Journal of Theoretical and Applied Information Technology
31st July 2018. Vol.96. No 14
© 2005 – ongoing JATIT & LLS
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

BLACK BOX EVALUATION OF WEB APPLICATION SCANNERS: STANDARDS MAPPING APPROACH

MALIK QASAIMEH[1*], ALA'A SHAMLAWI[2], TARIQ KHAIRALLAH[3]
Princess Sumaya University for Technology, Amman, Jordan
m.qasaimeh@psut.edu.jo[1], Alaa.shamlawi@gmail.com[2], Tariq.khairallah@gmail.com[3]

ABSTRACT

The Secure Development Life Cycle (SDLC) of web applications aims to enhance the quality attributes of released applications. Security is among of the important attributes during the penetration testing phase. Web Application Vulnerability Scanners (WAVS) help the developers to identify existing vulnerabilities that could compromise the security and privacy of data exchanged between the client and web server during the development and deployment phases. WAVS are used during the deployment phase to continuously evaluate the security of web applications by checking for possible vulnerabilities that can threaten the client services. This paper evaluates the effectiveness and accuracy of five WAVSs (Acunetix WVS, Burp Suite, NetSparker, Nessus and OWASP ZAP) to identify possible vulnerabilities of web applications. The selected scanners are among the top ten recommended web vulnerability scanning software for 2017. The method of black box testing was adopted to evaluate the five WAVSs against seven vulnerable web applications. The evaluation is based on different measures such as the vulnerabilities severity level, types of detected vulnerabilities, numbers of false positive vulnerabilities and the accuracy of each scanner. The evaluation is conducted based on an extracted list of vulnerabilities from OWASP and NIST. The accuracy of each scanner was measured based on the identification of true and false positives. The results show that Acunetix and NetSparker had the best accuracy with the lowest rate of false positives.

Keywords—*Web Application Security Scanners, Evaluation, Owasp, Nist, Security Vulnerabilities.*

1. INTRODUCTION

Secure software development aims to activate security development early during the software development life cycle [1]. In the era of internet of things (IoT) it is expected that web applications will be increasingly integrated with sensor-based systems to provide vital services [2] (e.g. smart homes and cities, transportation and logistics and healthcare services) and web-based applications requiring users and sensors to input critical data in order to complete certain transactions to enable smart services [3]. Security measures should be properly configured in order to secure the information exchanged over the web application as well as ensuring the security of the hosting web server. It is said that "the best defense against cyber-attacks is a good offense" [2], which is achieved by testing the web application for any possible vulnerabilities during the verification phase, as described by different Software Security Development Lifecycle (SSDL) institutions, such as Microsoft's Security Development Lifecycle (SDL) [4], Touch Points [5] and Comprehensive Lightweight Application Security Process (CLASP) [6]. Vulnerabilities scanning tools are useful during penetration testing in order to reduce opportunities to exploit potential vulnerabilities early during the SSDL. WAVSs are defined as automated programs that examine web applications for potential security vulnerabilities such as Cross-Site Scripting (XSS), SQL Injection, directory traversal, insecure configurations, and remote command execution vulnerabilities [7, 8]. Most web application vulnerability scanners classify vulnerabilities into four categories: High, Medium, Low, and Informational [9]. Some scanners also consider Critical as a category. These categories are described below:

High: A vulnerability which when exploited allows attackers to take complete control of the web application and server. It allows attackers to access the application's database, modify accounts, and steal sensitive information. XSS and SQL injection are examples of high severity vulnerabilities which should have the utmost priority for fixing if detected by a scanner.

Medium: A vulnerability which when exploited allows attackers to access a logged-in user account to view sensitive content. It allows attackers access to information that helps them exploit other

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

vulnerabilities, or better understand the system so they can refine their attacks. Open redirection is an example of a medium severity vulnerability which allows an attacker to redirect a user to a malicious website. Medium severity vulnerabilities should be addressed at the earliest possible opportunity if detected by a scanner.

Low: A vulnerability that has a minimal impact or cannot be exploited by an attacker. Cookies not marked as HttpOnly is an example of a low severity vulnerability. Marking Cookies as HttpOnly makes the cookie unreadable by client-side scripts and hence provides an additional layer of protection against XSS attacks. Low severity vulnerabilities are worth investigating and fixing if the time and budget allows.

Informational: These are not considered vulnerabilities, rather they are alerts that provide information about the web application. Examples include NTLM Authorization Required and Database Detected (MySQL). No action is needed for these informational alerts.

Black box and white box testing are the main approaches for the security evaluation of web applications. White box testing studies the internal structures of applications, however web applications usually consist of multiple technologies and programming languages, including the client- and server-side languages, therefore the such testing may fail to capture all security flaws and vulnerabilities due to the code complexity [10]. Black box testing, also known as dynamic security evaluation, includes an analysis of the application execution under a certain conditions and inputs (i.e. functions) to identify possible vulnerabilities [11]. This approach, also known as penetration testing, consists of the following phases [10]:

- Crawling and identifications: in this phase the scanner operations include browsing all possible links and web directories. One of the main challenges in this phase is crawling pages that require human input, such as user passwords. The main objective of this phase is to obtain the HTML format of all server replies. By the end of this phase the scanner identifies the entry points that require special input, such as the username and password. The forms and functions such as the GET and POST are also identified in this phase. The scanner should also be able to identify the application structure and functionality to extract information that will be useful in the next phase.
- Parsing and attack: in this phase the scanner generates or uses already existing data from directories expected to match the real data input required by the web application. Fuzz testing is deployed in this phase to generate and submit data of different sizes. Some scanners use malicious input patterns to identify the possible vulnerable response from the web server. The identified actions from the crawling phase along with input filled are sent to the server to observe its reply.
- Analysis: In this phase the scanner analyzes the server response, which is generally influenced by the data submitted. The scanner also classifies the server response and observe the valid ones. The scanners list errors that help in the classification of possible vulnerabilities. For example, if an error was related to XSS, the scanner concludes that XSS vulnerability may exist.

