International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2018 IJSRCSEIT | Volume 3 | Issue 3 | ISSN : 2456-3307

Web Application Vulnerability and Comparison of Scanning Tools for SQL Injection and XSS Attacks

Vandana*, Shubham Khandelwal
M.Tech, Cyber Security, Raksha Shakti University, Ahmedabad, Gujarat, India

ABSTRACT

The number of vulnerabilities/threats that are being found today are much higher in applications than in operating systems. Therefore, the attacks aimed at web applications are exploiting vulnerabilities at the application level and not at the transport or network level like common attacks from the past. At the same time, quantity and impact of threats to security or vulnerabilities in such applications has grown as well. Many transactions are performed online with various kinds of web applications. Almost in all of them user is authenticated before providing access to backend database for storing all the user information. A well-designed injection can provide access to malicious or unauthorized users and mostly achieved through SQL injection and Cross-site scripting (XSS). In this paper, we are providing a vulnerability scanning and analyses tool for various kinds of SQL injection and Cross Site Scripting (XSS) attacks. Our approach can be used with any web application it is not limited to the known ones. As well as it supports the most famous Database management servers (DMS), namely MS SQL Server, Oracle, and MySQL. We have also compared the performance results of vulnerability scanner with performance of similar tools.

Keywords: Web Application Vulnerability, SQL Injection (SQLi), Cross-Site Scripting (XSS), DMS, OWASP, Vulnerability Scanner

I. INTRODUCTION

Web applications are to a great degree well known today. Almost all data frameworks and business applications like internet business, managing an account, transportation, web mail, web journals, etc., are currently working as an online web application having database application at backend. They are so exposed to assaults that any current security vulnerability will most likely be revealed and exploited, which may have an exceedingly negative effect on clients. Programmed web vulnerability scanners are regularly utilized by web application engineers and framework heads to test web applications against vulnerabilities. Thusly, believing the consequences of web weakness scanners is basic. Whatever degree would one be able to put stock in the decision conveyed by automated vulnerability scanners, particularly at the point when the device report recommends that there are no vulnerabilities in the web applications? The response to this inquiry is the point of convergence of this paper. Customary security instruments like system firewalls, interruption discovery frameworks (IDS), and utilization of encryption can secure the system yet cannot moderate assaults focusing on web applications, notwithstanding expecting that key foundation parts, for example, web servers and database administration frameworks (DBMS) are completely secure. Thus, programmers are moving their concentration from system to web applications where poor programming code speaks to a noteworthy hazard. This can be affirmed by various vulnerability reports accessible in specific destinations like www.securityfocus.com, www.ntbugtraq.com, www.kb.cert.org/vuls, www.cvedetails.com, etc. Open Web Application Security Project (OWASP) undertaking discharged its ten most web application security vulnerabilities in every 4 years [1] and the view of information gave by MITRE [2]. These reports positioned XSS as the most basic powerlessness, trailed by Injection Flaws, especially SQL infusion.

An Acunetix review result says "all things considered 70% of sites are at genuine and impending danger of being hacked. What's more, 91% of these sites contained a few type of site weakness, going from the more genuine ones, for example SQL Injection and Cross Site Scripting (XSS) [4]. These assaults essentially exploit improper coded applications due to unchecked information fields at UI. This permits the attackers to change the SQL commands that are sent to the database (SQL Injection) or through the contribution of HTML and a scripting dialect (XSS). The notoriety of these misuses is because of the effortlessness of finding furthermore, abusing such vulnerabilities the significance of the benefits they can revelation and the level of harm they may exact. These enable assailants to get to unapproved information i.e. read, write, embed, delete, truncate or modify, access privileged database accounts, impersonate another client, mimicry web applications, destroy pages, gain admittance to the web server, and so on. To keep this situation designer are empowered to take after the best coding hones, perform security surveys of the code and customary evaluating, to utilize code vulnerability analysers. Be that as it may, web application engineers ordinarily concentrate around application functionalities and on fulfilling the client's necessities because of time limitations, and effectively disregard security perspectives. Indeed, even the basic generally utilized Rapid Application Development situations create code with vulnerabilities. Web weakness scanners are frequently viewed as a simple approach to test the security of web applications, counting basic vulnerabilities, for example, SQL infusion and XSS.

