Evaluation of Open Source Web Application Vulnerability Scanners

[1] Hilmi S. Abdullah

[1] IT Department, Amedi Technical Institute, Duhok Polytechnic University, Kurdistan Region - Iraq

ABSTRACT

Nowadays, web applications are essential part of our lives. Web applications are used by people for information gathering, communication, e-commerce and variety of other activities. Since they contain valuable and sensitive information, the attacks against them have increased in order to find vulnerabilities and steal information. For this reason, it is essential to check web application vulnerabilities to ensure that it is secure. However, checking the vulnerabilities manually is a tedious and time-consuming job. Therefore, there is an exigent need for web application vulnerability scanners. In this study, we evaluate two open source web application vulnerability scanners Paros and OWASP Zed Attack Proxy (OWASP ZAP) by testing them against two vulnerable web applications buggy web application (bWAPP) and Damn Vulnerable Web Application (DVWA).

Keywords: Web Application Security, Open Web Application Security Project (OWASP), Vulnerability Scanner, Penetration Testing.

1. Introduction

In recent years, web application hacking has increased dramatically. This is because the importance of the web applications in our daily lives. Ensuring the security of web applications and finding their vulnerabilities is crucial as the majority of information and services on the internet are provided through web applications, such as e-commerce, social networking and e-governance [1, 2]. There are several vulnerabilities which can result in data breach and shutdown of the web applications. Open Web Application Security Project (OWASP) lists the top 10 web applications security risks and vulnerabilities [3]. However, finding the vulnerabilities manually is costly, time-consuming and difficult. Therefore, there is a need for vulnerability scanners [4].

Web applications vulnerability scanners perform the checking process automatically. They generally have three components; first, the web crawler for gathering website data; second, the attacker component which sends random and invalid input to the web application; and the last component is the analyzer which analyzes the returned data, detects the vulnerabilities and generates the report. There are various available scanners both commercial and free [4, 5]. In this study we test and evaluate two open source web application vulnerability scanners OWASP Zed Attack Proxy(OWSAP ZAP) and Paros.

Section II provides the basic concepts of web application vulnerabilities according to OWASP top 10 list as well as penetration testing methods in addition to an overview of how web application vulnerability scanners work. Section III explains the experiment environment including tested scanners and web applications in addition to the methodology. Section IV

Academic Journal of Nawroz University
(AJNU) Volume 9, No 1 (2020).
Regular research paper : Published 17 Feb 2020
Corresponding author's e-mail : hilmi.salih@gmail.com

presents the results, analysis and assessment of the tested scanners. Finally, we provide the conclusion in section V.

2. Background

2.1 Web Application Vulnerabilities

There are variety of vulnerabilities that threatens web applications. OWASP lists the top 10 application security risks, following is the list for 2017 [3].

- *Injection:* Injection happens when untrusted data is included in a query or a command to trick the interpreter and get unauthorized access to data [3].
- *Broken Authentication:* Attackers might steal identities of other users and compromise passwords when authentication and session management isn't perfect [3].
- *Sensitive Data Exposure:* Attackers might get access to sensitive data such as financial data if they are not fully protected [3].
- *XML External Entities (XXE):* These attacks will result in denial of service or revealing confidential data by uploading malicious XML files which are parsed by poorly configured XML parsers [3].
- *Broken Access Control:* This vulnerability occurs when authenticated users' permissions are not restricted properly, which enables the attackers to access sensitive data and change other users' data and permissions [3].
- *Security Misconfiguration:* This security breach happens when the default configurations are insecure. Continuous upgrade of servers, frameworks and applications is necessary [3].
- *Cross-Site Scripting (XSS):* The XSS vulnerability enables the attackers to inject and run scripts on the victim's browser [3].
- *Insecure Deserialization:* This vulnerability occurs when the web application doesn't secure the deserialization process properly which can be used to perform injection and other attacks [3].
- *Using Components with Known Vulnerabilities:* Using vulnerable components such as frameworks and libraries might enable different attacks that lead to data loss or server takeover [3].
- *Insufficient Logging and Monitoring:* Logging of the events and persistent monitoring to discover any suspicious events is necessary [3].

