WASec, a Web Security Scanner for Web Applications

Diaddeen Sidoon[1], Anwar Alhenshiri[2] and Inass Husien[3]
[1]Internet Systems Department, Faculty of Information Technology, Misurata University
[2]Information Systems Department, Faculty of Information Technology, Misurata University
[3]Documentation and Information Center, Faculty of Information Technology, Misurata University
diaddeenss, alhenshiri, e.ismail @ it.misuratau.edu.ly

Submission data: 12.6.2020, Acceptance data:11.7.2020, Date of electronic publishing 1.8.2020
https://www.misuratau.edu.ly/journal/sci/upload/file/R-12 64-ISSUE-10%20PAGES%2048-56.pdf

Abstract: Web application security scanners are automated testing and scanning programs that examine web applications for potential security vulnerabilities. WASec (Web Application Security) was created as a security scanner that uses black-box testing to scan web applications for error-based and time-based SQL injection along with Reflected Cross-site Scripting vulnerabilities and report them. WAsec was tested against web applications of known vulnerabilities. The testing process, although preliminary, has showed promising results. The design and implementation of this tool was intended to tackle the current security problems in Libyan websites.

Keywords: Web, security, vulnerability, testing, tool, software, application, scripting, website.

Introduction

Web application security is as important as the security of banks and governmental offices. it is concerned with the security of web applications, websites, and web services [3]. Web applications of different kinds have been a target of direct attacks by hackers who seek either the end users' information or the corporate behind the website [1].

Security in web applications is approached in many different ways, when designing, building, testing and even after finishing the construction of the application. Web Application Firewalls (WAFs). For example, is one of the most popular security solutions that are implemented in web applications. Their main job is to prevent attackers from exploiting the system's vulnerabilities, by monitoring, filtering and blocking HTTP traffic (i.e. traffic through the HTTP port, port 80) [6].

Another mechanism of security is Intrusion Prevention Systems (IPS)/Intrusion Detection Systems (IDS). This form of security works on intrusion detection and stopping any detected incidents of this kind. IPS and IDS systems constantly watch the network, identify possible incidents, stop the incidents, log information about them, and report them to security administrators [4].

One of the interesting methods of security is Black-box Testing. It is a technique to test web applications based on their external visible behavior and functionality.[5] The tester does not have access to the application's source code and only works by observing the application's output in response to a specific input. Some tools that use black-box testing include web application security scanners, vulnerabilities scanners, and penetration testing software [2].

On the other hand, White-box Testing tests the web application's internal structure using statistical code analysis tools. The tester must have access to the web application's source code,

and must have experience in the programming languages used in the application being tested. Static source code analyzer is one of the tools that use white-box testing [2].

In addition to the aforementioned techniques, there are other security testing approaches such as Gray-box Testing. This technique falls between the white-box and black-box testing techniques. The tester knows some parts of the source code used for building the application. Moreover, Fuzzing is a technique used for input testing. It involves invalid, unexpected, and random data inputs to a web application [2]. Finally, Password Cracking is used to test password strength. Brute-Force is a commonly used technique that tries to guess repeatedly for passwords and check them against any available cryptographic hash of the password [5].

According to the Open Web Application Security Project (OWASP), the most critical web applications vulnerabilities are Cross-Site Scripting (XSS) and Sql Injections [4]. Most Web applications are susceptible to some kind of vulnerability. Last year's research shows that over 80% of Web applications were vulnerable to Cross Site Scripting (XSS) and over 25% were vulnerable to SQL injection flaws [7].

Cross Site Scripting and Injection Flaws are the two most spread vulnerabilities as indicated earlier. These are also the most easily detected by scanners. Therefore, most of the available scanners search for them, more or less successfully. However, most of vulnerability scanners look only for particular subsets of these vulnerabilities. SQL injection is a typical injection flaw that is most often sought. Reflected XSS is the easiest XSS type to detect [8].

The research in this paper focuses on those two vulnerabilities to allow web developers in Libya to benefit from the designed system. The current status of web design lacks security measures and makes web applications very vulnerable.