The approach followed in this paper comprises of infusing programming deficiencies into a web application code what's more, checking if web vulnerability scanners can distinguish the potential vulnerabilities made by the infused issues. The conceivable making of vulnerabilities is affirmed physically keeping in mind the end goal to get precise measures of the location scope and false positives. The programming issues infused speak to the most well-known sorts of programming shortcomings found in a field think about [8].

The structure of the paper is as per the following. Segment 2 exposing vulnerability in web application. Segment 3 explain the OWASP top 10 threads and web vulnerability. Segment 4 portrays the SQLi web vulnerability and scanners. Analyses and talks about the comparison Segment 5 introduces the XXS web vulnerability and scanners. Analyses and talks about the comparison and Section 6 contains conclusion. Section 7 finishes the paper and portrays future work.

II. EXPOSING VULNERABILITIES IN WEB APPLICATIONS

There are two fundamental ways to deal with test web applications for vulnerabilities: "white box" and "black box". The "white box" approach comprises of the examination of the source code of the web application. This should be possible physically or by utilizing code examination apparatuses like FORTIFY, Ounce, Pixy, and so forth. To distinguish SQL infusion, the static analyser apparatus employments the web application code to take after all the conceivable ways and the progressions it might experience due to the control procedure of the SQL inquiry content lastly parses the outcome. Comprehensive source code investigation may not discover all security imperfections in view of the many-sided quality of the code. In these circumstances it is desirable over utilize the "discovery" approach. In this approach the scanner does not know the internals of the web application and it utilizes fluffing procedures over the web HTTP demands. It gives a programmed approach to look for vulnerabilities keeping away from the dull and repetitive errand of completing hundreds or even a huge number of tests by hand for every helplessness write. This strategy is called entrance testing and is really a type of strength testing, as the instrument submits gibberish or malignant qualities to the web application assessing its reaction to check whether the infiltration endeavours were effective. As indicated by the overview displayed in [8], infiltration testing is the second most utilized system to assess the viability of security, being utilized by 66% of the respondents. That is the reason the approach proposed in this paper receives the "discovery" testing approach utilizing web powerlessness scanners.

There are numerous business web powerlessness scanners, for example, Acunetix Web Vulnerability Scanner [4], HP Webinspect [9], w3af [12], wapiti [13] etc yet they are normally restricted scripting instruments not completely programmed as their business equal. These scanners incorporate ordinarily three primary stages: arrangement, creeping, and filtering. The arrangement organize incorporates the meaning of the Uniform Resource Locator (URL) of the web application and the setup of parameters. In the slithering stage the helplessness scanner produces a guide of the inside structure of the web application. This stage is of most extreme significance in light of the fact that neglecting to find a few pages of the application will keep their testing (in the consequent checking organize). The scanner calls the primary site page and afterward looks at its code scanning for joins. Each connection found is asked for and this technique is executed over and over until the point when no more connections or pages can be found. The filtering stage is the place the computerized entrance test is performed against the web application by re-enacting a program client tapping on connections and filling in shape fields. A mid of this stage a huge number of tests are executed. Twisted solicitations are likewise sent keeping in mind the end goal to take in the mistake reactions. The demands and the reactions are recorded and dissected utilizing weakness approaches. The reactions are too approved utilizing information gathered amid the creeping arrange. Amid this stage new connections are often found and when this happens they are added to the aftereffect of the crawler keeping in mind the end goal to be likewise examined for vulnerabilities. After the filtering stage the outcomes are appeared to the client and they might be put something aside for later investigation. Most scanners likewise demonstrate some nonspecific data about the vulnerabilities found, including how to maintain a strategic distance from or revise them. Other than the graphical UI, most scanners likewise have an order line application with a few parameters went for robotization by utilizing clump occupations.