2.2 Testing Methods

Penetration testing of the web applications is necessary prior to their launching and during their operation. The test can be performed either automatically or manually [6].

- *Automated Testing:* Is a technique of using software tools to scan web application pages to discover vulnerabilities and generate reports at the end of the test. There are several tools used for automated testing such as OWASP ZAP, Burp Suite, Paros, W3af, etc. [5]
- *Manual Testing:* Sometimes automated testing is not enough to assess the vulnerabilities of the web application and there is a need for human intervention to perform the attacks as in social engineering. [5]

2.3 Web Application Vulnerability Scanners

Generally, web application vulnerability scanners contain crawler, attacker and analyzer components [7].

Firstly, the crawler component is responsible for finding the reachable pages of the scanned web application as well as identifying its entry points such as HTML forms input and the parameters of GET or POST, etc.

Secondly, the attacker component is responsible for analyzing the data obtained by the crawler, then generates values for each vulnerability type and sends form data to the web server to obtain the response.

Finally, the analyzer component interprets and

doi : 10.25007/ajnu.v9n1a532

analyzes the response from the web server. Then it determines if a specific attack was successful. Next, it generates the report of the scan [7, 8]. Fig. 1 [8] shows the main stages of penetration testing.

Fig. 1. Main stages of penetration testing

3. EXPERIMENT ENVIRONMENT

To perform the experiment we used the operating system Kali Linux, web application vulnerability scanners and vulnerable web applications.

3.1 Vulnerable Web Applications

There are several available vulnerable web applications developed intentionally for the purpose of learning and testing. For this experiment, the following two vulnerable web applications have been selected which are available online and can be downloaded for free.

- *DVWA:* Damn Vulnerable Web Application (DVWA) is a web application used by security professionals and web developers for testing and teaching purposes. It is GNU General Public License version 3; version 1.10 is used for this experiment. DVWA is PHP/MySQL web application and contains many vulnerabilities. It has four different security levels; the lowest level has been selected for this experiment [4].
- *bWAPP:* buggy Web Application (bWAPP) is a free and open source vulnerable web application designed for testing and teaching purposes. It is written in PHP and uses MySQL database; version 2.2 is used for this experiment. It has three different security levels low, medium and high. For this experiment, the lowest one is used [9].

3.2 Operating System

To perform the test, Kali Linux is used as a virtual machine in Oracle VirtualBox. Kali Linux is an open source operating system based on Debian which is used for penetration testing and digital forensics. For this experiment version 3.28.0 is used. It contains several built-in penetration tools such as OWASP ZAP, Paros, etc. [10, 11].

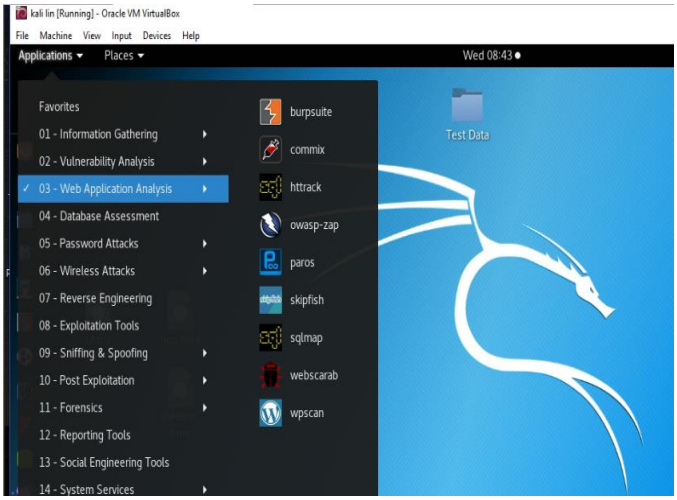

Fig. 2. Kali Linux

3.3 Open Source Web Application Vulnerability Scanners

There are various open source tools used for web application penetration testing. For this experiment OWASP ZAP and Paros are used, both are built-in Kali Linux as shown in Fig. 2.