1.1 motivation

The current study is motivated by the diversity of the available commercial and open source tools available for testing and evaluating the security of web applications that can produce relatively quick results. However, the numbers and categorization of detected vulnerabilities differ from one scanner to the other. Some tools will be successful in identifying all "true" vulnerabilities while maintaining a low false positive rate, while others will fail in the detection of true positive vulnerabilities and have a considerably high false positive rate [12]. For example A study conducted by WhiteHat Security revealed that 86% of tested web applications had on average 56% of vulnerabilities per web application, at least one of which was classified as serious. Another study conducted by Symantec executed 1400 scans to find that 76% of websites have at least one serious vulnerability, and 20% of servers consist of critical vulnerabilities [7]. This research is also motivated by the needs for selecting the appropriate WASs for testing and evaluating the web application during the development phases SDLC.

1.2 Contribution

In this paper five state-of-the-art WAVSs (Acunetix WVS, Burp Suite, NetSparker, Nessus and OWASP ZAP) were evaluated to assess their capabilities for detection of web application vulnerabilities. The aim of this paper is to answer the three following research questions:

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

RQ1: what is the number of vulnerabilities that are detected by the selected WASs categorized by severity level?

RQ2: what is the number of vulnerabilities that are detected by the selected WASs categorized by their types?

RQ3: what is the accuracy of each WAS based on the analysis of false positive rates?

The investigated vulnerabilities were extracted from NIST and OWASP standard based on a mapping criteria that analyze the similarities and differences between each standard. The scanners were evaluated against seven intentionally vulnerable applications designed for the purpose of WAVSs evaluation. The evaluated is based on different measures such as the vulnerabilities severity level, types of detected vulnerabilities, numbers of false positive vulnerabilities and the accuracy of each scanner. The accuracy of each scanner was measured based on the identification of true vs. false positive results. The latest versions of the scanners were used to perform the tests and to the best of our knowledge no previous work has been conducted using these versions. The use of seven web applications increased the granularity and diversity of this work, as most of the previous work tested the scanners against one or two web applications at most.

1.3 Road Map

The remainder of this paper is organized as follows: section 2 describes the related work previously performed, section 3 outlines the preliminary investigations, section 4 discusses the methodology adopted for this paper and the evaluation criteria, section 5 presents the results and a discussion of the findings, section 6 presents final remarks and a conclusion is presented in section 7.

2. RELATED WORK

Several research papers have addressed the issue of evaluating the effectiveness of web application security scanners. Some authors performed their study on only open-source or commercial tools, while some combined open-source and commercial tools. Table 1 shows a comparison and analysis of the related work. The table illustrates the evaluated scanners, selected vulnerabilities, testbeds, measures and the publication date for the related studies.

Doupé, Cova and Vigna [13] evaluated ten WAVS: Acunetix, AppScan, Burp, Grendel-Scan, Hailstorm, Milescan, N-Stalker, NTOSpider Paros, w3af and Webinspect. The scanners were selected to cover a wide range of both open source and commercial scanners. The evaluation focused mainly on the capabilities of the selected WAVS against XSS, SQL injection, code injection and broken access controls. Each of the mentioned vulnerabilities were detailed into multiple sub-categories to enable the evaluation process for a total of seventeen different vulnerabilities. The authors chose to develop their own test application called wackopicko, which is fully functional and enables the evaluation process to test the scanner under a realistic conditions. The results show that the running time of N−Stalker is higher than Acunetix, Webinspect and Burp, which provide competitive results of true positive vulnerabilities detection. The authors also note that the crawling process is challenging and needs further investigation to improve the automated identification of vulnerabilities.

Bau, Bursztein, Gupta and Mitchell [14] authors evaluated eight commercial scanners, namely Acunetix WVS, Cenzic HailStrom Pro, HP WebInspect, IBM AppScan, McAfee SECURE, N-Stalker QA Edition, QualysGuard PCI, and Rapid7 NeXpose. They used black box scanning to test the scanners against a custom web application with known vulnerabilities. The vulnerability categories targeted in this study are XSS, SQL Injection, Cross Channel Scripting, Session Management, Cross-Site Request Forgery (CSRF), SSL/Server Configuration, and Information Leakage. The results presented focused on the fact that all scanners were successful in the detection of straightforward XSS and SQL injection vulnerabilities, but failed to detect second order (stored) forms of XSS and SQL injection vulnerabilities.

Parvez, Zavarsky and Khoury [15] evaluated two commercial scanners, Acunetix WVS and IBM AppScan, and one open source scanner, OWASP ZAP. Their evaluation was performed against a custom web application with intentional vulnerabilities. This work focused on only two web application vulnerabilities, XSS and SQL Injection. The analysis revealed that the scanners show some improvement over previous studies in the detection of second order XSS and SQL Injection vulnerabilities.

Makino and V. Klyuev [16] evaluated two open-source scanners, OWASP ZAP and Skipfish. The evaluation was performed against two vulnerable web applications, the Web Application Vulnerability

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Scanner Evaluation Project (WAVSEV) and the Damn Vulnerable Web Application (DVWA). The vulnerabilities investigated were SQL injection, Stored and Reflected XSS, Local and Remote File Inclusion, Command Injection, and CSRF. The results compared the performance of the two WAVSs and found that OWASP ZAP is superior to Skipfish.

Suteva, Zlatkovski and Mileva [17] evaluated six free/open source scanners, namely NetSparker Community Edition, N-Stalker Free 2012, OWASP ZAP, W3AF, Iron WASP and Vega. The evaluation was performed against a vulnerable web application called WackoPicko [13]. The tested vulnerabilities were XSS, SQL Injection, Command Injection, File Inclusion, File Exposure, and several other vulnerabilities. The total number of detected vulnerabilities and the number of false positives were identified. The results showed that NetSparker performed better than the other tested scanners.

