The 8th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications
24-26 September 2015, Warsaw, Poland

Evaluation of Web Vulnerability Scanners

Yuma Makino[1] , Vitaly Klyuev[1]
[1] University of Aizu,
Tsuruga, Ikki-Machi, Aizu-Wakamatsu, Fukushima, Japan, 965-8580
e-mail: yumarisa52141144@gmail.com; vkluev@u-aizu.ac.jp
http://web-ext.u-aizu.ac.jp/labs/is-se/

***Abstract* — In recent yeas a lot of web applications have been released in the world. At the same time, cyber attacks against web application vulnerabilities have also increased. In such a situation, it is necessary to make web applications more secure. However checking all web vulnerabilities by hand is very difficult and time-consuming. Therefore, we need a web application vulnerability scanner. In this work, we evaluate two open source vulnerability scanners OWASP Zed Attack Proxy (OWASP ZAP) and Skipfish using vulnerable web applications Damn Vulnerable Web Application (DVWA) and The Web Application Vulnerability Scan- ner Evaluation Project (WAVSEP).**

Keywords* — *security; web application; scanner;

I. Introduction

Web applications are integral part of our life nowadays. At the same time, the attacks using web application vulnerabilities and the damage caused by them are increasing. However, it is difficult to develop completely secure Web applications. Furthermore, checking all web application vulnerabilities by hand is difficult. These are time-consuming, error-prone and costly. Therefore we consider that a good automated tool to detect Web application vulnerabilities is needed. To solve the above problems, it must have the functions such as a Web crawler, a Web scraping tool and fuzzing tool at least.

A web Crawler is an Internet bot that systematically browses the World Wide Web, typically for the purpose of Web indexing [1]. Web scraping is a computer software technique of extracting information from websites.

Fuzzing is a software testing technique, often automated or semi-automated, that involves providing invalid, unexpected, or random data to the inputs of a computer program [2].

There are such vulnerability detection tools on the software market but their accuracy of detection is not perfect. And the most of them are not easy to use for novices [3]. The easy to use feature is very important because novices make Web applications including more vulnerabilities than an expert. Now we can believe that the evaluation of a web application scanner is needed because a perfect scanner does not exist.

Several studies have been done on evaluation of the web application scanners [4]–[6]. However, evaluation of web application vulnerability scanners such as OWASP Zed Attack Proxy (OWASP ZAP) [7] and Skipfish [8] have never been studied so far.

In this work we evaluate these two scanners. These are open source, automated and multi platform (Windows, Linux, OS X) software. We think these features are very important because the web application scanner must be useful for many people to develop more secure web applications.

The remainder of this thesis is structured as follows: Section 2 introduces basic concepts of the web vulnerability scanners and web application vulnerabilities. Section 3 explains our experimental environment. To put more concretely, we explain the target scanner for evaluation, vulnerable web applications for evaluating the scanner and a method of evaluation. Section 4 shows experimental results and analyze it. Finally, Section 5 summarizes results obtained in our work.

II. Background

A. Basic Concepts

Almost all Web vulnerability scanners consist of three main components: a crawling component (crawling function), an attacker component (fuzzing function), and an analysis component (scraping function) [1]. There are two main approaches [9] to testing applications for finding it's vulnerabilities:

White box testing: consists of the analysis of the source code of the web applications. This can be done manually or by using code analysis tools. The problem is that the perfect source code analysis may be difficult and cannot find all security flaws because of the complexity of the code.

Black box testing: includes the analyses of the execution of the application to search for vulnerabilities. In this approach, also known as penetration testing, the scanner does not know the internals of the web application and it uses fuzzing techniques over the web HTTP requests [2].

We can consider that the black box testing is better for testing web applications for the above features. OWASP ZAP and Skipfish adopting black box testing. This is one of the reasons why we use OWASP ZAP and Skipfish.

978-1-4673-8361-5/15/$31.00 ©2015 IEEE

B. Vulnerabilities

We must consider what type of vulnerability the scanner should cover, so we referred to The OWASP Top Ten 2013 [10]. The OWASP Top Ten 2013 offers a list of the most critical application vulnerabilities, including different types of injection, broken authentication and session management, cross-site scripting cross-site request forgery, etc. We characterize major vulnerabilities below.