Scanners utilize the design motor of one web program to process the reactions of the web server. There are a few design motors accessible, similar to Gecko from Mozilla, WebKit from Safari, Presto from Opera, in any case, the scanners generally utilize the Trident from Internet Voyager. The design motors decipher the HTML code diversely and do not totally bolster its related principles. A few vulnerabilities influence just a particular program or form, for the most part due to the casual way the format motor treats the HTML code. Scanners additionally have an accumulation of marks of known vulnerabilities of various forms of web servers, working framework and furthermore of some system designs. These marks are refreshed frequently as new vulnerabilities are found. They likewise have a pre-characterized set of trial of some nonspecific sorts of vulnerabilities like SQLi and XSS. In the scan for vulnerabilities like XSS and SQLi, the scanners execute heaps of example varieties adjusted to the particular test keeping in mind the end goal to find the defencelessness furthermore, to check in the event that it is not a false positive. The tests for these sorts of vulnerabilities, including both the groupings of info- esteems. The best approach to distinguish achievement or disappointment, are very not quite the same as scanner to scanner, so the outcomes acquired by various apparatuses may change a considerable measure (this is really one reason why it is so vital to have intends to think about various scanners).

III. WEB VULNERABILITY ATTACK THREATS

The numbers of security vulnerabilities that are being found today are much higher in applications than in operating systems. This means the attacks aimed at web applications are exploiting vulnerabilities at the application level and not at the transport or network level like common attacks from the past. The Open Web Application Security Project (OWASP) has put together what is considered the definitive standard list of top threats to Web applications. It is called "The OWASP Top 10 Project" and it represents a consensus on the major areas of threat by category. The Top 10 threats [TABLE 1] as they exist currently are as follows: [1]

Table 1. Owasp Top 10 Vulnerability List

OWASP TOP 10 - 2013	**OWASP TOP 10 - 2017**
*A1 - Injection	
*A2 - Broken Authentication and Session Management	
A3 – Cross-Site Scripting	A3 - Sensitive Date Exposure
A4 – Insecure Direct Object Reference	A4 – XML External Entities (XXE)
A5 – Security Misconfiguration	A5 – Broken Access Control
A6 – Sensitive Date Exposure	A6 - Security Misconfiguration
A7 – Missing Function Level Access Control	A7 - Cross-Site Scripting
A8 – Cross Site Request Forgery (CSRF)	A8 – Insecure Deserialization
*A9 – Using Components with Known Vulnerabilities	
A10 – Unvalidated Redirections and Forwards	A10 – Insufficient Logging & Monitoring

*Common vulnerability with same Place

IV. WEB VULNERABILITY SCANNING TOOLS FOR SQL INJECTION

A. SQL INJECTION (SQLi)

SQL injection is an attack in which the SQL code is inserted or appended into application/user input parameters that are later passed to a back-end SQL server for parsing and execution. The primary form of SQL injection consists of direct insertion of code into parameters that are concatenated with SQL commands and executed [5]. A less explicit attack embedded with malicious code and strings and stored in a table or as metadata. When the stored strings are subsequently concatenated into a dynamic SQL command, the malicious code is executed. SQLi is amongst the top 10 threat to web application vulnerabilities, having 1st position in OWASP top 10 vulnerability list 2017 as well as 2013.

SQLi attack is classified in three categories as –

1. **Error-Based SQLi:** Asking the Database a question that will cause an error, and gleaning information from the error.
2. **Union-Based SQLi:** The SQL command UNION is used to combine the results of two or more SELECT SQL statements into a single result. Same useful for SQLi.
3. **Blind SQLi:** Asking the Database a true/false question and using whether valid page returned or not, or by using the time it took valid page to return as the answer to the question.