A recent study by El Idrissi et,al. [18] was conducted on eleven WAVSs. Five were commercial tools (BurpSuite, Acunetix, Netsparker, AppSpider

and Arachni), and six were open-source tools (Wapiti, SkipFish, W3AF, IronWASP, ZAP and Vega). The scanners were evaluated against the WAVSEV application. The tested vulnerabilities were XSS, SQL Injection, Local and Remote File Inclusion, and Path Traversal. The results show that most scanners have better detection rate for XSS and SQL injection vulnerabilities compared to other types of vulnerabilities. The authors focused on comparing the performance between commercial and open-source tools. The results show that some of the open-source tools like ZAP and Vega have better results than other commercial tools such as AppSpider and Arachni.

Table 1: Comparison And Analysis Of The Related Work

Ref. No	Evaluated scanners	Evaluated vulnerabilities	Testbeds	Measures	date
[13]	Acunetix 1 7 4, AppScan Burp, Grendel-Scan, Hailstorm, Milescan, N-Stalker, NTOSpider, Paros, w3af, Webinspect	Cross-Site Scripting, SQL Injection, Code, Injection, Broken Access Controls	WackoPicko	Accuracy, execution times and Threat scores	2010
[14]	Acunetix, Cenzic, WebInspect Rational AppScan, McAfee SECURE, N-Stalker QualysGuard PCI, NeXpose	Cross-Site Scripting SQL Injection, Cross Channel Scripting, Session Management, Cross-Site Request Forgery	Drupal, phpBB, Wordpress	Scanner footprint, Vulnerability detection, false positive	2010
[15]	Acunetix, Rational AppScan, ZAP	SQL Injection, Stored XSS	WackoPicko, Scan-bed	Detection rate for SQLI and XSS	2015
[16]	ZAP, Skipfish	SQL injection, cross site scripting, file injection, Command execution, request forger	DVWA WAVSEP	Detection rate, false positive rate	2015
[17]	NetSparker, N-Stalker, OWASP, ZAP, W3Af, Iron WASP, Vega	SQLI, XSS, Session ID Injections, File exposure Parameter manipulation Directory traversal, logic flow, forceful browsing, weak passwords	WackoPicko	False positive rate	2013
[18]	BurpSuite, Acunetix, Wapiti, SkipFish, Netsparker, W3AF, AppSpider, IronWASP, Arachni, ZAP, Vega	Common vulnerabilities such as XSS, SQLI and uncommon vulnerabilities such as Missing Function Level Access Control with a total of 10 vulnerabilities	WAVSEP	Precision, Recall, and F-measure	2017

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

3. PRELIMINARY INVESTIGATIONS

WAVSs can be found as open source (completely free of charge) or commercial tools with varying costs. The commercial tools usually offer a free trial version for customers and evaluators, however they lack support and features available in non-trial versions. Selecting one scanner over another requires considering several aspects that can be summarized by the following guidelines [8]:

- The scanner should support the protocol and authentication scheme used by the web application.
- The scanner should support the main types of input delivery methods and be able to detect vulnerabilities in the web application with a low false positive rate.
- The scanner should be within the technical abilities of the person who is going to use it.
- The scanner should be stable, and regularly updated with the latest security updates to cover the ongoing vulnerabilities being discovered.
- The scanner should be within the budget limit of the project.

This paper aims to evaluate the scanner's capabilities to detect the true vulnerabilities in web applications based on multiple vulnerable web applications. The five WAVSs selected for this paper were reported among the top WAVSs tools available on the market for the year 2017, as reported by Concise Courses [19]. Table 2 illustrates some characteristics of the evaluated scanners such as the vendor, version, license and the operating platforms. In this paper the scanners were tested against seven vulnerable web applications to increase the granularity and diversity of the detected vulnerabilities. The evaluation assessed the capabilities of each scanner to detect a set of eight vulnerabilities extracted from the NIST and OWASP Standards. The following is a description of the selected WAVSs tools.

Table 2. General Characteristics Of Evaluated Scanners

Scanner	Vendor	Version	License	Platform
Acunetix WVS	Acunetix	11.0	Commercial	Windows
BurpSuite	PortSwigger	1.7.30	Commercial	Linux, MAC, Windows
NetSparker	NetSparker Ltd.	4.7.1	Commercial	Windows
Nessus	Tenable	Cloud based	Cloud Based	Linux, MAC, Windows
OWASP ZAP	OWASP	2.7.0	Open Source	Linux, MAC, Windows

Acunetix WVS is a leading web vulnerability scanner used to automatically check web applications for vulnerabilities. Acunetix WVS is used to discover if a website is secure by crawling and analyzing the web application to find if there are SQL injections, XSS, Host Header Injection and over 3000 other web vulnerabilities [20]. Acunetix is a commercial tool which works on Windows operating systems only. Older versions of Acunetix were evaluated in [14], [15], and [18].

BurpSuite is an integrated platform for security testing of web applications. It has an advanced web application scanner for automating the detection of numerous types of vulnerabilities. It is available in a free version with limited features and in a commercial version with maximum features [21]. BurpSuite works on Linux, MAC OS and Windows operating systems. An older version of BurpSuite was among the scanners evaluated in [18].

Nessus is a vulnerability scanner with one of the largest knowledge bases of security vulnerabilities and hundreds of plugins which can be activated for detailed customized scans. Nessus can detect security vulnerabilities in the operating system of targeted hosts, patches, services. It also demonstrate the ability to propose solutions, which can mitigate these security vulnerabilities [22]. Nessus is a commercial tool which works on Linux, MAC OS, and Windows operating systems. For this work a cloud-based version of Nessus vulnerability scanner was evaluated. Nessus vulnerability scanner was not

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

evaluated in any of the previous works mentioned above.