1) Cross Site Scripting: Cross Site Scripting (XSS, sometimes also abbreviated as CSS but it is confusable with cascading style sheets) refers to a range of attacks in which the attacker injects malicious JavaScript code into a web application [1]

2) SQL injection: SQL injection attacks are based on injecting strings into database queries that alter their intended use. This can occur if a web application does not properly filter (sanitize) user input [1].

3) File inclusion: File inclusion vulnerability allows an attacker to include a file, usually through a script on the web server. The vulnerability occurs due to the use of user-supplied input without proper validation. It has two types of inclusion, Remote File Inclusion (RFI) and Local File Inclusion (LFI).

III. Experimental Environment

A. Vulnerable web application

For evaluating scanners, vulnerable test applications are needed. We use following two vulnerable web applications which used in our experiments. They run on the OWASP Broken Web Applications Project virtual machine [11], it has many intentionally vulnerable applications.

1) DVWA: Damn Vulnerable Web Application (DVWA) is a PHP/MySQL web application that is vulnerable. Its main goals are to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and aid teachers/students to teach/learn web application security in a class room environment [12]. It is GNU General Public License version 3. We use it's version 1.8.

2) WAVSEP: The Web Application Vulnerability Scanner Evaluation Project (WAVSEP) is a vulnerable web application designed to help assessing the features, quality and accuracy of web application vulnerability scanners. This evaluation platform contains a collection of unique vulnerable web pages that can be used to test the various properties of web application scanners [13]. This application is GNU General Public License version 3. We use it's version 1.2. This is written in Java JSP. What is remarkable about this application, it has false positive test cases. It includes codes which seems to vulnerabilities.

B. Tested Web Vulnerabilities Scanners

As we mentioned earlier, we worked with OWASP ZAP and Skipfish. The scanners were run on a machine with a Intel(R) Core(TM) i5-2320 3.00GHz CPU, 4GB of RAM, and Windows 7 Home Premium.

1) OWASP ZAP: The OWASP Zed Attack Proxy (ZAP) is an easy to use integrated penetration testing tool for finding vulnerabilities in web applications. It is designed to be used by people with a wide range of security experience and as such is ideal for developers and functional testers who are new to penetration testing as well as being a useful addition to an experienced pen testers toolbox. [14]

2) Skipfish: Skipfish is an active web application security reconnaissance tool. It prepares an interactive sitemap for the targeted site by carrying out a recursive crawl and dictionary-based probes. The resulting map is then annotated with the output from a number of active security checks. The final report generated by the tool is meant to serve as a foundation for professional web application security assessments. [8]

Table I lists characteristics of two scanners used in this study and their general characteristics. As we mentioned earlier, all they are open source, automated and multi platform software.

Table I. General characteristics of the evaluated scanners

	OWASP ZAP	skipfish
Company/ Creator	OWASP	Google
Version	2.3.1	2.10b
Licence	ASF2	ASF2
Operating System	Windows Linux OS X	Windows (Cygwin) Linux OS X FreeBSD

Table II. Supporting input vectors by the evaluated scanners

	OWASP ZAP	Skipfish
GET	Yes	Yes
POST	Yes	Yes
COOKIE	Yes	Yes
HEADER	Yes	Yes
XML	Yes	
XML Attributes	Yes	
JSON	Yes	

Their supporting input vectors are given in Table 2.

In this paper, we check following vulnerabilities summarized in Table III. This table characterize vulnerabilities of both vulnerable web applications. Both scanners cover all audit features.