SQLi can be broken up into three times –

1. **In-Band:** Data is extracted using the same channel that is used to inject the SQL code. This is the most straightforward kind of attack, in which the retrieved data is presented directly in the application web page.
2. **Out-of-Band:** Data is retrieved using a different channel.
3. **Inferential:** Here no actual transfer of data, but the attacker/hacker is able to reform the information by sending requests and observing the resulting behaviour of the website/database server.

B. IMPORTANT CONCEPTS IN SQLi ATTACK

- **Dynamic String Building -**
  Dynamic string building is a programming technique that enables developers to build SQL statements dynamically at runtime. Developers can create general-purpose, flexible applications by using dynamic SQL.
- **Parameterized queries -**
  Parameterized queries are queries that have one or more embedded parameters in the SQL statement. Parameters can be passed to these queries at runtime; parameters containing embedded user input would not be interpreted as commands to execute, and there would be no opportunity for code to be injected.

```
// A dynamically built SQL string statement in PHP
$query = “SELECT * FROM table WHERE field = ‘$_GET[“input”]’”;
// A dynamically built SQL string statement in .NET
query = “SELECT * FROM table WHERE field = ‘” + request.getParameter(“input”) + “’”;
```

C. SIMPLE SOLUTION FOR SQLi

SQL databases interpret the quote character (‘) as the boundary between the code and data. They assume that anything following a quote is a code that it needs to run and anything encapsulated by a quote is data.

If we were to enter the single-quote character as input to the application, we may be presented with either one of the following errors; the result depends on a number of environmental factors, such as programming language and database in use, as well as protection and defence technologies implemented.

```
// Build dynamic SQL statement
$SQL = “SELECT * FROM table WHERE = ‘$_GET[“input”]’;”;
// Execute SQL statement
$result = mysql_query($SQL);
// check to see how many rows were returned from the database
$rowcount = mysql_num_rows($result);
// iterate through the record set returned
$row = 1;
while ($db_field = mysql_fetch_assoc($result)) {
    if ($row <= $rowcount) {
            print $db_field[$row]. “<BR>”;
             $row++;
          }
}
```

Note: mysql_fetch_assoc() - supplied argument is not a valid MySQL result resource.

D. WEB APPLICATION SCANNERS AGAINST SQL

1. COMMERCIAL TOOLS

1.1. HP Webinspect

HP WebInspect is a commercial penetration testing tools from HP [9]. For this paper the 15-day evaluation version will be used. The list of vulnerabilities it claims to test for can be found in its data sheet. It appeared that this evaluation version can only be used against a web application on the web server of HP, therefore this penetration testing tool will only be used in one very limited test.

1.2. JSky

JSky is a commercial penetration testing tools from NOSEC [11]. For this paper the 15-day fully functional evaluation version will be used. The list of vulnerabilities it claims to test for can be found on the tool's website.

2. OPEN SOURCE TOOLS

2.1. W3AF

w3af is the abbreviation of the Web Application Attack and Audit Framework [12]. It is an open-source program, written in Python. It uses plugins to perform the attacks on the web application. A description of the vulnerabilities these plugins are claimed to detect can be found on the tool's website. It uses a menu-driven text-based structure, but it also has a GUI. Results are outputted to the console or to an XML-, text-, or HTML-le.

2.2. Wapiti

Wapiti is another open-source program written in Python [13]. The vulnerabilities it claims it can detect can be found on the tool's website. It works from the command-line completely automatically, however command-line options can be used to customize scanning. Output is written to the console or an XML-, text-, or HTML-le.

2.3. Arachni

Arachni is a Web Application Vulnerability Scanning Framework [14]. It is an open source program written in Ruby. It has a modular setup. A description of what the modules can test for and this can be found on the tool's website. Now it only has a command-line interface. Running the program with as parameter a URL will automatically audit the web application on that URL with all modules. The audit can be customized with options on the command line. The output can be sent to the console or a text-, XML-, HTML- or AFR (Arachni Framework Report)-le.

2.4. Websecurify

Websecurify is an open-source integrated web security-testing environment [15]. A list of vulnerabilities it claims it can detect can be found on the tool's website. It has a GUI interface and performs the testing automatically. Very few options can be controlled via settings.