NetSparker is a web application security scanner which is designed to discover and audit web application vulnerabilities such as SQL Injection and XSS possibilities. NetSparker is used by cybersecurity space professional and is considered by many to be easy to use [23]. NetSparker is a commercial tool that works on Windows Operating Systems only. Older versions of NetSparker were evaluated in [17] and [18].***OWASP Zed Attack Proxy (ZAP)*** is an open source integrated penetration testing tool for finding vulnerabilities in web applications. It is designed to be used by people with a wide range of security experiences such as developers and functional testers who are new to penetration testing as well as being a useful addition to an experienced pen testers toolbox [24]. OWASP ZAP is a free tool which works on Linux, MAC OS, and Windows operating systems. Older versions of OWASP ZAP were evaluated in [15], [16], [17] and [18].

The selected vulnerable applications are designed specially to allow web developers, security auditors and penetration testers to practice their knowledge and testing skills, without any legal concerns. Usually the evaluation of WAVS against a real and live web application may pose legal concerns, especially if the web scan is not authorized by the web application's owner, and the live scan could also cause disruptions to main features and services of the tested web application. Table 3 lists the seven web applications and the main features of each.

Table 3. Vulnerable Web Applications

Index	Web Address	Features
W1	*altoromutual.com*	An online banking web application created by IBM to test web application scanners, written in C#.NET with a ported JAVA version.
W2	*crackme.cenzic.com/kelev /view/home.php*	An online banking web application created by Trustwave to test automated WAVSs, written in PHP language.
W3	*testaspnet.vulnweb.com*	Anews blog website created by Acunetix as a testing application for scanners, written in asp.net language.
W4	*testphp.vulnweb.com*	An online shopping web application created by Acunetix as a testing application for scanners, written in PHP language.
W5	*zero.webappsecurity.com*	An online banking web application created by Hewlett-Packard (HP) to test web vulnerability scanners.
W6	*www.webscantest.com*	A web application created by NTOSpider (now maintained by rapid7) with several services to test web application scanners, written in PHP language.
W7	*testhtml5.vulnweb.com*	An online social networking application created by Acunetix to test web application scanners, written in HTML5 language

4. EVALUATION DESIGN

4.1 Evaluation Methodology

The evaluation in this paper is based on the approach of black box testing. That main benefit of this approach that it is provides a similar scenario to real, common attacks. Security testing based on back box approach is used to evaluate web application scanners with no prior knowledge of the web applications' internal structure. In addition, it is technology independent, and hence is suitable for testing and evaluating the efficiency of web application vulnerability scanners regardless of the underlying technology of the web application. The main components of a web application security scanner consist of three modules: the crawling, fuzzing, and analysis and reporting modules [18]. In black box security testing the scan starts by crawling the web application to find all internal links to identify the main entry points that require special input. Then, the fuzz testing is preformed to generate random input for each entry point identified in the crawling step. Analysis is then performed on the results obtained from the fuzzing process to generate detailed reports of the vulnerabilities at the end of the testing process [16].

The selected scanners were deployed to scan each of the seven web applications in order to identify the possible vulnerabilities. To generate default results the scanners were operated on default profile mode, where no customization or tuning has been provided to the scanners. For each web application a report was generated by the scanner that lists the discovered vulnerabilities. Since each

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

scanner has its own format and presentation, a manual organization and classification was also conducted at the end of the scanning process to compare the resulted vulnerabilities between the selected scanners. Figure 1 illustrates the methodology phases conducted in this study.

4.2 Evaluation Criteria

The evaluation criteria were developed based two standards proposed by leading software security organizations and researchers to aid the formulation of baselines for evaluating web application vulnerability scanning tools. The first standard is the special NIST publication 500-269 [8] proposed by the National Institute of Standards and Technology (NIST) to provide guidelines to measure the usability and effectiveness of the WAVSs. It lists some specifications for mandatory and optional features of WAVSs. The second standard is the Open Web Application Security Project (OWASP), which provides a recent updated taxonomy for the top ten most critical security risks that threaten the security of modern web applications [25]. The NIST Standard [8] has stated a list of fourteen vulnerabilities that should be analyzed and identified for a web application. This standard also provides a baseline for the evaluation of web application security scanners and has been used by many recent works [26-28]. The fourteen vulnerabilities defined in the NIST standard are listed in table 4. OWASP 2017 released a set of 10 vulnerabilities, as shown in table 4, however some of the OWASP vulnerabilities are feature-dependent and could not be identified by the WAVSs. For example, A4, A9 and A10 are only detected by source code analysis (i.e. white box testing).

The vulnerabilities that form the basis of the evaluation were extracted based on mapping criteria between the NIST and OWASP. The mapping aims to identify the overlapping vulnerabilities between the two standards. Finally, eight vulnerabilities were chosen to form the baseline for evaluation in this work. The mapping criteria focused on the correlation between the two taxonomies, since some of the vulnerabilities listed in NIST standard can be mapped to OWASP vulnerabilities. For example, SQL Injection OS, Command Injection and XML Injection are mapped to the Injection vulnerability from the OWASP standard. It was also noticed that some vulnerabilities are feature dependent and others may require source code analysis to be detected. Such vulnerabilities are also eliminated since they are beyond the purview of black box vulnerabilities scanning. Table 5 lists the eight chosen vulnerabilities based on the mapping criteria.