Related Work

Several studies proposed new algorithms and approaches of detecting vulnerabilities and testing web applications. The researches discussed in this section explores SQL Injection and Cross-site Scripting vulnerabilities.

Suhina et al. (2008) suggested clustering web pages for vulnerabilities detection. They analyzed the structure of response pages, i.e. HTML elements, to find differences between a single page or form requested and collections of input sets that are constructed for use in testing. They discussed how clustering security audit tools should be built and what problems could be encountered when building and using those tools.

Boyer-Moore (BM) string matching algorithm was proposed in the work of Saleh et al. (2015) to detect web application vulnerabilities. The BM algorithm compares the pattern characters with the text characters from right to left. Although the results of this research proved that the method caused low false negatives and false positives for all detections with low processing time, it could only detect individual webpages but not the entire web application.

SWAT – the Secure Web Application Tactics, a tool introduced by (Jain and Sethi, 1991), detected an average of 60 faults and 424 warnings over the six web applications discussed in the study. SWAT uses Search Based Software Engineering algorithms to

implement automated web applications testing.
WAPITI is a free command line open source web application security scanning tool. It is built with Python and it scans for a number of vulnerabilities such as time-based and error-based SQL injection and XSS. It was studied in the work of Alsaleh et al. (2017) in comparison to a number of other web application scanners, such as Arachni. The results of the study showed that WAPITI recorded a 42.86% observation of XSS false alarm rate whereas no false positives were recorded for arachni.

Other Scanners, such as W3AF and OWASP Zed Attack Proxy (ZAP), were also studied alongside some other web application security scanners in the work of *Ferreira and Kleppe* (2014) against a vulnerable web application they created. The results showed that the scanners could only detect stored XSSs and Cross-site Request Forgery (CSRF). However, Reflected XSS and SQL Injections were not detected.
Abbas and Elgabar (2014) compared several security scanners and focused on injection, cross-site scripting, broken authentication, and session management vulnerabilities. They eliminated the platforms that did not support those risks. Eventually, W3AF, arachni, and Skipfish showed the best results among all scanners.
WASec will be tested against a group of web applications with known vulnerabilities from the Open Web Application Security Project (OWASP) to assess its performance efficiency in crawling, and scanning for both SQL Injection and Cross-site Scripting vulnerabilities. In addition, WASec will be tested against a number of Libyan websites that are powered by PHP and run on MySQL databases to evaluate the security of these web applications.

Methodology

For the purpose of this research a system called WASec was developed. This system consisted of four parts as shown in Figure 1.

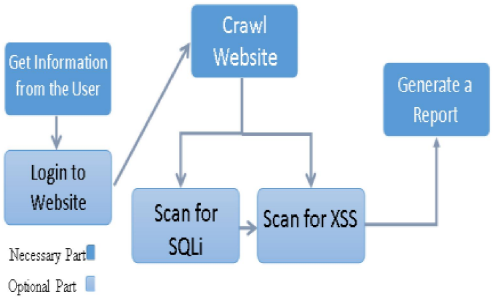

Figure 1 Process Flow Diagram for WASec System

1. Getting User Information
The user interface is designed to allow the user to enter the URL of the website to be tested. The root URL is entered and the system forwards that URL to the embedded crawler. WASec crawls only those pages that belong to the root of the given URL.

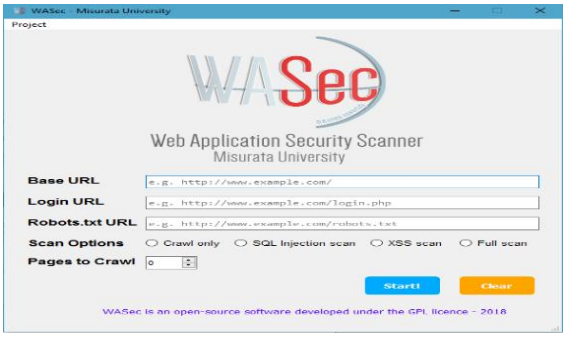

Figure 2 User Interface

2. Crawling
The crawling process takes the root URL and fetches the content of each URL found starting from the root URL. Two lists are kept, one for links that can be crawled which belong to that given website while the other one is for links that should not be crawled.

