future work for this area of this research. It is about producing a compelling methodology and metric system to quantify web application security scanner's quality, to precisely deliver the findings of related research field to practitioners.

Appendix

The assortment of methodologies by objectives

Objective	Prototype scanners	The scanners	Test-beds	Vulnerabilities	Metrics	Authors
Benchmarking recently designed algorithms.	WebSSARI		230 random open-source web applications of SourceForge.	• SQL injection • Cross-site scripting	Not clearly defined.	[80−81]
	Name unknown	• Burp Suite • W3af • Acunetix web vulnerability scanner	• Django basic blog • Django forum • Satchmo online shop	• Reflected cross-site scripting • Persistent cross-site scripting	• The number of injection points.	[98]
	Secubat		• 100 random web applications	• SQL injection. • Cross-site scripting	• The number of web pages. • The number of forms visited. • The number of vulnerabilities.	[85]
	Name unknown	• Wget • W3af • Skipfish	• Gallery • WordPress V.2 • WordPress V.3 • SCARF • Vanilla Forum • WackoPicko	• SQL injection • Cross-site scripting	• Code coverage. • The number of vulnerabilities. • The number of false alarms. • The number of true vulnerabilities.	[99]
	ITS4	• Grep	• I-pay	• C++ and C code vulnerabilities	• Elapsed scanning time.	[86]
	Saner		• Jetbox • MyEasyMarket • PBL GuestBook • PHP-Fusion • SendCard	• Input sanitization function	• The number of vulnerabilities.	[87]
	VS. WS	• WebInspect • AppScan • Acunetix Web Vulnerability Scanner	• 300 random web applications	• SQL injection	• The number of true vulnerabilities. • The number of false positives.	[8]
	CIVS-WS	• Acunetix web vulnerability scanner • AppScan • WebInspect • FindBugs • Yasca • IntelliJIDEA	• ProductDetail • NewProducts • NewCustomer • ChangePaymentMethod • JamesSmith • PhoneDir • Bank • Bank3 • Xoperation	• SQL injection • XPath injection	• The number of true vulnerabilities. • The number of false positives.	[61, 9]
	Name unknown	• WebScarab • Webravor • Acunetix web vulnerability scanner	• RenRen • Kaixin001 • 163.com	• SQL injection	• The number of visited web pages. • The number of true positives. • The number of false positives.	[49]
	Andromeda		• AJAXChat • Altorol • App. A • Blojsom • BlueBlog • Contineo • Dlog • Friki • GestCV • Ginp • JBoard • JpetStore • JugJobs • Photov • StrutsArticle • WebGoat	• Cross-site scripting • SQL injection	• The number of false positives. • The number of true positives. • The number of vulnerabilities. • Elapsed scanning time.	[88]
	XSS analyser		• 15552 server defences	• Cross-site scripting	• The number of vulnerabilities.	[50]
	Pixy		• PHPBlog • PHPNuke • Gallery • PhpMyAdmin	• Cross-site scripting	• The number of false positives. • The number of vulnerabilities.	[5]

Objective	Prototype scanners	The scanners	Test-beds	Vulnerabilities	Metrics	Authors
			• Serendipity • Yapig			
	Multi-agent scanner		• Drupal	• Stored cross-site scripting	• The number of vulnerabilities.	[89]
	Attack Injection Tool	• AppScan • WebInspect	• TikiWiki • phpBB • MyReferences	• SQL injection	• The number of vulnerabilities.	[101]
	Name unknown		• Timeclock-software • RoomPHPlaning • PHP inventory • Green Desktiny • Meshoutbox	• SQL injection • Cross-site scripting	• The number of false positives. • The number of false negatives. • The number of web pages visited. • The number of attack vectors. • The number of vulnerabilities	[20]
	RWSS	• AppScan • WebInspect	• Open-source blogging platform • Open-source customer management	• Not clearly defined	• The number of false positives. • The number of vulnerabilities. • The number of links. • Surface coverage.	[90]
	Name unknown		• Employee directory • Bookstore • Events • Classified • Portal	• Command injection attack	• Precision.	[65]
	Wasapy	• Skipfish • W3af • Wapiti	• phpBB • SecurePage • Hardware Store • Insecure • Damn vulnerable web application	• SQL injection	• The number of vulnerabilities.	[66]
	Wasapy.		• 6 self-developed web applications	• SQL injection	• The number of false positives. • The number of false negative. • Detection rate.	[70]
	WASC		• PHP-Post • Jupiter CMS • PHP Gallery • Absolute path traversal • MyBBoard	• SQL injection • Script injection	• Parsing processing time.	[69]
	PAPAS		50000 unique URLs from public database of Alexa	• Parameter pollution	• The number of vulnerabilities.	[29]
	Name unknown	• WebScarab	• WebGoat	• SQL injection • Cross-site scripting • Cross-site request forgery • Predictable resource location • HTTP request smuggling • HTTP response splitting • Cache poisoning • Denial of service • Content spoofing • Hidden field manipulation • Driver-by download • Information leakage • Session fixation • Insufficient authentication • Insufficient authorization • Brute force	• The number of false positives. • The number of false negatives. • The number of attack vectors, • Detection rate. • False alarm rate.	[91]
	PIUIVT	• Nikto2 • Wikto	• MvnForum	• SQL injection • Cross-site scripting	• The number of vulnerabilities.	[51]
	Sania	• Paros	• E-learning	• SQL injection	• The number of	[52]

