ACSIJ Advances in Computer Science: an International Journal, Vol. 3, Issue 6, No.12 , November 2014
ISSN : 2322-5157
www.ACSIJ.org

ACSIJ
WWW.ACSIJ.ORG

A survey on web penetration test

Mahin Mirjalili[1], Alireza Nowroozi[2], Mitra Alidoosti[3]

[1]Department of information security, Malek-Ashtar university of technology, Tehran, Iran
Mahinmirjalili@mut.ac.ir

[2]Department of information security, Malek-Ashtar university of technology, Tehran, Iran
Nowroozi@mut.ac.ir

[3]Department of information security, Malek-Ashtar university of technology, Tehran, Iran
alidoosti@mut.ac.ir

Abstract

This paper reviews the penetration test specifically in the field of web. For this purpose, it first reviews articles generally on penetration test and its associated methods. Then articles in the field of web penetration test are examined in three aspects: comparing automatic penetration test tools, introduction of new methods or tools for manual penetration test, and articles that presented a test environment for training or checking various instruments and methods. This article studied 4 different methodologies for web penetration test, 13 articles for comparing web vulnerability scanners, 10 articles that proposed a new method or tool for penetration test and 4 test environments.

Keywords: *Penetration test, web scanner, web application, web vulnerabilities.*

1. Introduction

Penetration test is a security evaluation process for network or computer systems that simulates an attack by an ethical hacker. The most important distinction between a hacker and a penetration tester is that penetration test is done with a license and a signed contract with an organization or company, and the output is provided as a report. The goal of penetration test is to increase data security. Security information and weaknesses that are specified in penetration test are considered confidential and shall not be disclosed until complete resolution of defects.

Given the importance of web application security, this article reviews studies in the field of penetration test, particularly web penetration test.

Questions that engaged the mind of researchers in the field of penetration test can be expressed as follows:

- How the penetration test is performed?
- What are types of penetration test?
- How the penetration test is done automatically?
- What tools can we use to perform an automatic penetration test?
- Comparison of tools and their effectiveness
- What are the new tools and methods and what are their features?
- How can we examine various tools and techniques?

This paper attempts to answer these questions by examining 4 different methodologies for penetration test, 13 articles for comparing web vulnerability scanners, 10 articles that proposed a new method or tool for penetration test and 4 test environments.

Due to the large volume of papers in the studied area, some criteria were used for paper selection such as covering a wide time period from 2006 to 2014 and the number of citations per paper. Most selected articles were considered by many authors in previous years.

We will initially review articles that provided a method for the penetration test, then papers in the field of web penetration test in three views:

- Articles on the comparison of existing methods and tools for web penetration test.
- Articles that proposed a new tool or method for web penetration test.

Copyright (c) 2014 Advances in Computer Science: an International Journal. All Rights Reserved.

- Studies that proposed a test environment for testing tools and ideas.

2. Penetration testing

2.1. History and Importance

Penetration testing is one of the oldest methods for assessing the security of a computer system. In the early 1970's, the Department of Defense used this method to demonstrate the security weaknesses in computer systems and to initiate the development of programs to create more secure systems. Penetration testing is increasingly used by organizations to assure the security of Information systems and services, so that security weaknesses can be fixed before they get exposed [1].

Large companies have important data and one of their concerns is to protect the data. The penetration test tests security mechanisms of companies by simulating multiple attacks. In the situations like new infrastructure is added, Software is installed, System updates are applied, Security patches are applied and User policies are modified it is necessary to do penetration testing. Some of the principal reasons for adopting penetration testing are Security Issues, Protect Information, Prioritize security risks and Financial Loss [2].

The penetration test can be done either manually or automatically. Manual penetration test requires a skilled and experienced tester team to control all things. They must be physically present at the test duration. Thus this is not an affordable option. Automatic penetration test is a simple and safe way to perform all tasks related to the penetration test. Moreover, since most work is done automatically, it is more economical in terms of time. Another advantage of this test is the ability to reuse the set parameters for the test. In [3], a comparison is made between the two tests, as follows:

Table 1- manual vs. automated penetration testing

	MANUAL	AUTOMATED
Testing Process	Labor-intensive, inconsistent and error-prone, with no specific quality standards. Requires many disparate tools. Results can vary significantly from test to test. Generally requires highly-paid, experienced security personnel to run and interpret tests.	Fast, easy and safe. Eliminates errors and tedious manual tasks. Centralized and standardized to produce consistent and repeatable results. Easy to use and provides clear, actionable reports.
Network Modification	Often results in numerous systems modifications.	Systems remain unchanged.
Exploit Development and Management	Developing and maintaining an exploit database is time-consuming and requires significant expertise. Public exploits are suspect and can be unsafe to run. Re-writing and porting code is necessary for cross-platform functionality.	Product vendor develops and maintains all exploits. Exploits are continually updated for maximum effectiveness. Exploits are professionally developed, thoroughly tested, and safe to run. Exploits are written and optimized for a variety of platforms and attack vectors.
Cleanup	Tester must remember and undo all changes. Backdoors can be left behind	Leading products offer comprehensive cleanup with one click and never install backdoors.
Reporting	Requires significant effort, recording and collating of all results manually. All reports must be generated by hand.	Comprehensive history and findings reports are automatically produced. Reports are customizable.
Logging/ Auditing	Slow, cumbersome, often inaccurate process.	Automatically records a detailed record of all activity.
Training	Testers need to learn non-standardized, ad-hoc testing methods.	Users can learn and install in as little as one day.

Copyright (c) 2014 Advances in Computer Science: an International Journal. All Rights Reserved.

Penetration testing can be segregated into the following classes [4]: Attack visibility: Blue-teaming or Red-teaming, and system access: Internal testing or External testing.

Blue-teaming is done with the consent of an entire organization. The information security team is fully aware of the testing requirements as well as resources needed. Blue-teaming is a more efficient way to perform testing as the system availability is not an issue and hence there is a considerable reduction in the overall time for testing. The shorter test times mean lesser system idle time and reduced testing costs. Red-teaming refers to testing that is performed in a stealth manner without the knowledge of IT staff [4]. Upper-level management authorizes such an exercise. The objectives of the test are to judge the strength of the network security, the awareness of IT organization, and its ability to follow the standard protocols. The entire test is done without the support of the organization's resources.

2.2. Available methodologies and techniques

A methodology is a scheme that is used to reach the destination. Lack of use of a methodology for penetration test may lead to an incomplete test, high time-consumption, failure and test ineffectiveness. Despite the large number of methodologies, there is nothing called "true methodology" and each penetration test can have a different methodology but the use of a methodology leads to a professional and efficient penetration test at lower cost.

The methodology considered for the penetration test usually has 4 or 7 phases. Although the name or number of phases is different in different methodologies, they all show an overview of penetration test. For example, some methodologies use the term "information gathering" while some call this process "reconnaissance."

The methodology proposed in [5] consists of four phases: reconnaissance, scanning (port scanning, vulnerability scanning), exploitation and maintaining access. The first phase of the penetration test is reconnaissance which focuses on information gathering. The more information gathered at this stage, the more successful the next stages will be. The second phase of this methodology is divided into two categories: port scanning which obtains a list of open ports and services running on each of them, and vulnerability scanning which is a process to recognize weaknesses of desired services and applications. According to the results obtained in step 2 and knowing what ports are open, what services are running on this port and what vulnerabilities they have, one can attack the target. Maintaining access is the last phase. Most accesses obtained in the attack phase are temporary, and are removed after disconnection. In this phase, it is tried to maintain access. Although this reference ignored "reporting" as a step in the penetration test, it stated the last activity of a penetration test as reporting. According to [5], reporting should include details on how to perform the test, a summary of the found security threats, cases the test does not cover, etc.

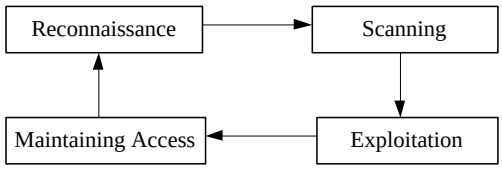

Figure 1 - proposed methodology in [5]

In the NIST penetration test methodology, the penetration test consists of four phases: planning, discovery, attack, reporting. In the planning phase, rules are defined and objectives of the test are set. The discovery phase is performed in two stages. The first includes test initiation and information collection and the second stage, which takes place after the attack phase, includes vulnerability analysis. In the attack phase, which is known as the heart of penetration test, various vulnerabilities in the target are examined. The report is prepared in conjunction with the other phases. In the planning phase, the evaluation plan is developed. In the discovery and attack phases, events are usually recorded and periodically reported to the director. At the end of the test, a report is provided to describe recognized vulnerabilities, ranking of risks, and tips for how to improve the known weaknesses [6].