Links to other websites as well as those with robot.txt exceptions are not crawled. To enforce politeness, a certain delay period between each two consecutive crawls to the same server is applied.

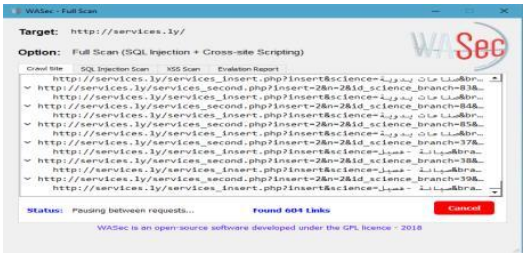

Figure 3 Crawling Results

3. Scanning for Attacks

The process of scanning starts by parsing the links crawled in the previous stage. The parsing finds forms in the page to perform vulnerability testing. The testing happens by submitting both harmful content and other data. The results of form submission is tested by comparing it to a list of SQL injections and a list of XSS payloads. The server response is then analysed to see whether or not the target page is prone to attacks. All kinds of attacks kept in the lists accompanying WASec are tested.

Figure 4 Scanning for Attacks

4. Report Generation

A report is generated for each URL tested. The report shows the link and the kind of vulnerability associated with that link. The repost is shown in Figure 5. The number of pages crawled in addition to the percentage of vulnerable pages are shown in the report.

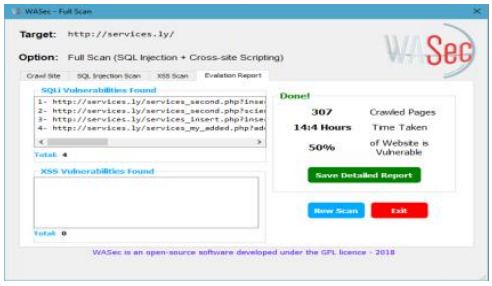

Figure 5 Final Report and Results of Tests

Evaluation

The black box evaluation method was adopted in this project. It is a widely used method of software testing that can be applied virtually to any type of software and to any level of testing (e.g. unit, system, etc.). Moreover, black box testing is widely used in web applications scanning tools.

This testing technique is not based directly on the web application's source code architecture. It is only concerned with the input and the output of the web application as it focuses on the external visible behavior of the tested application, not the code that causes it (the external behavior). The tester does not have access to the web application's source code or its details. The web application is a "black box" to the tester who cannot see inside the box (i.e. the code). The tester only knows that they can input some data to the application, and that the black box will send something in response.

To evaluate the effectiveness of WASec, a simple experiment was conducted. A sample of websites with known vulnerabilities was selected to test our software against real web applications. Testing included pages that are prone to SQL Injections and XSS. The dataset of pages used in the evaluation was taken from OWASP Vulnerable Web Applications Direc-

tory Project. Table 1 shows the applications involved in the experiment. In addition, a sample of Libyan websites was selected to test both the software developed and its effectiveness in the case of local websites. Table 2 shows the Libyan websites involved in the Experiment.

Table 1 Vulnerable Web Applications Involved in the Experiment

Application Name	Author	Technology	Require Login
Damn Vulnerable Web Application (DVWA)[23]	RandomStorm	PHP	Yes
Mutillidae II[24]	OWASP	PHP	No
BTS Lab[25]	Cyber Security & Privacy Foundation	PHP	No
SQLi-Labs[26]		PHP	No
Buggy Web App (bWAPP)[27]		PHP	Yes

Table 2 Libyan Websites Involved in the Experiment

Website Name	Website URL	Technology	Require Login
Libyan Telecom and Technology (LTT)	www.ltt.ly	PHP	No
Libyana	www.libyana.ly	PHP	No
Libya Services	www.services.ly	PHP	No
Libyan Libraries	www.services.ly/libraries	PHP	No

The testing process was performed in three phases. First, during crawling the website to test the crawler abilities. Then, automated vulnerability testing of pages that are specifically vulnerable to SQL Injection and Cross-site Scripting. Finally, a manual vulnerability testing was conducted using the payloads that reported successful exploits to check if they were truly effective and record the number of correct exploits, false positives and false negatives.