Objective	Prototype scanners	The scanners	Test-beds	Vulnerabilities	Metrics	Authors
			• Bookstore • Portal • Event • Classified • Employee directory		false positives. • The number of vulnerabilities.	
	Sign-WS	• WebInspect • Rational AppScan • Acunetix web vulnerability scanner	• TPC-APP • TPC –C • TPC-W	• SQL injection	• Detection coverage. • The number of false positives.	[62]
	WS-Attacker		• Apache Axis • JBossWS Native • JBossWS CXF • .NET web service	• SOAP action spoofing • WS-addressing spoofing	• The number of vulnerabilities.	[92]
	Name unknown		• Top 1000 websites from Alexa	• Clickjacking attack	• Detection rate. • The number of true positives. • The number of false positives.	[93]
	Vulnerability & injection tool		• TikiWiki • phpBB • MyReferences	• SQL injection	• Test coverage. • The number of false positives.	[102]
	Name unknown		3 custom web applications	• SQL injection • Cross-site scripting • Cookie poisoning • Iframe session • Session hijacking	Not clearly defined.	[94]
	Confleagle	• W3af • Skipfish • WebSecurity	• SquirrelMail • Gallery • myBB • TestLink • phpMyAdmin • Elgg • Moodle • SugarCRM • MediaWiki	• Misconfiguration	• The number of vulnerabilities.	[95]
	SOA-Scanner		• TV-Shows • FeedRegistry • TvHelper • FeedSearch • RssFeeder	• SQL injection • XPath injection	• The number of false positives. • Test coverage.	[71]
	SQLIVDT	• W3af • Nikto • Wapiti • Vega • ZAP • Acunetix web vulnerability scanner	• 3 self-developed web application by 7master students and 2 teaching assistant	• SQL injection	• The number of vulnerabilities.	[72]
	LiGRE	• PownMe • Wapiti • W3af • Skipfish	• WebGoat • Gruyere • WordPres • Elgg • phpBB • E-Health • P0wnMe!	• Cross-site scripting	• The number of vulnerabilities. • The number of false positive. • The number of false negative.	[53]
	ETSSDetector	• Acunetix web vulnerability scanner • N-Stalker • WebCruisher • PowerFuzz • WebSecurify • WebXSSDetector	• Testphp • Webscantest	• Cross-site scripting	• The number of vulnerabilities. • Elapsed scanning time. • The number test generates.	[54]
	Name unknown	• W3af • Wapiti	• LampCMS	• Crawling AJAX web application	• The number of web pages. • Elapsed scanning time.	[47]
	NVS	• Acunetix Web Vulnerability Scanner • NetSparker • Web Cruiser	• Karnel Travel • Online Real State • ICC World Cup II • Online tutorial • Graphics • Travel	• SQL injection	• The number of false positives. • The number of vulnerabilities. • Elapsed scanning time.	[73]

Objective	Prototype scanners	The scanners	Test-beds	Vulnerabilities	Metrics	Authors
	Name unknown	• Acunetix web vulnerability scanner • WebInspect • AppScan	• Jobsite • Education • TPC-App • TPC-C • TPC-W	• SQL injection • XPath injection	• The number of false positives. • Test coverage.	[63]
	Name unknown	• Vega • ZAP Proxy • Mikito • Wapiti • Acunetix web vulnerability scanner • W3af • AppScan	• HR • Farm • News	• SQL injection	• The number of vulnerabilities.	[31]
	WebGuardia	Not clearly defined	• WackoPicko	• SQL injection • Cross-site scripting • Unvalidated redirect • Secure direct object references • Security misconfiguration	• The number of vulnerabilities. • The number of false positives. • The number of false negatives.	[96]
	WAP	• Pixy • PhpMinerII	• phpMyAdmin • Multillidae	• SQL injection • Cross-site scripting	• The number of vulnerabilities. • The number of false positives. • Elapsed scanning time.	[27]
	Name unknown	• WebInspect • AppScan	• ProductDetail • NewProducts • NewCustomer • ChangePayment Method	• SQL injection	• Detection coverage. • The number of false positives.	[64]
	Name unknown	• Wasapy • Skipfish • W3af • Wapiti • AppScan • Acunetix Web Vulnerability Scanner • WebInspect	• phpBB-3 • SecurePage • HardwareStore • Insecure • Damn vulnerable web application (DVWA) • Cyphor • Seagull • Ftss • Rioptx • Pligg	• SQL injection	• The number of vulnerabilities, • The number of false positives.	[68]
	• WebSSARI • WAVES	• Teleport • WebSpnix • Larbin • Web-Glimpse	• 230 random web applications of SourceForge	• SQL injection • Cross-site scripting	• The number of vulnerabilities.	[84]
	SQLfast		• WebGoat • Damn vulnerable web application (DVWA) • Joomla! • Yet another vulnerable web application (YAVWA)	• SQL injection	• The number of data extracted. • Capability to bypass authentication scheme.	[79]
	Idea	• SQLfast	• WAVSEP	• SQL injection	• The number of false positives. • The number of vulnerabilities.	[55]
	Name unknown	• FindBugs	• ChangePaymentMethod • NewCustomer • NewProducts • ProductDetail	• SQL injection • XPath injection	• The number of vulnerabilities. • The number of false positives.	[97]
	Volcano		• Web applications from cyber security bulletin	• SQL injection	• The number of vulnerabilities.	[105]
	ANOVA		• APhpKb • PhpPlanner • Yapig • Mantis	• Cross-site scripting	• Coverage. • Fitness. • Time. • Productivity.	[74]
	PMVT	• Rational AppScan • NTOSpider • W3af • Skipfish • Arachni	• Stud-e	• Multi-step cross-site scripting	• Coverage. • Fitness. • Time. • Productivity.	[74]