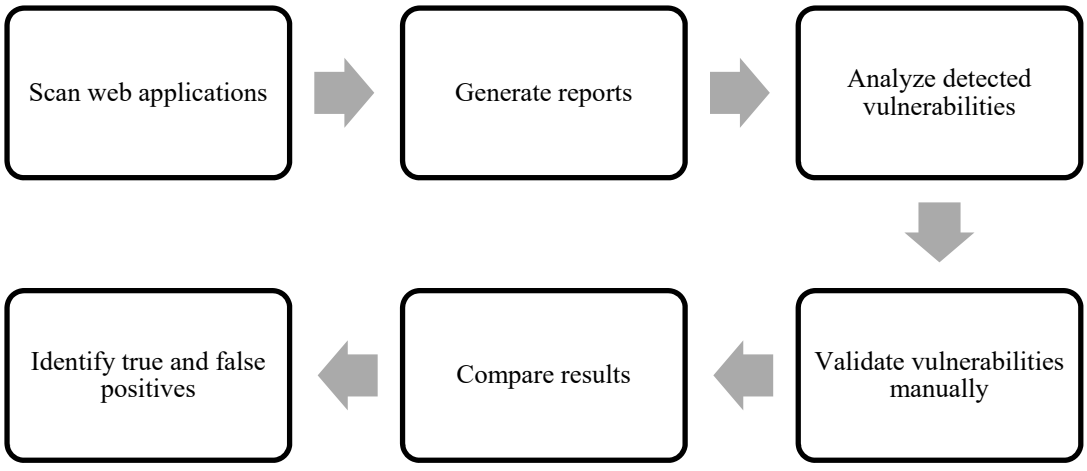

Figure 1. Evaluation Methodology For The Wavss

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 4. NIST And OWASP Vulnerabilities

NIST Index	NIST vulnerabilities	OWASP Index	OWASP vulnerabilities
N1	Cross Site Scripting (XSS)	A1	Injection
N2	SQL Injection	A2	Broken Authentication
N3	OS Command Injection	A3	Sensitive Data Exposure
N4	XML Injection	A4	XML External Entities (XXE)
N5	HTTP Response Splitting	A5	Broken Access Control
N6	Malicious File Inclusion	A6	Security Misconfiguration
N7	Insecure Direct Object Reference	A7	Cross Site Scripting (XSS)
N8	Cross Site Request Forgery (CSRF)	A8	Insecure Deserialization
N9	Information Leakage		
N10	Improper Error Handling	A9	Using Components with Known Vulnerabilities
N11	Weak Authentication	A10	Insufficient Logging and Monitoring.
N12	Session Fixation		
N13	Insecure Communication		
N14	Unrestricted URL Access		

Table 5. Investigated Vulnerabilities Mapping Results

NIST			OWASP		Mapping results		
V	M	D	V	D	V	D	Investigated Vulnerabilities
N1	A7	✓	A1	✓	V1	✓	Cross Site Scripting (XSS)
N2	A1	✓	A2	✓	V2	✓	Injection
N3	A1	✓	A3	✓	V3	✓	Broken Authentication
N4	A1	✓	A4	×	V4	✓	Security Misconfiguration
N5	-	✓	A5	✓	V5	✓	Sensitive Data Exposure
N6	-	✓	A6	✓	V6	✓	Malicious File Inclusion
N7	A9	×	A7	✓	V7	✓	Cross Site Request Forgery (CSRF)
N8	A7	✓	A8	✓	V8	✓	Insecure Communication
N9	A3	✓	A9	×			
N10	A6	✓	A10	×			
N11	A2	✓					
N12	A6	✓					
N13	A2	✓					
N14	A5	✓					

(V for Vulnerability, M for Mapping to OWASP, and D for Detectability by a scanner)

5. DETAILED VULNERABILITIES ANALYSIS

The subsections below present and discuss the obtained results based on the following measures: the number of vulnerabilities detected by each scanner, the classification of the vulnerabilities based on their severity level, the classification of detected vulnerabilities by their types, the number of false positive vulnerabilities and finally the accuracy of each scanner.

5.1 Vulnerabilities Severity Level

This section is aimed to answer the first research question (**RQ1**). Table 6 shows the total number of vulnerabilities that have detected by the five

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

scanners for the vulnerable web applications. It illustrates that the number of detected vulnerabilities varies to some extent from one scanner to another. For example, in W1 the highest number of vulnerabilities was found by OWASP Zap and the lowest number was found by Nessus. No generalized pattern could be concluded since the vulnerable applications originally differ in the number of existing vulnerabilities. The total number of discovered vulnerabilities is not an accurate measure and further investigation into the severity levels of the discovered vulnerabilities is needed. Vulnerabilities have a certain severity level which reflects their impact on the web application if successfully exploited by attackers. Figure 2 shows the number of vulnerabilities classified into informational, low, medium and high severity levels.

Table 6. Total Number Of Vulnerabilities Detected By Each Scanner Per Web Application

Vulnerable Applications	Evaluated Scanners				
	Acunetix	BurpSuite	NetSparker	Nessus	ZAP
W1	88	93	78	33	225
W2	33	94	28	70	154
W3	57	28	48	23	88
W4	206	140	86	104	265
W5	144	49	109	24	23
W6	267	178	159	62	258
W7	34	33	27	24	26