Crawling

First, the applications listed in Table 3.4 were crawled. In the results, the total links found refer to the total number of links WASec found in each application. The crawled links is the valid and never-crawled links count. Invalid links is the number of links in WASec's black list. The time taken refers to the time WASec took to crawl each web application. Table 3 shows vulnerable web applications crawling results. Table 4 shows libyan websites crawling results

Table 3 Vulnerable Web Applications Crawling Results

App Name	Total Links Found	Crawled Links	Invalid Links	Time Taken
Damn Vulnerable Web Application (DVWA)	76	23	46	4 min.
Mutillidae II	641	183	458	2 hr.
BTS Lab	144	31	113	27.55 min.
SQLi-Labs	90	74	16	45.38 min.
Buggy Web App (bWAPP)	48	5	43	2.53 min.

Table 4 Libyan Websites Crawling Results

Website Name	Total Links Found	Crawled Links	Invalid Links	Time Taken
Libyan Telecom and Technology (LTT)	253	29	224	56.27 min.
Libyana	299	151	148	6 hr.
Libya Services	307	8	299	3.4 min.
Libyan Libraries	121	14	107	21.5 min.

Vulnerability Testing

As shown in Table 3, tested pages refers to the number of links that were found suitable for testing**.** Exploited pages is the number of pages WASec found vulnerable. False negatives is the number of pages that contained a vulnerability but were not reported vulnerable while false positives is the number of pages that did not contain any vulnerability but WASec had reported vulnerable. False negatives and positives are tested manually to review the automated testing WASec had performed. Time taken is the time taken to perform the vulnerability tests. Table 5 shows Vulnerable web applications for SQL Injection Results. Table 6 shows Libyan websites SQL injection results. Table 7 lists vulnerable web applications XSS Results. Table 8 shows Libyan websites XSS Rrsults .

Table 5 Vulnerable Web Applications SQL Injection Results

App Name	Pages Tested	Pages Exploited	False Positives	False Negatives	Time Taken
Damn Vulnerable Web Application (DVWA)	2	0	0	2	19.7 min.
OWASP Mytilidae II	5	3	2	2	2:02 hr.
BTS Lab	4	4	1	0	1.18 min.
SQLi-Labs	13	13	0	0	3 hr.
Buggy Web App (bWAPP)	5	3	0	2	2 hr.

Table 6 Libyan Websites SQL Injection Results

Website Name	Pages Tested	Pages Exploited	False Positives	False Negatives	Time Taken
Libyan Telecom and Technology (LTT)	5	2	0	0	4:37 hr.
Libyana	2	0	0	0	7:13 hr.
Libya Services	8	4	0	0	5:28 hr.
Libyan Libraries	*14*	*3*	*0*	*0*	*9:24 hr.*

Table 7 Vulnerable Web Applications XSS Results

App Name	*Pages Tested*	*Pages Exploited*	*False Positives*	*False Negatives*	*Time Taken*
Damn Vulnerable Web Application (DVWA)	*2*	*0*	*0*	*2*	*2 hr.*
OWASP Mutillidae II	*5*	*4*	*0*	*1*	*3.24 min.*
BTS Lab	*8*	*7*	*0*	*1*	*36 sec.*
Buggy Web App (bWAPP)	*12*	*9*	*0*	*3*	*1 hr.*

Table 8 Libyan Websites XSS Results

Website Name	Pages Tested	Pages Exploited	False Positives	False Negatives	Time Taken
Libyan Telecom and Technology (LTT)	5	0	0	0	4:21 hr.
Libyana	2	0	0	0	2 hr.
Libya Services	8	0	0	0	8:36 hr.
Libyan Libraries	14	0	0	0	16 hr.

RESULTS AND DISCUSSION

The vulnerability testing was performed to evaluate the effectiveness of WASec's vulnerability exploiters. Figure 6 summarizes the results (from Table 5) concerning SQL Injection vulnerabilities while Figure 7 has the results (from Table 7) regarding the XSS vulnerabilities.