- *OWASP ZAP:* Is an open source web application penetration testing tool. It is used by web developers and security professionals to scan and find the vulnerabilities of web applications. For this experiment version 2.7.0 is used [12, 13].
- *Paros:* Is a web application vulnerability scanner

doi : 10.25007/ajnu.v9n1a532

which is open source and cross-platform. Paros Version 3.2.13 is used for this experiment [14, 15]. Table I lists the general characteristics of the tested open source scanners.

Table 1: General characteristics of the tested scanners

	OWASP ZAP	**Paros**
License	ASF2	GPL
Version	2.7.0	3.2.13
Scanning Method	Automated	Automated
Status	Up to date	Outdated
Operating System	Cross-Platform	Cross-Platform

3.4 Methodology

We ran an automated scan for both scanners OWASP ZAP and Paros against the vulnerable web applications DVWA and bWAPP which were installed on the local host. For both scanners the default configuration were selected. Later on, when the scanning process finished, a report was generated about the detected vulnerabilities in the tested vulnerable web applications.

4. Result and discussion

To evaluate OWASP ZAP and Paros, we compared the number and types of the detected vulnerabilities by each scanner. In addition, their features and ease of use is assessed.

At the end of the scanning process we analyzed the reports generated by the tested scanners against the tested vulnerable web applications. Both scanners categorize the vulnerabilities per risk levels high, medium and low. Table II presents the summary of the detected vulnerabilities in the tested vulnerable web applications.

Table 2: Detected vulnerabilities summary

Tool / Risk Level	**ZAP (DVWA)**	**PAROS (DVWA)**	**ZAP (BWAPP)**	**PAROS (BWAPP)**
High	7	2	7	2
Medium	2	3	6	5
Low	4	0	8	3

Obviously there are several high risk (critical) vulnerabilities detected in the tested web applications by both scanners. The majority of these vulnerabilities were Injection and Cross Site Scripting (XSS) which are in the OWASP top 10 list for 2017.

OWASP ZAP was able to detect the following critical vulnerabilities:

- SQL Injection
- Cross Site Scripting (Reflected)
- Cross Site Scripting (Persistent)
- Remote OS Command Injection
- Path Traversal
- External Redirect
- Remote File Inclusion

On the other side, Paros was able to detect the following critical vulnerabilities only:

- SQL Injection
- SQL Injection Fingerprinting

Overall, the performance of OWASP ZAP was better than Paros as can be seen in Fig. 3 which makes a comparison between the tested scanners based on the number of detected vulnerabilities.

doi : 10.25007/ajnu.v9n1a532

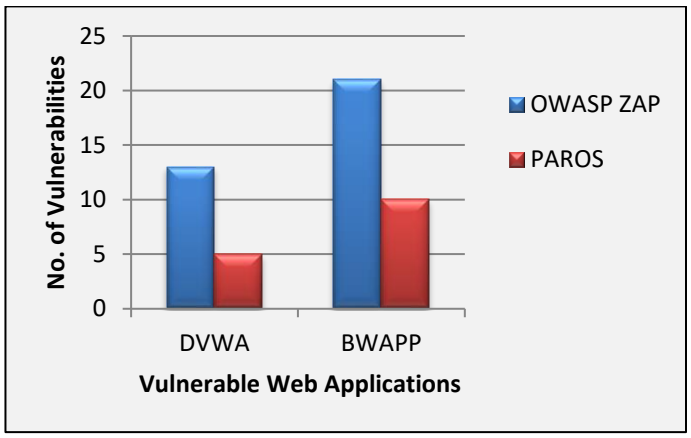

Fig. 3. Comparison of the detected vulnerabilities

Finally, considering the features of both scanners for the assessment, OWASP ZAP has more features compared to Paros as well as it is more user-friendly and it is regularly updated unlike Paros which is outdated. Furthermore, OWASP ZAP detected various types of critical vulnerabilities while Paros was able to detect only SQL injection.