Simple Comparison of SQLi tools –

Table 2. Results of the simple SQL injection test

Type / Tool	SQL Injection	Blind SQL Injection
JSky	Yes	No
W3af	Yes	Yes
Wapiti	Yes	Yes
Arachni	Yes	Yes
Websecurify	Yes	No

V. WEB VULNERABILITY SCANNING TOOLS FOR XXS

A. CROSS-SITE SCRIPTING (XXS)

Cross-site Scripting (XSS) refers to client-side code injection attack by which an attacker can execute malicious script (or malicious payloads) into a legitimate website or web application. XSS is amongst the top 10 threat to web application vulnerabilities, having 7th position in OWASP top 10 vulnerability list 2017, and occurs when a website makes use of invalidated or not properly encoded user input within the output it generates. By XSS, an attacker does not only target a victim directly. Instead, an attacker would exploit a vulnerability within a website or web application that the victim would visit, essentially using the vulnerable website as a vehicle to deliver a malicious script to the victim's browser. While XSS can be take advantage of within VBScript, ActiveX and Flash (although now considered legacy or even obsolete), unquestionably, the most widely abused is JavaScript – primarily because JavaScript is fundamental to most browsing experience. XSS attacks can be classified on the basis of, who is executing the script as follows:

1. **Reflected XSS:** When malicious script is executed by the client because of the vulnerability in a webpage

2. **Stored XSS:** When malicious script is executed by the server because of the vulnerability in a webpage
3. **DOM-based XSS:** when the client using DOM. executes malicious script.
4. **Self-XSS:** A hacker may ask you to open up the developer console (opens with Ctrl+Shit+I in Firefox) and ask you enter a script there.

B. WEB APPLICATION SCANNERS AGAINST XSS

This area presents the web application scanners utilized as a part of our task. Most of the time, commercial scanners are less demanding to utilize. They have further developed UIs making a difference control their examining exercises. Interestingly, numerous open source tools have simple support, and some of them only rely on command line operations to configure their scanning processes. They usually take more time to set up. But on the other hand, open source scanners have no restriction for users to access their technical details, which is helpful for further studying their detection mechanisms.

C. WEB APPLICATION SCANNERS AGAINST CROSS-SITE SCRIPTING

1. COMMERCIAL TOOL

1.1. Netsparker (Community Edition)

The release we utilize is Netsparker community edition, version 1.7.2.13. It is a business scanner guaranteed to be free from false-positives, as portrayed in the product site. It shares the same user interface with the professional edition. To perform automatic authentications, Netsparker enables clients to utilize cookies of verified/authenticated sessions. In our assessment, cookie information is obtained by the network tamper tool, i.e., the tamper data add-on of Firefox, which can help see and alter the contents in request for headers and parameters [10]. To keep up the authenticated session status, Netsparker enables clients to specify the key words that ought to be included or rejected in the website pages being scanned and users can use this method to detect and avoid logout pages. This component is very helpful, since it is frequently that the session has a logout state when a logout page is visited, and numerous pages can't be reached afterwards. In spite of the fact that a few advanced reporting functionalities are disabled in this free version, regardless it gives adequate data, for example, the severity type, background description, request and response content focusing on the reported locations, and attack strings used to exploit the vulnerabilities. The crawling results of the target sites can likewise be seen in the report page.

1.2. Acunetix Web Vulnerability Scanner (Free Edition)

Acunetix consists of advanced penetration testing tools to take web security testing further, while integrating both with external tools as well as tools to aid-in testing business-logic web applications. Acunetix focuses more on web application vulnerabilities and variants thereof, and does a much better at detection than traditional VM tools. Acunetix centres around being a decent scanner giving great specialized outcomes and a palette of reports [4]. Acunetix is a strong item to get your Application Security Testing program off the ground. As dependably guarantee that you comprehend your SDLC so you get the scope you have to test. Acunetix have additionally as of late discharged an online variant of the scanner for the review of open web confronting Web Servers and Network Interfaces. Acunetix free release is another free scanner with no period restriction, and the adaptation we use is 9.0. It has an advanced graphic use interface. To perform the login operations naturally, Acunetix has a recorder with a scaled down program to record the clients' logging activities, counting the URLs went by, the secret word entered, and so on. The scanner can recover the recorded data, which is alluded as login succession, to perform programmed confirmations at later checking stages.

