Poster: Evaluating Black-Box Testing Tools

Esteban Alejandro Armas Vega, Ana Lucila Sandoval Orozco, Luis Javier García Villalba*

Group of Analysis, Security and Systems (GASS)
Department of Software Engineering and Artificial Intelligence (DISIA)
Faculty of Computer Science and Engineering, Office 431, Universidad Complutense de Madrid (UCM)
Calle Profesor José García Santesmases, 9, Ciudad Universitaria, 28040 Madrid, Spain
Email: esarmas@ucm.es, {asandoval, javiergv}@fdi.ucm.es

***Abstract*—The benchmarking of tools for dynamic analysis of vulnerabilities in web applications is something that is done periodically. Unfortunately, the vast majority of these evaluations are made by software enthusiasts who publish their results on blogs or on non-academic websites and always with the same evaluation methodology. Similarly, academics who have carried out this type of analysis from a scientific approach, the majority, make their analysis within the same methodology as well the empirical authors. This paper is based on the interest of finding answers to questions that many users of this type of tools have been asking over the years, such as, to know if the tool truly test and evaluate every vulnerability that it ensures do, or if the tool, really, deliver a real report of all the vulnerabilities tested and exploited. This kind of questions have also motivated previous work but without real answers.**

1. Introduction

Web applications manage, receive, send and store much of information which, in most cases, is personal and confidential, therefore, security within these web applications should be a number one priority. Use of dynamic analysis tools to improve the security within web application reduces time and effort and allows to focus on more complex safety tasks [1]. Such tools, for those who are not familiar with this kind of sofware are not trivial to set up correctly [2].

The aim of this work is to evaluate most of the generated data over the interaction between the black-box tools and the analyzed web applications. This is achieved by gathering carried out attacks and also the alerts and vulnerabilities showed in the report. All this obtained information was contrasted and compared.

2. Black-box Testing Tool

The black-box analysis type, analyzes the security of a web application without any interaction with the source code or with a previous knowledge of its structure. This approach allows to analyze the web application in a simulated production environment [3], the same environment as the attacker will see. The Figure 1 shown the common interaction of a black-box testing tool.

- The **first phase of Black-box analysis** is the passive stage in which the pentest tool analyze the components (pages, forms and links) of a web application to find all the inputs within the web site, this phase is also known as crawling phase.
- The **second phase of Black-box analysis** is the active stage in which the pentest tool generates malicious inputs that send to the target application through the found inputs in the passive phase and analyze the response from the web application to determine if there are or not a vulnerability. The generated report consist of the potential vulnerabilities within the application that can be used by an attacker.

3. Experiments and Results

The system was run on two computers, one attacker and another as a server. Both are connected by a physical network and a network router. The Table 2 shows the

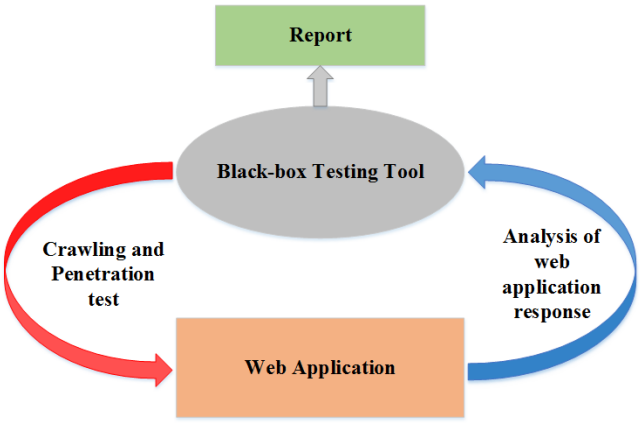

Figure 1: Black-box Testing Interaction

TABLE 1. RESULTS OF ANALYZE DVWA

Vulnerability	Wacko Picko	IDS	Acunetix WVS	IDS	OWASP ZAP	IDS	HP Web Inspect	IDS	Arachni
Reflected Cross Site Scripting	1	X	1	X	2	X	5	X	1
Persistend Cross Site Scripting	1		4		1		-		2
SQL Injection	2	X	4	X	1	X	3	X	-
Blind SQL Injection	1	X	3	-	-	X	-	-	-
Cross Site Request Forgery	1	-	-	-	-	-	-	-	-
Local File Inclusion	1	X	-	-	-	-	-	-	1
Command-line Injection	1	X	2	X	1	X	1	X	-
Path ravesal	-	X	1	X	1	X	11	X	1
DoS	-	X	1	X	-	X	-	-	-
Remote File Inclusion	-	-	-	X	1	-	-	X	1

TABLE 2. TEST ENVIRONMENT FEATURES

	Macbook Pro	Mac Mini	
	Attacker	IDS	Web Server
SO	Windows 7.1 Profesional	Ubuntu Server 14.04 LTS x86	Ubuntu Server 14.04 LTS x86
Hardware	8GB Ram, Intel Core i5 2.7GHz	1GB Ram, 1 CPU, (Virtual)	1GB Ram, 1 CPU, (Virtual)
Software	Acunetix WVS, OWASP ZAP, HP Webinspect, Arachni	Snort IDS, MySQL, Barnyard2, Snorby	Apache 2, MySQL, DVWA, Darkstat

characteristics of each one of the components used in the experiments of this paper.

Each tool has been use once and separately. At the end of the analysis with each tool both, web server and SNORT, have been reset. In each single tests the reports generated by SNORT alerts are compared. All of the vulnerabilities of the evaluated web application and the report generated by each of the pentesting tools have been taken into account.

The DVWA application analysis with Acunetix WVS not reported the vulnerability Local File Inclusion. This vulnerability is present in DVWA and according to alerts SNORT was a vulnerability exploited during the analysis. But Acunetix WVS do not included it in its final report. With the OWASP ZAP tool do not differ from those obtained in previous studies [2]. According to the SNORT reports OWASP ZAP do not tests all of OWASP top 10 OWASP [4] vulnerabilities. The analysis with HP WebInspect was the longest, although it was the tool that found less vulnerabilities. On the positive side, the number of false positives in the report were the lowest. The analysis with Arachni, identified 5 of total vulnerabilities with a low number of false positives. SNORT identified attacks which Arachni did not show it in the report.

A summary of the results obtained from the analysis of DVWA with those tools can be shown in the Table 1.

4. Conclusion

The approach proposed in this paper allowed to obtain details on the results that in previous work were not considered. The evaluated tools showed deficiencies to identifying vulnerabilities in the tested web applications.

Most of the tools that were used in this paper demonstrated that attacks were made and confirmed by SNORT, but in the final report of the tool was not considered as vulnerable, despite its existence in the application. In addition becames clear that there was not carried out attacks in search of at least OWASP TOP 10 2013. This can be considered as a disadvantage in these tools, mainly because its intention is to cover at least the vulnerabilities more important and documented and that according to reports obtained in this work was not done.

Acknowledgements

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 700326. Website: http://ramses2020.eu

References

[1] P. Baral. Web Application Scanners: A Review of Related Articles. *IEEE Potentials*, 30(2):10–14, March 2011.

[2] Y. Makino and V. Klyuev. Evaluation of Web Vulnerability Scanners. In *Proceedings of the IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, volume 1, pages 399–402, Warsaw, PL, September 2015.

[3] A. Sagala and E. Manurung. Testing and Comparing Result Scanning Using Web Vulnerability Scanner. *Advanced Science Letters*, 21(11):3458–3462, November 2015.

[4] The Open Web Application Security Project OWASP. OWASP Top 10 - 2013 The Ten Most Critical Web Application Security Risks. Release, The Open Web Application Security Project OWASP, June 2013.