Table III. AUDIT FEATURES OF EVALUATED SCANNERS

	OWASP ZAP	Skipfish
SQLi[1]	Yes	Yes
BSQLi[2]	Yes	Yes
RXSS[3]	Yes	Yes
PXSS[4]	Yes	Yes
LFI[5]	Yes	Yes
RFI[6]	Yes	Yes
CMDExec[7]	Yes	Yes
CSRF[8]	Yes	Yes

[1] SQL injection
[2] Brind SQL injection
[3] Reflected cross site scripting
[4] Persistent (stored) cross site scripting
[5] Local file injection
[6] Remote file injection
[7] Command execution
[8] Cross site request forgery

Table IV. THE TOTAL NUMBER OF INTENTIONAL VULNERABILITIES IN WAVSEP

vulnerability	WAVSEP
RXSS	73
SQLi	105
LFI	325
RFI	108

Table V. THE TOTAL NUMBER OF INTENTIONAL VULNERABILITIES IN DVWA

vulnerability	DVWA
RXSS	1
PXSS	1
SQLi	2
BSQLi	1
CSRF	1
LFI	1
CMDExec	1

Table V and IV show the total number of intentional vulnerabilities in DVWA and WAVSEP.

C. Methodology

We tested mentioned above two scanners against given above two vulnerable web applications for detecting vulnerabilities. We decided whether detected problems are true vulnerabilities, by checking the alert information. When the alerted vulnerability is executable or has error output that includes information for an attack, we counted it as a vulnerability.

We are not concerned with the alerts for non-intentional vulnerabilities such as "X-Content Type header missing" for simplify because they are often alerted from web pages whose vulnerability is not intentional.

We added links (index-active.jsp and index-passive.jsp) on wavesep's index page because both scanners could not crawl without the links.

IV. RESULTS AND DISCUSSIONS

The number of found vulnerabilities classified (by risk) according to scanner's severities are given in Table VI and Table VII.

Table VI. NUMBER OF VULNERABILITIES OF THE EVALUATED SCANNERS DETECTED IN WAVSEP

	OWASP ZAP	Skipfish
High Vulnerabilites	0	0
Medium Vulnerabilities	34	11
Low Vulnerabilities	2535	14
Informational Vulnerabilities	625	353
SUMMARY	3194	378

Table VII. NUMBER OF VULNERABILITIES OF THE EVALUATED SCANNERS DETECTED IN DVWA

	OWASP ZAP	Skipfish
High Vulnerabilites	13	7
Medium Vulnerabilities	10	2
Low Vulnerabilities	83	3
Informational Vulnerabilities	1	1
SUMMARY	107	13

These reports are useful for efficient debugging web application. However these severities include non-target informations like X-Content-Type-Options header missing. As mentioned in Section 3.3, we ignored these alert for simplify because they are alerted from not intentionally vulnerable web pages a lot.

We look up in terms of precision of scanners. Table VIII summarizes the precision of two scanners evaluated for WAVSEP taking the result from table VI.

Table VIII. PRECISION OF TWO SCANNERS ACCORDING TO WAVSEP

	OWASP ZAP	skipfish
RXXS	100%	8.2%
SQLi	100%	9.5%
LFI	43.2%	1.0%
RFI	0.0%	0.0%

The following formula is used to calculate precision.

$$Precisionrate \quad = \quad \frac{N_a}{N_a + N_b} \times 100$$

$$N_a : Numberofvulnerabilitiesdetectedcorrectly.$$

$$N_b : Numberofvulnerabilitiesdetectedincorrectly.$$

We see from Table VIII that OWASP ZAP detected significantly large number of vulnerabilities compared to

skipfish drastically. We also see that neither of them could not detect RFI vulnerabilities in spite of both scanner support RFI detection. .

Let us focus on the false positive rate of the scanners against WAVSEP. WAVSEP has "False Positive Test Cases". It includes codes which seems to vulnerabilities. Table IX shows the result of "False Positive Test Cases".

Table IX. FALSE POSITIVE RATE WITH WAVSEP'S FALSE POSITIVE TEST CASE

	OWASP ZAP	skipfish
RXXS	0%	100%
SQLi	0%	100%
LFI	0%	100%
RFI	0%	100%

The false positive rate is calculated as follows.

$$Falsepsitiverate \quad = \quad \frac{N_b}{N_b + N_c} \times 100$$

$$N_b : Numberofvulnerabilitiesdetectedincorrectly.$$

$$N_c : Numberofnotvulnerabilitiesdtectedcorrectoly.$$

According to Table IX, OWASP ZAP could distinguish all "False Positive Test Cases" correctly. On the other hand, Skipfish was deceived thoroughly. Compared with previous work [6], false positive rate of them are inverted. In work [6], it tested Skipfish and Paros (Paros is OWASP ZAP's antecedents) against DVWA and etc. Judging from the above, we can consider that OWASP ZAP are improved from Paros.