1.3. Skipfish

Skipfish is an active web application security reconnaissance tool, made in Google for developer or sys admins. It is a cross-platform security analysis tool, released under the Apache license [16]. It is compatible with Windows (Cygwin), Mac OS X, BSD and Linux. Designed for site and web applications developers, Skipfish conduct an audit of the code looking for security vulnerabilities. It is written in C, optimized for HTTP, it is presented as little CPU intensive. This tool easily fills 2,000 requests per second with adapted objectives. The tool includes heuristics for most websites and is seen with machine learning capabilities. It also provides automatic completion of forms.

1.4. Wapiti

Wapiti is a web application vulnerability scanner [13]. This tool allows to audit the security of web applications. It performs "black-box" scans, i.e. it does not study the source code of the application but will scans the web pages of the deployed web application, looking for scripts and forms where it can inject data. Once it gets this list, Wapiti acts like a fuzzer, injecting payloads to see if a script is vulnerable.

1.5. XSSploit,

It is an XSS scanner and exploiter written in Python. It first crawls your website; identifying and analyzing any forms, it finds to detect any XSS vulnerabilities. If used as part of a penetration test, any vulnerabilities can then be exploited using the exploit generation engine to automatically create the exploit payload [17].

In the evaluation for XSS injection patterns, we can use some other tools, including IBM Rational AppScan version 7.8 trial edition, NStalker free edition 2012, and Nikto version 2.1.4. IBM AppScan is quite powerful at detecting various types of vulnerabilities, but its trial edition has a Trial period limitation and can only scan applications deployed on the test websites specified by the vendor. NStalker does not support session maintenance in its free edition. Nikto has many features for detecting problems on the server side, but XSS detection is not its specialty. For these reasons, we do not evaluate these scanners in our case studies of real-life web applications,

Table 3. Usability Comparison for 4 scanners

FUNCTIONLITY	TOOLS	NETSPARKER	ACUNETIX (free edition)	SKIPFISH	WAPITI
Overal l	GUI	YES	YES	NO	NO
Crawling	Stop after Crawling	NO	YES	NO	NO
	Exclude URL	YES	YES	YES	YES
Session Maintenance	Login method	Cookie	Login Sequenc	Cookie	Cookie File
	Exclude Logout	YES	YES	YES	YES
Reporting	Show crawl result	YES	YES	YES	NO
	Severity classificati on	YES	YES	YES	YES
	Request & Response detail	YES	YES	YES	NO
	Attack pattern	YES	YES	YES	YES

VI. CONCLUSION

Based on our evaluation, we can see that among Netsparker, Acunetix, Wapiti, and Skipfish, commercial scanners Netsparker and Acunetix have overall better performance. Comparing to Wapiti, Skipfish has better performance in the real-life application and it can bypass some validation mechanisms in controlled test applications, whereas, Wapiti cannot. , Wapiti could not reach enough web resources in the crawling phase during several scanning runs, and did not reach its full potential in XSS detection. In order to improve this, scanners should have good usability in the functionalities like auto login and session maintenance, Netsparker and Acunetix can identify more injection points. Due to their better performance in this step, they both are able to send injections to more locations, which are actually vulnerable to XSS, attack, thus have better scanning results.