Objective	Prototype scanners	The scanners	Test-beds	Vulnerabilities	Metrics	Authors
	jÄk	• Skipfish • W3af • Wget • State-aware crawler • Crawljax	• WIVET • Joomla • Modx-CMS • Nibbleblog • WordPress • Tidio • myBB • phpNN • Gallery • Piwigo • OwnCloud • MediaWiki	• SQL injection • Cross-site scripting	• The number of tests.	[100]
	THAPS		• WordPress	• SQL injection • Cross-site scripting	• The number of false positives. • The number of vulnerabilities.	[48]
	Name unknown	• Acunetix web vulnerability scanner • WatchFire AppScan • WebInspect	• MyReferences	• SQL injection • Cross-site scripting	• The number of vulnerabilities.	[103]
	XqueryFuzzer	• ZAP Attack Proxy	• Bookstore • Classified • WIVET	• XQuery injection	• The number of vulnerabilities.	[57]
	Name unknown.	• Acunetix web vulnerability scanner • NetSparker	Not clearly defined	• SQL injection • Buffer overflow • Cross-site scripting • Cross-site request forgery	• The number of false positives. • The number of false negatives. • Elapsed scanning time.	[75]
	Name unknown	• Nikto • Wikto	• phpBB	Not clearly defined	• Detection rate. • The number of false positives.	[76]
	WAPTT	• W3af • Nikto • Wapiti • Vega • ZAP Proxy • Acunetix web vulnerability scanner	3 vulnerable web application from postgraduate students and teaching assistants.	• SQL injection • Cross-site scripting • Buffer overflow	• The number of vulnerabilities.	[28]
	BIOFUZZ	• ARDILLA • SQLmap	• WebChess • Schoolmate • FaqForge • geccBBlite • phpMyAddressBook • Elemate	• SQL injection	• The number of vulnerabilities.	[58]
	KamaleonFuzz	• P0wnMe • W3af • Wapiti • Skipfish	• P0wnMe! • WebGoat • Gruyer • WordPress • Elgg • phpBB • E-Health	• Cross-site scripting	• The number of false positives. • The number of vulnerabilities.	[59]
	Cross-request scanner (CRS)		• HSBC • BEA • BOC • HSB • CitiBank • Webjet • JetStar	• Parameter tampering	• The number of true positives. • The number of true negatives. • The number of false positives. • The number of false negatives.	[25]
	XSS Peeker	• Acunetix web vulnerability scanner • NetSparker • N-Stalker • NTOSpider • Skipfish • W3af	• WackoPicko. • Custom developed web applications.	• Cross-site scripting.	• The number of vulnerabilities. • The number of attack payloads.	[23]
	Inferential	• Acunetix web vulnerability scanner • SQLMap • AppScan	• WAVSEP	• SQL injection	• The number of false positives. • The number of true positives. • The number of	[77]

Objective	Prototype scanners	The scanners	Test-beds	Vulnerabilities	Metrics	Authors
					URLs. • False positive rate.	
Quantification of scanners quality.	XiParam		• 5 web applications from GotoCode • Custom developed web applications	• XQuery injection • Parameter tampering	• The number of vulnerabilities. • The number of attack requests. • The number of successful attacks. • The number of vulnerable forms. • The number of false positives. • The number of false negatives.	[60]
	Not clearly defined		1854 PHP projects on Github	• SQL injection • Command injection • Code injection • Arbitrary file read/ write • Cross-site scripting • Session fixation	• The number of sinks. • The number of calls.	[78]
	DetLogic	• LogicScope	• WackoPicko • Scarf • OpenIT • Puzzlemall	• Logic flaws	• The number of URLs. • The number of Forms. • The number of vulnerabilities. • The number of false positives. • The number of false negatives.	[108]
		• Acunetix web vulnerability scanner • HailStorm • WebInspect • Rational AppScan • McAfee SECURE • QualysGuard.PCI • NeXPose	• Drupal • phpBB • WordPress	• SQL injection • Cross-site scripting • Arbitrary file upload • Remote file inclusion • OS command injection • Code injection • Session fixation • Session prediction • Authentication bypass • Cross-site request forgery • SSL misconfiguration • Insecure HTTP methodologies • Insecure temporary file • Path traversal • Source code disclosure • Error message disclosure	• Elapsed scanning time. • The number of generated network. • The number of vulnerabilities. • The number of false positives.	[37]
		Not clearly defined	• WackoPicko	• SQL injection • Cross-site scripting • Code injection • Broken access control	• Elapsed scanning time. • Detection score. • Reachability score.	[46]
		• Acunetix web vulnerability scanner • AppScan • WebInspect • Qualys	• 27 custom developed web applications	• SQL injection. • Cross-site scripting • Information leakage • Cross-site request forgery	• The number of vulnerabilities. • The number of false positives.	[38]
		• AppScan • Acunetix web vulnerability scanner • WebInspect	• 300 random web applications	• SQL injection • XPath injection • Code execution • Buffer overflow • Username/ password disclosure	• The number of vulnerabilities. • The number of false positives. • Test coverage.	[7]