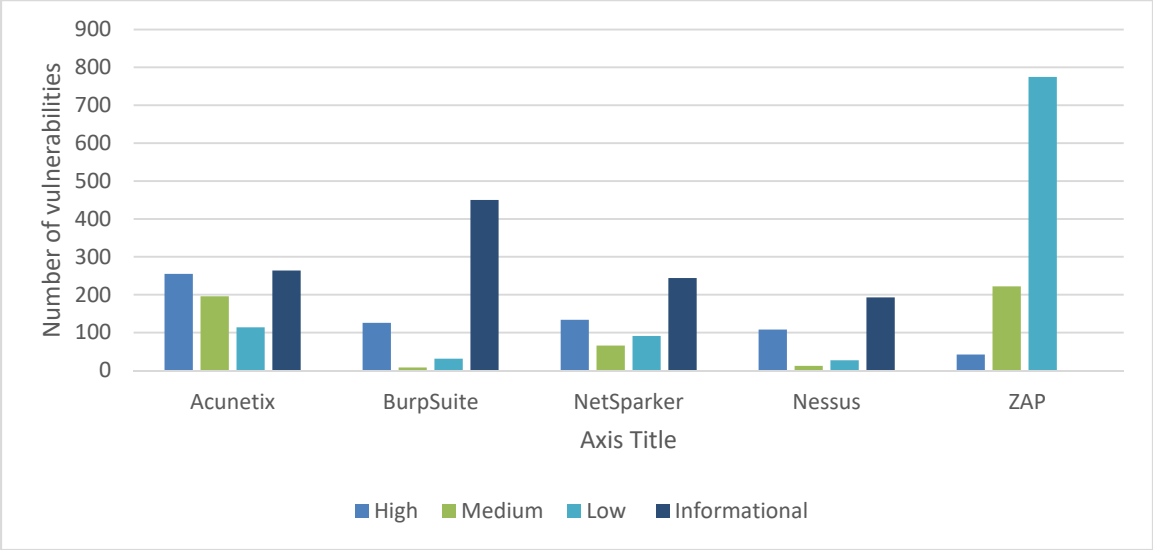

Figure 2. Total Number Of Detected Vulnerabilities Classified By Severity Level

It can be seen from figure 2 that Acunetix was able to detect the largest number of high severity vulnerabilities, followed by NetSparker, however there is a big gap between the two. BurpSuite was third, followed by Nessus, and finally OWASP ZAP detected a very low number of high severity level vulnerabilities. ZAP classifies the vulnerabilities as high, medium, and low without considering the category of informational vulnerabilities. The results show that ZAP detected the highest number of medium and low severity level vulnerabilities, however most of these vulnerabilities are not in fact "true", as explained in the following sections.

5.2 Types of Detected Vulnerabilities

This section is aimed to answer the second research question (**RQ2**). The total number of detected vulnerabilities classified by their type was calculated from the results obtained by the five scanners. As shown in table 7, Acunetix detected the highest number of vulnerabilities related to XSS,

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Injection, Sensitive data exposure, broken authentication, malicious file inclusion and CSRF, however it was less successful in identifying security misconfiguration vulnerabilities. The OWASP ZAP had the lowest number of detected vulnerabilities for most of the types listed, however the number of detected vulnerabilities related to security misconfiguration is very high for the OWASP ZAP, which is will verified in the next section for the possibilities of false positive.

Table 7. Total Number Of Detected Vulnerabilities Classified By Type

Vulnerabilities		Evaluated Scanners				
		Acunetix	BurpSuite	NetSparker	Nessus	ZAP
V1	Cross Site Scripting (XSS)	115	66	60	54	26
V2	Injection	133	55	115	49	13
V3	Broken Authentication	2	0	0	0	0
V4	Security Misconfiguration	165	179	140	100	743
V5	Sensitive Data Exposure	143	90	113	29	11
V6	Malicious File Inclusion	2	1	1	1	11
V7	Cross Site Request Forgery (CSRF)	45	29	15	5	8
V8	Insecure Communication	22	27	25	15	0

Figure 3 shows the detection percentage for each type of vulnerability. Security misconfiguration had the highest percentage of 51%. OWASP states that security misconfiguration as the most commonly seen systems vulnerability that can be caused by numerous actions, such as incomplete or ad hoc configurations, open cloud storage, misconfiguration, HTTP headers and verbose error messages containing sensitive information [25].

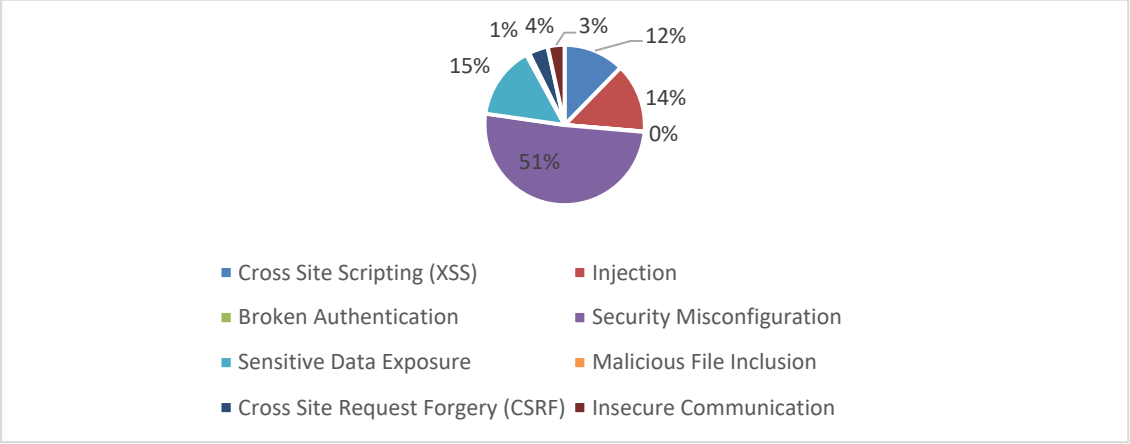

Figure 3. Percentage Of Vulnerability Detection

5.3 Vulnerabilities Validation

This section is aimed to answer the third research question (**RQ3).** The black box evaluation of web applications could generate false positive results, due to the fact that this approach is conducted with no awareness of the internal structure and the authenticating credentials of the web application. Hence, the scanner might not be able to access all the necessary information required to correctly identify the vulnerabilities. The scanner in this type of test mostly relies on the information obtained from service banners and signature matching checks, which often lead to false detection of vulnerabilities. Hence, it is required to verify the obtained results manually by exploiting and checking its validity. This process consumes a considerable amount of time and effort. There are two types of measures to validate the propagated results by the web scanners: false negative, which is defined as a vulnerability

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

missed by the scanner, and hence not detected or reported; and a false positive, which is defined as a vulnerability that the scanner mistakenly reports, however after manual investigation it was found to be false vulnerability [29]. In this context we focus on validating the reported vulnerabilities against the false positive measure.