XSS and SQL injection vulnerabilities in web applications has huge risk not only for the web applications but also for users as well. We studied many existing approaches to detect and prevent these Vulnerabilities in an application, giving a brief note on their advantages and disadvantages. All the approaches followed by different authors' leads to a very interesting solution; however some failures are associated with almost each one of them at some point. Furthermore these scanners don't support all web applications, many of them supports only known web applications with known vulnerabilities. In this paper we have explained a vulnerability scanning and analyzing tool of various kinds of SQL injection and Cross Site Scripting (XSS) attacks. Our approach can be used with any web application not only the known ones. As well as it supports the most famous Database management servers, namely MS SQL Server, Oracle, and MySQL.

VII. FUTURE WORK

We can evaluate more tools or the newest version for the four tools in the future. Netsparker and Acunetix have very close performance according to our evaluation results. To distinguish the performance of different scanners, we can have more case studies of real-life vulnerable web applications, written in different languages. Develop a GUI for the scanner script, so make it would be easy for anyone to install and use the scanner. Add full support for all known XSS detection and SQLI techniques. Implement a local proxy module to be included in the scanner work, which will make a full vulnerability analysis environment

VIII. REFERENCES

[1]. Prof Open Web Application Security Project (OWASP): OWASP Top Ten Project. https://www.owasp.org/index.php/Top_10-2017_Top_10 Accessed Jan 7th, 2018

[2]. CWE. 2011 CWE/SANS Top 25 Most Dangerous Software Errors. September 13, 2011. http://cwe.mitre.org/top25/. Accessed Jan 16th, 2018

[3]. Chen, S. Web Application Scanners Accuracy Assessment. Security Tools Benchmarking. December 26,2010.http://sectooladdict.blogspot.com/2010/12/web-application-scanner-benchmark.html. Accessed Jan 20th, 2018

[4]. Acunetix Web Application Security. http://www.acunetix.com/ Accessed February 2nd, 2012.

[5]. Justin Clarke Author, "SQL Injection Attacks and Defense," Second ed. Accessed February 4, 2018.

[6]. Gordeychik, S. Web Application Security Statistics. Web Application Security Consortium. http://projects.webappsec.org/w/page/13246989

/Web%20Application%20Security%20Statistics . Accessed February 8, 2018.

[7]. Kiezun, A., Guo, P.J., Jayaraman, K., and Ernst, M.D. Automatic creation of SQL Injection and cross-site scripting attacks. Proceedings of the 31st International Conference on Software Engineering. 2009. pp. 199-209. Accessed February 10, 2018.

[8]. Doup´e, A., Cova, M., and Vigna, G. Why Johnny Cannot Pentest: An Analysis of Black-box Web Vulnerability Scanners. Proceedings of the 7th international conference on Detection of intrusions and malware, and vulnerability assessment. 2010, pp. 111-131. Accessed February 12, 2018.

[9]. HP: HP WebInspect (2009) http://www.hp.com/spidynamics/products/webinspect/. Accessed February 14th, 2012

[10]. Mavituna Security - Netsparker Vulnerability Scanner http://www.mavitunasecurity.com/netsparker/. Accessed February 16th, 2018

[11]. NOSEC: JSky Vulnerability Scanner (2010) http://www.nosecinc.com/en/products/jsky/ Accessed February 18, 2018.

[12]. Riancho, A.: w3af Vulnerability Scanner (2011) http://w3af.sourceforge.net. Accessed February 20, 2018.

[13]. Surribas, N.: Wapiti Vulnerability Scanner (2009) http://wapiti.sourceforge.net. Accessed February 26, 2018.

[14]. Laskos, A.: Arachni - Web Application Vulnerability Scanning Framework (2011) https://github.com/Zapotek/arachni. Accessed March 1, 2018.

[15]. Websecurify - Vulnerability Scanner (2011) http://www.websecurify.com. Accessed March 8, 2018.

[16]. Zalewski, M., Heinen, N., Roschke, S. Skipfish – Web Application Security Scanner. http://code.google.com/p/skipfish/wiki/SkipfishDoc. Accessed March 20th, 2018

[17]. XSSploit - SCRT Information Security https://www.scrt.ch/en/attack/downloads/xssploit Accessed March 25th, 2018