Objective	Prototype scanners	The scanners	Test-beds	Vulnerabilities	Metrics	Authors
				• Server path disclosure		
		• Acunetix web vulnerability scanner • AppScan • BurpSuite • HailStorm • Retina • Qualys • WebInspect	• Vendor's test sites	• SQL injection • Cross-site scripting • Authentication bypass • Command injection • XPath injection • SOAP/ AJAX attack • Cross-site request forgery • HTTP response splitting • Arbitrary file upload • Remote file inclusion	• The number of vulnerabilities. • Elapsed scanning time. • The number of false positives. • The number of false negatives.	[6]
		• AppScan • HailStorm. • Acunetix web vulnerability scanner	• Hackme • OWASP Site Generator Project. • WebGoat	• File inclusion • SQL injection • Cross-site scripting	• The number of false positives. • The number of vulnerabilities.	[34]
		• Splat • WAVES • Secubat • ARDILLA • MUBOT • MUSIC • Wilela's prototype • Tappenden's prototype • Salas's prototype • Breech's prototype • Offutt's prototype • McAllister's prototype • MUFORMAT • MUTEC	• Not defined	• Buffer overflow • SQL injection • Format string bug • Cross-site scripting	• Vulnerability coverage. • Test automation level. • Testing level. • Granularity of test cases. • Source of test case. • Test case generation method.	[39]
		• Acunetix web vulnerability scanner • AppScan • QualysGuard	• PCI • WackoPicko • MatchIt	• Stored SQL injection	• Traffic of scanners.	[40]
		• AppScan • WebInspect • Paros • Acunetix web vulnerability scanner	• W-VST	• Not clearly defined	• F-measure. • Precision. • Recall.	[18]
		• Acunetix web vulnerability scanner • AppScan • ZAP	• WackoPicko • Scan-bed	• Stored SQL injection • Stored cross-site scripting	• The number of attack vectors.	[41]
		Not clearly defined	• W-VST	Not clearly defined	• The number of true duplication. • The number of false duplication.	[104]
		• Zap attack proxy. • Skipfish.	• Damn vulnerable web application (DVWA) • Web application scanner evaluation project (WAVSEP)	• Cross-site scripting • SQL injection • File inclusion	• Precision • The number of false positives.	[30']
		• SAMATE	• CBMC • K8-sight • Pcline • Prevent • SCA • Gianna • Cx-enterprise • Codesonar	Not clearly defined	• Precision. • Recall. • F-measure.	[42]
		• Acunetix Web Vulnerability Scanner. • AppScan. • QualysGuard.	• MatchIt • PCI • WackoPicko	• Persistent SQL injection	• The number of vulnerabilities.	[26]

Objective	Prototype scanners	The scanners	Test-beds	Vulnerabilities	Metrics	Authors
		• BurpSuite • ZAP Proxy	• WebGoat • Multillidae II • Damn vulnerable web application (DVWA) • Bodgeit • Gruyere	• Cross-site scripting	• Coverage.	[43]
		• Arachni • Wapiti • Skipfish	• WAVSEP • AltoroMutual • Web scanner test site • WIVET • Acunetix test sites	• SQL injection • Cross-site scripting	• Crawler coverage. • True positive rate. • True negative rate. • False positive rate. • False negative rate. • Positive predictive values. • Negative predictive values. • False omission rate. • Accuracy. • F-measure. • Scanning speed. • Vulnerability detection accuracy.	[44]
		• Acunetix web vulnerability scanner • BurpSuite • ZAP Proxy • NetSparker • AppSpider • Arachni • Vega • Wapiti • Skipfish • ironWASP • W3af	• WAVSEP	• SQL injection • Cross-site scripting • Remote file inclusion • Path traversal / local file inclusion	• Precision. • Recall. • F-measure. • The number of false positives. • The number of false negatives. • The number of true positives.	[45]
		• Vega • Arachni • ZAP Proxy	• Multillidae II • Butterfly project • WackoPicko • DVWA • Juice hop	• Null byte • SQL injection • Insufficient password recovery • Code injection • SSI injection • Abuse of functionality • XPath injection • Insufficient process validation	• Detection rate.	[22]
Web application security canners comparison.		• WebInspect • AppScan • Acunetix web vulnerability scanner • FindBugs • Yasca • IntellijIDEA	• TPC-APP service • TPC-C web service • TPC-W web service	• SQL injection	• Precision. • Recall. • F-measure.	[33]
Quantification of scanner coverage.	WAVES	• Teleport • Web Sphnix • Harvest • Larbin • Web-Glimpse • Google	• NAI • Lucent • Trend Macro • Palm • Olympic • Apache • Verisign • Ulead • Cert • Maxtor • Mazda • Linux Journal • Cadillac • Web500	• SQL injection • Cross-site scripting	• The number of webpage.	[82, 83]

Acknowledgment

We would like to express our gratitude to Dr Nilashi Mesbah for his contribution in sharing the knowledge. Besides this, we would also like to express our appreciation to Miss Hazinah Kutty Mammi upon her help in improving the paper quality and readability.

Conflicts of interest

The authors have no conflicts of interest to declare.