The results obtained from the scanners were evaluated manually to verify the likelihood of false positive vulnerabilities. NetSparker reported an SQL injection vulnerability from the website www.testphp.vulnweb.com, specifically the link: http://testphp.vulnweb.com/listproducts.php?artist. The payload identified by the scanner was verified manually and it was found that the SQL injection was unsuccessful and an error was reported. Another example is an XSS vulnerability reported by Acunetix in the login page of the website www.altoromutual.com, specifically in the link: http://www.altoromutual.com/bank/login.aspx. The payload used by the scanner was verified manually, however the XSS attack was unsuccessful and an error was reported. Using a similar procedure we verified the obtained vulnerabilities. Figure 4 illustrates the number of false positives for each scanner in comparison with the total number of detected vulnerabilities.

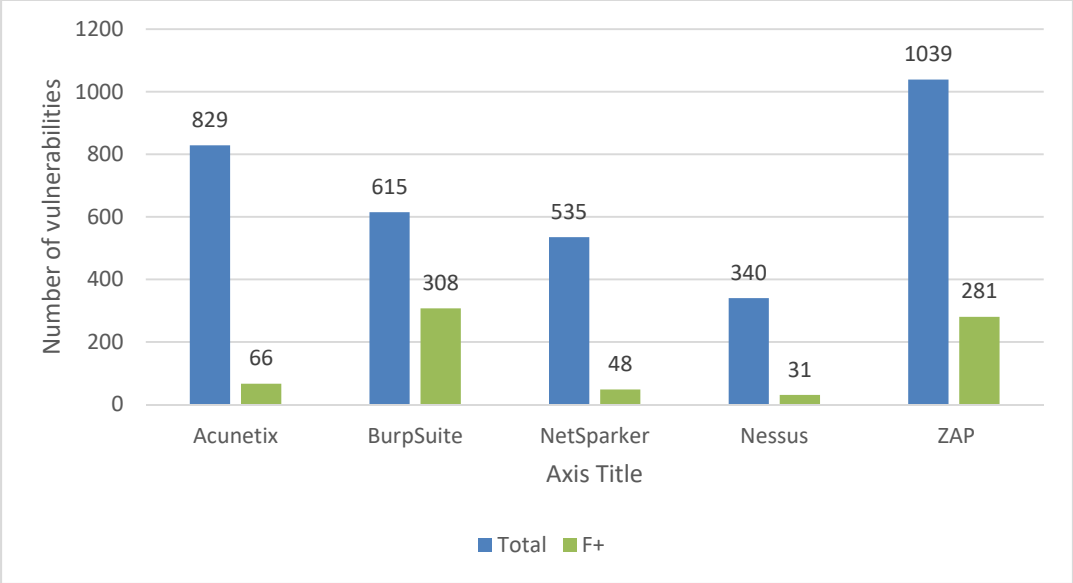

Figure 4. False Positive Count

False positives vulnerabilities is a useful measure of web scanner accuracy. In this context the accuracy for each scanner was obtained after computing the ratio of correctly identified vulnerabilities the "True Positive" to the total number of detected vulnerabilities. As indicated in table 8, the accuracy scores for Acunetix, NetSparker, and Nessus were higher than 90%, while those for Burp Suite and OWASP ZAP were significantly lower. To improve the accuracy of scanners a built-in exploitation functionality is supported by some web scanners like NetSparker to automatically validate the detected vulnerability. If the vulnerability was successfully exploited, it is reported as a "Certain" vulnerability. Vulnerabilities that cannot be exploited automatically are reported as "Possible or probable". This improves the scanner accuracy and shortens the time and effort required by manual verification.

Table 8. Calculated Accuracy For Each Scanner

Vulnerability Count	Acunetix	BurpSuite	NetSparker	Nessus	ZAP
True positive	763	307	487	309	758
False positive	66	308	48	31	281
Total	829	615	535	340	1039
Accuracy	91%	50%	91%	90%	73%

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

6. FINAL REMARK

Previous studies assessed WAVSs against either a limited number of vulnerable web applications or a specific set of vulnerabilities. For example, experiments in [13], [17] and [18] evaluated the scanners against one vulnerable web application. In [16] the scanners were evaluated against two vulnerable web applications. Also, in [15] the authors focused on the detection of only a specific type of vulnerability, such as XSS and SQL Injection. The experiment conducted in this paper provides an updated investigations for the cuurent version of WAVSs based on relatively wider testbeds. The obtained result aligned with the previous finding and also shows an improvement for the current versions of WAVs in terms of accuracy and low rate of false positive for both Acunetix and NetSparker.

7. CONCLUSION AND FUTURE WORK

The SDLC of web applications consists of many activities that aim to enhance the overall applications' robustness against different types of attack. Black box testing is an important activity used during SDLC to enhance the security of web applications. Developers and penetration testers utilize WAVSs to dynamically scan web applications and investigate any possible vulnerability before it can threaten client services. Currently, many WAVSs are proposed and they have different capabilities in detecting the true positive vulnerabilities. In this paper five WAVSs were evaluated by scanning seven intentionally vulnerable web applications. The selected WAVs are among the top ten scanners for the year of 2017. A mapping approach was developed to extract a list of vulnerabilities based on NIST and OWASP standards. The scanners were evaluated using different measures, such as the vulnerabilities severity level, types of detected vulnerabilities, numbers of false positive vulnerabilities and the accuracy of each scanner. The number of vulnerabilities found by the ZAP scanner was higher than its counterparts, however manual validation revealed that fewer false positives were found in Acunetix and NetSparker, which means they have higher and more accurate "true" vulnerabilities identification. The results of this work is limited to the set of accessible WASs scanner at the time of the experiment and larger varieties of both commercial and open source scanner need to be investigated to obtain a generalized conclusion. As a part of our future work, we plan to develop plug-in algorithms that enable the exploitation of the vulnerabilities to reduce the time and effort needed for manual validation of results.