5. CONCLUSION

Web applications are used by people on everyday life for various services like e-governance, shopping and communication. However, the importance of web applications attracts the attackers for different purposes. Therefore, there is an increasing need to secure these web applications by penetration testing using vulnerability scanners. In this study we tested two open source scanners and compared their performance and features. Obviously, OWASP ZAP performed better, as it detected more vulnerabilities than Paros with a more diverse range of vulnerabilities. In addition, OWASP ZAP has more features and is regularly updated which makes it superior to Paros.

6. References

1. Al-Khurafi, O. & Al-Ahmad. M. (2015). Survey of Web Application Vulnerability Attacks. 4th International Conference on Advanced Computer Science Applications and Technologies (ACSAT), Kuala Lumpur, 2015 (pp. 154-158).
2. Farah T., Shojol M., Hassan M. & Alam D. (2016). Assessment of vulnerabilities of web applications of Bangladesh: A case study of XSS & CSRF. Sixth International Conference on Digital Information and Communication Technology and its Applications (DICTAP), Konya, 2016, (pp. 74-78).
3. OWASP. (2018).The Open Web Application Security Project. Retrieved from https://www.owasp.org/
4. Makino Y. & Klyuev V. (2015). Evaluation of web vulnerability scanners. Intelligent Data Acquisition and Advanced Computing Systems:Technology and Applications (IDAACS), IEEE 8th InternationalConference vol. 1. IEEE, 2015, (pp. 399–402).
5. Nagpure S. & Kurkure S. (2017). Vulnerability Assessment and Penetration Testing of Web Application. International Conference on Computing, Communication, Control and Automation (ICCUBEA), Pune, 2017, (pp. 1-6).
6. Srinivasan S. & Sangwan R. (2017). Web App Security: A Comparison and Categorization of Testing Frameworks. IEEE Software, vol. 34, no. 1, IEEE, 2017, (pp. 99-102).
7. Suteva N., Zlatkovski D. & Mileva A. (2013). Evaluation and testing of several free/open source web vulnerability scanners, 10th Conference for Informatics and Information Technology, Bitola, Macedonia, 2013.
8. Jiménez R. (2016). Pentesting on web applications using ethical - hacking. IEEE 36th Central American and Panama Convention (CONCAPAN XXXVI), San Jose, 2016, (pp. 1-6).
9. BWAPP. (2018). A buggy Web Application. Retrieved from http://itsecgames.com/
10. Gaddam R. & Nandhini M. (2017). An analysis of various snort based techniques to detect and prevent intrusions in networks proposal with code refactoring snort tool in Kali Linux environment. International Conference on Inventive Communication and Computational Technologies (ICICCT), Coimbatore, 2017 (pp. 10-15).
11. Denis M., Zena C. & Hayajneh T. (2016). Penetration testing: Concepts, attack methods, and defense strategies. IEEE Long Island Systems, Applications and Technology Conference (LISAT), Farmingdale, NY, 2016 (pp. 1-6).

doi : 10.25007/ajnu.v9n1a532

12. Daud N.,Bakar K. & Hasan M. (2014) .A case study on web application vulnerability scanning tools. Science and Information Conference, London, 2014 (pp. 595-600).
13. OWASP ZAP. (2018). Zed Attack Proxy Project - OWASP. Retrieved from https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project
14. Engebretson P. (2013). The Basics of Hacking and Penetration Testing. Waltham, MA: Syngress.
15. Goel J., Asghar M., Kumar V. & Pandey S. (2016). Ensemble based approach to increase vulnerability assessment and penetration testing accuracy. International Conference on Innovation and Challenges in Cyber Security (ICICCS-INBUSH), Noida, 2016 (pp. 330-335).

doi : 10.25007/ajnu.v9n1a532