References

[1] Roche X. Httrack website copier. Citato a. 2012.
[2] Hai-Jew S. Conducting surface web-based research with maltego carbon. Retrieve from: http://scalar.usc.edu/works/conducting-surface-web-based-research-with-maltego-carbon/index. Accessed 15 May 2018.
[3] https://www.acunetix.com/Websitesecurity/Cros%20s-Site-Scripting. Accessed 15 May 2018.
[4] Meucci M, Keary E, Cuthbert D. The OWASP testing guide v2. OWASP Foundation 2008.
[5] Jovanovic N, Kruegel C, Kirda E. Pixy: a static analysis tool for detecting web application vulnerabilities (short paper). Symposium on security and privacy 2006 (pp.258-63). IEEE.
[6] Suto L. Analyzing the accuracy and time costs of web application security scanners. San Francisco. 2010.
[7] Vieira M, Antunes N, Madeira H. Using web security scanners to detect vulnerabilities in web services. In international conference on dependable systems & networks 2009 (pp. 566-71). IEEE.
[8] Antunes N, Vieira M. Detecting SQL injection vulnerabilities in web services. In Latin-American symposium on dependable computing 2009 (pp. 17-24). IEEE.
[9] Antunes N, Vieira M. Comparing the effectiveness of penetration testing and static code analysis on the detection of SQL injection vulnerabilities in web services. In international symposium on dependable computing 2009 (pp. 301-6). IEEE.
[10] Antunes N, Vieira M. Defending against web application vulnerabilities. Computer. 2012; 45(2):66-72.
[11] http://projects.webappsec.org/w/page/13246986/Web%20Application%20Security%20Scanner%20Evaluation%20Criteria. Accessed 25 February 2018.
[12] Black PE, Fong E, Okun V, Gaucher R. Software assurance tools: web application security scanner functional specification version 1.0. Special Publication, National Institute of Standards and Technology. 2008.
[13] Qianqian W, Xiangjun L. Research and design on web application vulnerability scanning service. In international conference on software engineering and service science 2014 (pp. 671-4). IEEE.
[14] Baral P. Web application scanners: a review of related articles [Essay]. IEEE Potentials. 2011; 30(2):10-4.
[15] Fong E, Okun V. Web application scanners: definitions and functions. In annual Hawaii international conference on system sciences 2007. IEEE.
[16] Curphey M, Arawo R. Web application security assessment tools. IEEE Security & Privacy. 2006; 4(4):32-41.
[17] Tian-yang G, Yin-Sheng S, You-yuan F. Research on software security testing. World Academy of Science, Engineering and Technology. 2010; 4(9):1446-50.
[18] Tung YH, Tseng SS, Shih JF, Shan HL. W-VST: a testbed for evaluating web vulnerability scanner. In international conference on quality software 2014 (pp. 228-33). IEEE.
[19] Gol D, Shah N. Detection of web application vulnerability based on RUP model. In national conference on recent advances in electronics & computer engineering 2015 (pp. 96-100). IEEE.
[20] Chen JM, Wu CL. An automated vulnerability scanner for injection attack based on injection point. In international computer symposium 2010 (pp. 113-8). IEEE.
[21] Alssir FT, Ahmed M. Web security testing approaches: comparison framework. In proceedings of the international congress on computer applications and computational science 2012 (pp. 163-9). Springer, Berlin, Heidelberg.
[22] Muñoz FR, Cortes II, Villalba LJ. Enlargement of vulnerable web applications for testing. The Journal of Supercomputing. 2017:1-20.
[23] Bazzoli E, Criscione C, Maggi F, Zanero S. XSS peeker: a systematic analysis of cross-site scripting vulnerability scanners. arXiv preprint arXiv:1410.4207. 2014.
[24] Patil S, Marathe N, Padiya P. Design of efficient web vulnerability scanner. In international conference on inventive computation technologies 2016 (pp. 1-6). IEEE.
[25] Fung AP, Wang T, Cheung KW, Wong TY. Scanning of real-world web applications for parameter tampering vulnerabilities. In proceedings of the ACM symposium on information, computer and communications security 2014 (pp. 341-52). ACM.
[26] Khoury N, Zavarsky P, Lindskog D, Ruhl R. Testing and assessing web vulnerability scanners for persistent SQL injection attacks. In proceedings of the first international workshop on security and privacy preserving in e-societies 2011 (pp. 12-8). ACM.
[27] Medeiros I, Neves NF, Correia M. Automatic detection and correction of web application vulnerabilities using data mining to predict false positives. In proceedings of the international conference on world wide web 2014 (pp. 63-74). ACM.
[28] ĐURIĆ Z. WAPTT-Web application penetration testing tool. Advances in Electrical and Computer Engineering. 2014; 14(1):93-102.
[29] Balduzzi M, Gimenez CT, Balzarotti D, Kirda E. Automated discovery of parameter pollution vulnerabilities in web applications. In NDSS 2011.
[30] Makino Y, Klyuev V. Evaluation of web vulnerability scanners. In international conference on intelligent