REFERENCES:

[1] P. Salini and S. Kanmani, "Effectiveness and performance analysis of model-oriented security requirements engineering to elicit security requirements: a systematic solution for developing secure software systems," *International Journal of Information Security,* vol. 15, pp. 319-334, June 01 2016.

[2] Z. B. Babovic, J. Protic, and V. Milutinovic, "Web Performance Evaluation for Internet of Things Applications," *IEEE Access,* vol. 4, pp. 6974-6992, 2016.

[3] N. Teodoro and C. Serrão, "Web application security: Improving critical web-based applications quality through in-depth security analysis," in *International Conference on Information Society*, 2011, pp. 457-462.

[4] M. Howard and S. Lipner, *The Security Development Lifecycle*: Microsoft Press, 2006.

[5] B. D. Win, R. Scandariato, K. Buyens, J. Gr, #233, goire, *et al.*, "On the secure software development process: CLASP, SDL and Touchpoints compared," *Information and Software Technology* vol. 51, pp. 1152-1171, 2009.

[6] "OWASP, Comprehensive, lightweight application security process," ed, 2006.

[7] D. Pałka, M. Zachara, and K. Wójcik, "Evolutionary Scanner of Web Application Vulnerabilities," in *Computer Networks*, Cham, 2016, pp. 384-396.

[8] P. E. Black, E. Fong, V. Okun, and R. Gaucher, "Software assurance tools: Web application security scanner functional specification version 1.0," *Special Publication,* pp. 500-269, 2008.

[9] R. Lepofsky, "Web Application Vulnerabilities and Countermeasures," in *The Manager's Guide to Web Application Security: A Concise Guide to the Weaker Side of the Web*, ed Berkeley, CA: Apress, 2014, pp. 47-79.

[10] N. Khoury, P. Zavarsky, D. Lindskog, and R. Ruhl, "An Analysis of Black-Box Web Application Security Scanners against Stored SQL Injection," in *IEEE Third International*

Conference on Privacy, Security, Risk and Trust 2011, pp. 1095-1101.

[11] N. I. Daud, K. A. A. Bakar, and M. S. M. Hasan, "A case study on web application vulnerability scanning tools," in *Science and Information Conference*, 2014, pp. 595-600.

[12] F. van der Loo, "Comparison of penetration testing tools for web applications," Master's thesis, Computer Sceince University of Radboud, Netherlands, 2011.

[13] A. Doupé, M. Cova, and G. Vigna, "Why Johnny Can't Pentest: An Analysis of Black-Box Web Vulnerability Scanners," in *Detection of Intrusions and Malware, and Vulnerability Assessment*, Berlin, Heidelberg, 2010, pp. 111-131.

[14] J. Bau, E. Bursztein, D. Gupta, and J. Mitchell, "State of the Art: Automated Black-Box Web Application Vulnerability Testing," in *2010 IEEE Symposium on Security and Privacy*, 2010, pp. 332-345.

[15] M. Parvez, P. Zavarsky, and N. Khoury, "Analysis of effectiveness of black-box web application scanners in detection of stored SQL injection and stored XSS vulnerabilities," in *10th International Conference for Internet Technology and Secured Transactions (ICITST)*, 2015, pp. 186-191.

[16] Y. Makino and V. Klyuev, "Evaluation of web vulnerability scanners," in *8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, 2015, pp. 399-402.

[17] N. Suteva, D. Zlatkovski, and A. Mileva, "Evaluation and testing of several free/open source web vulnerability scanners," presented at the 10th Conference for Informatics and Information Technology, Bitola, Macedonia, 2013.

[18] S. El Idrissi, N. Berbiche, F. Guerouate, and M. Sbihi, "Performance Evaluation of Web Application Security Scanners for Prevention and Protection against Vulnerabilities," *International Journal of Applied Engineering Research,* vol. 12, pp. 11068-11076, 2017.

[19] Concise-courses. (January, 2018). *Web Vulnerability Scanners: Recommended Vulnerability Scanning Software For 2017*. Available: https://www.concise-courses.com/hacking-tools/web-vulnerability-scanners/

[20] A. WVS. (Jan 2018). *Acunetix WVS*. Available: https://www.acunetix.com

[21] B. Suite. (January, 2018). *Burp Suite*. Available: https://portswigger.net/burp

[22] N. Professional. *Nessus Professional* Available: www.tenable.com/products/nessus/nessus-professional

[23] NetSparker. (Janauray, 2018). *NetSparker*. Available: https://www.netsparker.com/

[24] O. ZAP. (January, 2018). *OWASP ZAP*. Available: https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project

[25] OWASP. *OWASP Top 10 - 2017, The Ten Most Critical Web Application Security Risks*. Available: https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf

[26] C. Zhu, "Experimental study of vulnerabilities in a web application," Master's Thesis Computer Science, Aalto University, 2017.

[27] A. Broström, "Integrating Automated Security Testing in the Agile Development Process: Earlier Vulnerability Detection in an Environment with High Security Demands," Master's Thesis in Computer Science, Computer Science and Engineering Royal Institute of Technology, 2015.

[28] Y. Takhma, T. Rachid, H. Harroud, M. R. Abid, and N. Assem, "Third-party source code compliance using early static code analysis," in *International Conference on Collaboration Technologies and Systems (CTS)* Atlanta, GA, USA, 2015, pp. 132-139.

[29] S. Patil, N. Marathe, and P. Padiya, "Design of efficient web vulnerability scanner," in *2016 International Conference on Inventive Computation Technologies (ICICT)*, 2016, pp. 1-6.