Now, we will focus on the results with DVWA. DVWA has fewer vulnerabilities than WAVSEP, so we give the result as the number of true positive. Table X shows comparison the number of true positive in DVWA.

Table X. THE NUMBER OF TRUE POSITIVE VULNERABILITIES IN DVWA

	This study		Study by Nakai et. al.	
	OWASP ZAP	Skipfish	Paros	Skipfish
RXSS	1	1	2	1
PXSS	-	-		
SQLi	2	1	1	1
BSQLi	-	-		
CSRF	-	1	-	8
LFI	1	-	-	-
CMDExec	-	-	-	-

Empty cells indicate that the scanner did not discover the vulnerability. Although it looks that the scanner could detect a few vulnerabilities. This result is similar to the result of previous work [6].

V. CONCLUSION

In this work, we evaluated OWASP ZAP and Skipfish vulnerability scanners. We found that OWASP ZAP is superior over Skipfish as far as in this experimental situation. However, we should note that both of them are not perfect yet especially with detection of the RFI vulnerability. Furthermore, we realize that we need a vulnerable web application without unintentional vulnerabilities for more accurate evaluation.

REFERENCES

[1] S. Kals, E. Kirda, C. Kruegel, and N. Jovanovic, "Secubat: a web vulnerability scanner," in *Proceedings of the 15th international conference on World Wide Web*. ACM, 2006, pp. 247–256.

[2] M. Sutton, A. Greene, and P. Amini, *Fuzzing: Brute Force Vulnerability Discovery*. Addison-Wesley Professional, 2007. [Online]. Available: http://www.amazon.com/Fuzzing-Brute-Force-Vulnerability-Discovery/dp/0321446119%3FSubscriptionId%3D0JYN1NVW651KCA56C102%26tag%3Dtechkie-20%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26creativeASIN%3D0321446119

[3] "Introduction of vulnerability testing in web site," http://www.ipa.go.jp/files/000035859.pdf, IPA(Information-Technology Promotion Agency, Japan), 2013.

[4] N. Suteva, D. Zlatkovski, and A. Mileva, "Evaluation and testing of several free/open source web vulnerability scanners," 2013.

[5] J. Fonseca, M. Vieira, and H. Madeira, "Testing and comparing web vulnerability scanning tools for sql injection and xss attacks," in *Dependable Computing, 2007. PRDC 2007. 13th Pacific Rim International Symposium on*. IEEE, 2007, pp. 365–372.

[6] R. Nakai, T. Tsuchiya, and T. Kikuno, "Evaluation of automated testing tools for web application vulnerability detection," *IEICE technical report . DC, Dependable Computing*, vol. 110, no. 229, pp. 19–23, oct 2010. [Online]. Available: http://ci.nii.ac.jp/naid/110008106158/

[7] S. Bennetts, "Owasp zed attack proxy," in *AppSec USA 2013*. Owasp, 2013.

[8] M. Zalewski, N. Heinen, and S. Roschke, "Skipfish-web application security scanner," 2011.

[9] C. Ghezzi, M. Jazayeri, and D. Mandrioli, *Fundamentals of software engineering*. Prentice Hall PTR, 2002.

[10] D. Wichers, "Owasp top-10 2013," 2010.

[11] C. Willis, "Owasp broken web applications project," 2010.

[12] "Damn vulnerable web application (dvwa)," http://www.dvwa.co.uk/, accessed: 2014-12-14.

[13] "The web application vulnerability scanner evaluation project (wavsep)," https://code.google.com/p/wavsep/, accessed: 2014-12-14.

[14] "The owasp zed attack proxy (zap)," https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project, accessed: 2014-12-14.