data acquisition and advanced computing systems: technology and applications 2015 (pp. 399-402). IEEE.
[31] Aliero MS, Ghani I. A component based SQL injection vulnerability detection tool. In Malaysian software engineering conference 2015 (pp. 224-9). IEEE.
[32] Auronen L. Tool-based approach to assessing web application security. Helsinki University of Technology. 2002 (pp. 1-20).
[33] Antunes N, Vieira M. Benchmarking vulnerability detection tools for web services. In international conference on web services 2010 (pp. 203-10). IEEE.
[34] Fong E, Gaucher R, Okun V, Black PE, Dalci E. Building a test suite for web application scanners. In proceedings of the Hawaii international conference on system sciences 2008 (pp. 1-8). IEEE.
[35] Cardwell K. Building virtual pentesting labs for advanced penetration testing. Packt Publishing Ltd; 2014.
[36] Moher D, Liberati A, Tetzlaff J, Altman DG. Preferred reporting items for systematic reviews and meta-analyses: the PRISMA statement. Annals of Internal Medicine. 2009; 151(4):264-9.
[37] Bau J, Bursztein E, Gupta D, Mitchell J. State of the art: automated black-box web application vulnerability testing. In symposium on security and privacy 2010 (pp. 332-45). IEEE.
[38] Bau J, Wang F, Bursztein E, Mutchler P, Mitchell JC. Vulnerability factors in new web applications: audit tools, developer selection & languages. Stanford, Tech. Rep. 2012.
[39] Shahriar H, Zulkernine M. Automatic testing of program security vulnerabilities. In international conference on computer software and applications 2009 (pp. 550-5). IEEE.
[40] Khoury N, Zavarsky P, Lindskog D, Ruhl R. An analysis of black-box web application security scanners against stored SQL injection. In third international conference on privacy, security, risk and trust (PASSAT) and social computing (SocialCom) 2011 (pp. 1095-101). IEEE.
[41] Parvez M, Zavarsky P, Khoury N. Analysis of effectiveness of black-box web application scanners in detection of stored SQL injection and stored XSS vulnerabilities. In international conference for internet technology and secured transactions 2015 (pp. 186-91). IEEE.
[42] Díaz G, Bermejo JR. Static analysis of source code security: assessment of tools against SAMATE tests. Information and Software Technology. 2013; 55(8):1462-76.
[43] Garn B, Kapsalis I, Simos DE, Winkler S. On the applicability of combinatorial testing to web application security testing: a case study. In proceedings of the workshop on joining AcadeMiA and industry contributions to test automation and model-based testing 2014 (pp. 16-21). ACM.
[44] Alsaleh M, Alomar N, Alshreef M, Alarifi A, Al-Salman A. Performance-based comparative assessment of open source web vulnerability scanners. Security and Communication Networks. 2017:1-14.
[45] Idrissi SE, Berbiche N, Guerouate F, Shibi M. Performance evaluation of web application security scanners for prevention and protection against vulnerabilities. International Journal of Applied Engineering Research. 2017; 12(21):11068-76.
[46] Doupé A, Cova M, Vigna G. Why Johnny can't pentest: An analysis of black-box web vulnerability scanners. In international conference on detection of intrusions and malware, and vulnerability assessment 2010 (pp. 111-31). Springer, Berlin, Heidelberg.
[47] Huiyao A, Yang S, Tao Y, Hui L, Peng Z, Jun Z. A new architecture of AJAX web application security crawler with finite-state machine. In international conference on cyber-enabled distributed computing and knowledge discovery 2014 (pp. 112-7). IEEE.
[48] Jensen T, Pedersen H, Olesen MC, Hansen RR. Thaps: automated vulnerability scanning of PHP applications. In Nordic conference on secure IT systems 2012 (pp. 31-46). Springer, Berlin, Heidelberg.
[49] Wang X, Wang L, Wei G, Zhang D, Yang Y. Hidden web crawling for SQL injection detection. In international conference on broadband network and multimedia technology 2010 (pp. 14-8). IEEE.
[50] Tripp O, Weisman O, Guy L. Finding your way in the testing jungle: a learning approach to web security testing. In proceedings of the international symposium on software testing and analysis 2013 (pp. 347-57). ACM.
[51] Li N, Xie T, Jin M, Liu C. Perturbation-based user-input-validation testing of web applications. Journal of Systems and Software. 2010; 83(11):2263-74.
[52] Kosuga Y, Kono K, Hanaoka M, Hishiyama M, Takahama Y. Sania: syntactic and semantic analysis for automated testing against SQL injection. In computer security applications conference 2007 (pp. 107-17). IEEE.
[53] Duchene F, Rawat S, Richier JL, Groz R. LigRE: reverse-engineering of control and data flow models for black-box XSS detection. In working conference on reverse engineering 2013 (pp. 252-61). IEEE.
[54] Rocha TS, Souto E. ETSSDetector: a tool to automatically detect cross-site scripting vulnerabilities. In international symposium on network computing and applications 2014 (pp. 306-9). IEEE.
[55] Dao TB, Shibayama E. Idea: automatic security testing for web applications. In international symposium on engineering secure software and systems 2009 (pp. 180-4). Springer, Berlin, Heidelberg.
[56] Avancini A, Ceccato M. Comparison and integration of genetic algorithms and dynamic symbolic execution for security testing of cross-site scripting vulnerabilities. Information and Software Technology. 2013; 55(12):2209-22.
[57] Palsetia N, Deepa G, Khan FA, Thilagam PS, Pais AR. Securing native XML database-driven web applications from XQuery injection vulnerabilities. Journal of Systems and Software. 2016; 122:93-109.