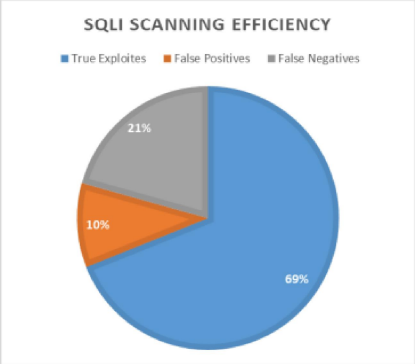

Figure 6

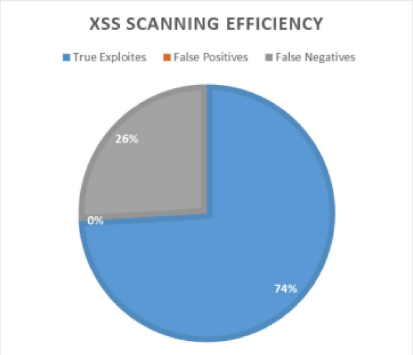

Figure 7

In the case of the Libyan websites shown in Table 8, no tested website proved to be vulnerable to cross-site scripting. When tested manually for vulnerabilities, the attacks were either stopped by the website's firewall or had no effect. Hence, it was concluded that no website showed any signs of vulnerability to XSS attacks.

It is notable though, as it appears in Table 7, that most of the tested websites were proven vulnerable to SQL Injection. The most vulnerable website was Libya Services being 50% vulnerable to SQLi followed by Libyan Libraries with 21%. The third place was taken by LTT with 6.9%. Finally, Libyana was shown to be nonvulnerable at all. However, Libyana did not have any connection to databases and operated with PHP alone. Hence, it could not have been vulnerable to SQL Injection.

Conclusion

The paper presented WASec, an open source tool developed for testing web application security vulnerabilities. The tool was designed—at the current stage—to measure vulnerabilities regarding SQL Injections and XSS. Further development may consider other kinds of threats. The experiment conducted to evaluate WASec provide good indications of the effectiveness of the tool. Further investigations are needed using larger datasets and more types of vulnerability testing.

REFERENCES

[1] Abena, A; Miguel, L; Mouanga, A; Ouamba, J; Sainard, D; Thiebolt, M; Hodi-assah, T; and Diatewa, M. (2004) Neuropsychopharmacologi- Cal effects of leaves qnd seeds extracts of Datura fastuosa. Biotech-Nologym, 3(2):109-113

[2] Jerry Gao; H.-S. J. Tsao; Ye Wu (2003). Testing and Quality Assurance for Component-based Software. Artech House. pp. 170–. ISBN 978-1-58053-735-3 .

[3] Akerele, O. (1984) WHO´s traditional medicine program: progress and perspective. WHO Chron., 38(2):76-81.

[4] Akinboro, A; and Bakare, A. (2007) Cytotoxic and genotoxic effects of aqueous extracts of five medicinal plants on Allium cepa Linn. J. Ethnopharmacol., 112(3):470-5

[5] Berger, F; Gage, F; and Vijayaraghavan, S. (1998) Nicotinc receptor- induced apoptotic cell death of hippocampal progenitor cells. J. of neuroscience, 18(7):6871-6881

[6] Shuaibu, Bala Musa; Norwawi, Norita Md; Selamat, Mohd Hasan; Al-Alwani, Abdulkareem (2013-01-17). "Systematic review of web application security development model".Artificial Intelligence Review. 43 (2): 259–276. doi:10.1007/s10462-012-9375-6. ISSN 0269-2821.

[7] Korolov, Maria (Apr 27, 2017). " Latest OWASP Top 10: The Ten Most Critical Web Application Security Risks".
[8] "Testing and Comparing Web Vulnerability Scanning Tools for SQL Injection and XSS Attacks". Fonseca, J.; Vieira, M.; Madeira, H., Dependable Computing, IEEE. Dec 2007. doi:10.1109/PRDC.2007.55