[58] Thomé J, Gorla A, Zeller A. Search-based security testing of web applications. In proceedings of the international workshop on search-based software testing 2014 (pp. 5-14). ACM.
[59] Duchene F, Rawat S, Richier JL, Groz R. KameleonFuzz: evolutionary fuzzing for black-box XSS detection. In proceedings of the conference on data and application security and privacy 2014 (pp. 37-48). ACM.
[60] Deepa G, Thilagam PS, Khan FA, Praseed A, Pais AR, Palsetia N. Black-box detection of XQuery injection and parameter tampering vulnerabilities in web applications. International Journal of Information Security. 2018:17(1):105-20.
[61] Antunes N, Laranjeiro N, Vieira M, Madeira H. Effective detection of SQL/XPath injection vulnerabilities in web services. In international conference on services computing 2009 (pp. 260-7). IEEE.
[62] Antunes N, Vieira M. Enhancing penetration testing with attack signatures and interface monitoring for the detection of injection vulnerabilities in web services. In international conference on services computing 2011 (pp. 104-11). IEEE.
[63] Antunes N, Vieira M. Penetration testing for web services. Computer. 2014; 47(2):30-6.
[64] Antunes N, Vieira M. Designing vulnerability testing tools for web services: approach, components, and tools. International Journal of Information Security. 2017; 16(4):435-57.
[65] Su Z, Wassermann G. The essence of command injection attacks in web applications. In SIGPLAN notices 2006 (pp. 372-82). ACM.
[66] Dessiatnikoff A, Akrout R, Alata E, Kaâniche M, Nicomette V. A clustering approach for web vulnerabilities detection. In Pacific Rim international symposium on dependable computing 2011 (pp. 194-203). IEEE Computer Society.
[67] Lounis O, Guermeche SE, Saoudi L, Benaicha SE. A new algorithm for detecting SQL injection attack in web application. In science and information conference (SAI) 2014 (pp.43-51).
[68] Akrout R, Alata E, Kaaniche M, Nicomette V. An automated black box approach for web vulnerability identification and attack scenario generation. Journal of the Brazilian Computer Society. 2014; 20(4):1-16.
[69] Nanda S, Lam LC, Chiueh TC. Dynamic multi-process information flow tracking for web application security. In proceedings of the international conference on Middleware companion 2007. ACM.
[70] Wei K, Muthuprasanna M, Kothari S. Preventing SQL injection attacks in stored procedures. In software engineering conference 2006. IEEE.
[71] Antunes N, Vieira M. SOA-scanner: an integrated tool to detect vulnerabilities in service-based infrastructures. In international conference on services computing 2013 (pp. 280-7). IEEE.
[72] Djuric Z. A black-box testing tool for detecting SQL injection vulnerabilities. In international conference on informatics and applications 2013 (pp. 216-21). IEEE.
[73] Singh AK, Roy S. A network based vulnerability scanner for detecting SQLI attacks in web applications. In international conference on recent advances in information technology 2012 (pp. 585-90). IEEE.
[74] Vernotte A, Dadeau F, Lebeau F, Legeard B, Peureux F, Piat F. Efficient detection of multi-step cross-site scripting vulnerabilities. In international conference on information systems security 2014 (pp. 358-77). Springer, Cham.
[75] Saleh AZ, Rozali NA, Buja AG, Jalil KA, Ali FH, Rahman TF. A method for web application vulnerabilities detection by using boyer-moore string matching algorithm. Procedia Computer Science. 2015; 72:112-21.
[76] Lee M, Lee Y, Yoon H. An enhanced rule-based web scanner based on similarity score. Advances in Electrical and Computer Engineering. 2016; 16(3):9-14.
[77] Liu L, Su G, Xu J, Zhang B, Kang J, Xu S, et a. An inferential metamorphic testing approach to reduce false positives in SQLIV penetration test. In computer software and applications conference 2017 (pp. 675-80). IEEE.
[78] Backes M, Rieck K, Skoruppa M, Stock B, Yamaguchi F. Efficient and flexible discovery of PHP application vulnerabilities. In European symposium on security and privacy 2017 (pp. 334-49). IEEE.
[79] De Meo F, Rocchetto M, Viganò L. Formal analysis of vulnerabilities of web applications based on SQL injection. In international workshop on security and trust management 2016 (pp. 179-95). Springer, Cham.
[80] Huang YW, Yu F, Hang C, Tsai CH, Lee DT, Kuo SY. Securing web application code by static analysis and runtime protection. In proceedings of the international conference on world wide web 2004 (pp. 40-52). ACM.
[81] Huang YW, Tsai CH, Lee DT, Kuo SY. Non-detrimental web application security scanning. In international symposium on software reliability engineering 2004 (pp. 219-30). IEEE.
[82] Huang YW, Huang SK, Lin TP, Tsai CH. Web application security assessment by fault injection and behavior monitoring. In proceedings of the international conference on world wide web 2003 (pp. 148-59). ACM.
[83] Huang YW, Tsai CH, Lin TP, Huang SK, Lee DT, Kuo SY. A testing framework for web application security assessment. Computer Networks. 2005; 48(5):739-61.
[84] Huang YW, Lee DT. Web application security-past, present, and future. In computer security in the 21st century 2005 (pp. 183-227). Springer, Boston, MA.
[85] Kals S, Kirda E, Kruegel C, Jovanovic N. Secubat: a web vulnerability scanner. In proceedings of the international conference on world wide web 2006 (pp. 247-56). ACM.
[86] Viega J, Bloch JT, Kohno Y, McGraw G. ITS4: a static vulnerability scanner for C and C++ code. In

annual conference on computer security applications 2000 (pp. 257-67). IEEE.
[87] Balzarotti D, Cova M, Felmetsger V, Jovanovic N, Kirda E, Kruegel C, et al. Saner: composing static and dynamic analysis to validate sanitization in web applications. In symposium on security and privacy 2008 (pp. 387-401). IEEE.
[88] Tripp O, Pistoia M, Cousot P, Cousot R, Guarnieri S. Andromeda: accurate and scalable security analysis of web applications. In international conference on fundamental approaches to software engineering 2013 (pp. 210-25). Springer, Berlin, Heidelberg.
[89] Galán E, Alcaide A, Orfila A, Blasco J. A multi-agent scanner to detect stored-XSS vulnerabilities. International conference for internet technology and secured transactions 2010 (pp.332-7).
[90] Suto L. Analyzing the effectiveness and coverage of web application security scanners. San Francisco. 2007.
[91] Razzaq A, Latif K, Ahmad HF, Hur A, Anwar Z, Bloodsworth PC. Semantic security against web application attacks. Information Sciences. 2014; 254:19-38.
[92] Mainka C, Somorovsky J, Schwenk J. Penetration testing tool for web services security. In world congress on services 2012 (pp. 163-70). IEEE.
[93] Balduzzi M, Egele M, Kirda E, Balzarotti D, Kruegel C. A solution for the automated detection of clickjacking attacks. In proceedings of the symposium on information, computer and communications security 2010 (pp. 135-44). ACM.
[94] Huyam AA, El-Qawasmeh E. Discovering security vulnerabilities and leaks in ASP. NET websites. In international conference on cyber security, cyber warfare and digital forensic 2012 (pp. 329-33). IEEE.
[95] Eshete B, Villafiorita A, Weldemariam K, Zulkernine M. Confeagle: automated analysis of configuration vulnerabilities in web applications. In international conference on software security and reliability 2013 (pp. 188-97). IEEE.
[96] Vithanage NM, Jeyamohan N. WebGuardia-an integrated penetration testing system to detect web application vulnerabilities. In international conference on wireless communications, signal processing and networking 2016 (pp. 221-7). IEEE.
[97] Laranjeiro N, Vieira M, Madeira H. Protecting database centric web services against SQL/XPath injection attacks. In international conference on database and expert systems applications 2009 (pp. 271-8). Springer, Berlin, Heidelberg.
[98] McAllister S, Kirda E, Kruegel C. Leveraging user interactions for in-depth testing of web applications. In international workshop on recent advances in intrusion detection 2008 (pp. 191-210). Springer, Berlin, Heidelberg.
[99] Doupé A, Cavedon L, Kruegel C, Vigna G. Enemy of the state: a state-aware black-box web vulnerability scanner. In USENIX security symposium 2012.
[100] Pellegrino G, Tschürtz C, Bodden E, Rossow C. JÄk: using dynamic analysis to crawl and test modern web applications. In international workshop on recent advances in intrusion detection 2015 (pp. 295-316). Springer, Cham.
[101] Fonseca J, Vieira M, Madeira H. Vulnerability & attack injection for web applications. In international conference on dependable systems & networks 2009 (pp. 93-102). IEEE.
[102] Fonseca J, Vieira M, Madeira H. Evaluation of web security mechanisms using vulnerability and attack injection. IEEE Transactions on Dependable and Secure Computing. 2014; 11(5):440-53.
[103] Fonseca J, Matarese F. Using vulnerability injection to improve web security. In innovative technologies for dependable OTS-based critical systems 2013 (pp. 145-57). Springer, Milano.
[104] Tung YH, Tseng SS, Shih JF, Shan HL. A cost-effective approach to evaluating security vulnerability scanner. In network operations and management symposium 2013 (pp. 1-3). IEEE.
[105] Dao TB, Shibayama E. Security sensitive data flow coverage criterion for automatic security testing of web applications. In international symposium on engineering secure software and systems 2011 (pp. 101-13). Springer, Berlin, Heidelberg.
[106] Loh PK, Subramanian D. Fuzzy classification metrics for scanner assessment and vulnerability reporting. IEEE Transactions on Information Forensics and Security. 2010; 5(4):613-24.
[107] OWASP T. Application Security Risks 2017.
[108] Deepa G, Thilagam PS, Praseed A, Pais AR. DetLogic: a black-box approach for detecting logic vulnerabilities in web applications. Journal of Network and Computer Applications. 2018; 109:89-109.

Lim Kah Seng received his master degree in Computer Science from University of Technology, Malaysia in 2013. He is currently a doctoral student of Department of Computer Science of University of Technology, Malaysia. He is currently working on the research topic of Authentication Scheme and Software Testing. He is also interested in the study of the Theory of Computation and Artificial Intelligence, and their applications for solving Computation Problem.
Email: lim0709@gmail.com

Norafida Ithnin received her PhD degree in Computation from University of Manchester Institute of Science and Technology. She is currently an Associate Professor of Department of Computer Science of Universiti Teknologi Malaysia. Her research interests are data and computer security, network security, security management, unified threat management, and information hiding & forensics.
Email: afida@utm.my

Syed Zainudeen Mohd Shaid received his PhD in computer science from Universiti Teknologi Malaysia in 2013. He is currently a lecturer at University of Technology, Malaysia. He is also a Certified Penetration Testing Professional (CPTP). He is active in researches of Malware, Network Packet Filtering, and Unmanned Aerial Vehicle.
Email: szainudeen@utm.my