Copyright (c) 2014 Advances in Computer Science: an International Journal. All Rights Reserved.

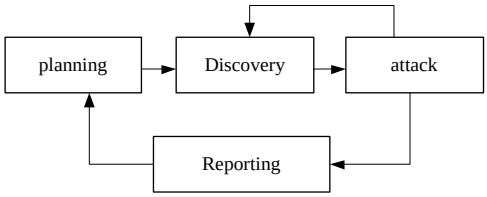

Figure 2- proposed methodology in [6]

The methodology presented in [7] consists of three main parts: information, team, and tools. In the information part, information is gathered on the target using different methods. In this paper, the information phase is defined in four steps: studying the network, identifying the OS, scanning ports and identifying services. The second part of the methodology is team formation. If teams are formed with different roles and responsibilities, the penetration test will be carried out more effectively. Another important parameter in the penetration test is to use tools. To do an effective penetration test, it is better to dominate a smaller number of tools, instead of many tools. Another point discussed in this article is to set policies that must be followed by the tester and the client. In the proposed policy, there are issues such as preservation of information obtained by the tester and reporting them completely at the end of the penetration test, scheduling agreement, confidentiality of all information such as contract, use of information obtained just for the test, lack of responsibility of the penetration tester in the event of a real attack, and so on.

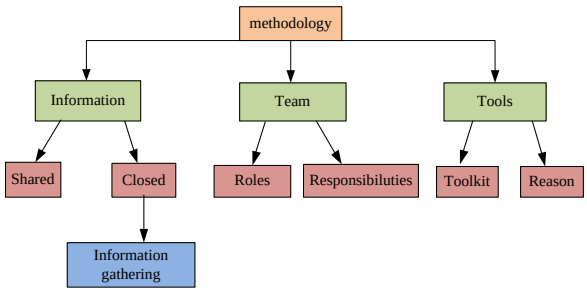

Figure 3- proposed methodology in [7]

One of the proposed methodologies is Open Source Security Testing Methodology Manual designed by ISECOM. The OSSTMM phases include induction, interaction, inquest, and intervention.

In the induction phase, the time period and type of the test should be specified. The interaction phase indicates the objectives of the penetration test. In the inquest phase, the maximum possible data is achieved on the target system. In the final phase, the security performance is measured. After completion of the penetration test, results are processed and a report is prepared. The OSSTMM uses a set of tools called Security Test Audit and Reporting for processing the results [8].

In [41], a penetration test scheme is presented for web application based on the RUP test scheme. Each of the described methodologies may be appropriate for different purposes and penetration tests. As mentioned earlier, it cannot be said that a methodology is better than another. This scheme provides a systematic, consistent and affordable method fully integrated with the security-based software development lifecycle for the penetration test and improves the accuracy, quality and performance of such tests. This study also presents a database of techniques and tools required for the penetration test of web applications which was compiled using various sources including valid guidelines and standards and test techniques on the Internet.

In [42], an penetration test methodology is presented based on the agile method which uses the benefits of the agile method in the process of penetration test, and a model is design based on the Scrum and XP methodologies which show information flows between activities. Thus, this method improves the penetration test cycle and can be a framework for improving the accuracy, efficiency, job satisfaction and quality of testers. In this process, change management can be easily done and information technology goals are aligned with business goals, interaction with customer increases, so prioritizing the depth and range of penetration test is easier.

3. Web penetration test

Today, with the Internet expansion and use of web applications in various fields such as military, medical, finance, etc. web security is an important concern, and the penetration test is used to ensure it. The penetration test can be performed manually or

Copyright (c) 2014 Advances in Computer Science: an International Journal. All Rights Reserved.

automatically. The two options are compared in Table 1.

Tools used to recognize vulnerabilities can be divided into three categories based on information of the target application that they use: white-box, black-box and grey-box.

A white-box tool uses the target application code to assess vulnerabilities. By analyzing the code application, a white-box tool can find all the hidden application paths which lead to finding vulnerabilities in the application path. In this set of tools, due to access to the application code, vulnerabilities may be reported that are not available. In other words, it is likely that there is no possibility to use a known vulnerability. The disadvantage of white-box test is its reliance on a specific language and framework.

Unlike white-box tools, black-box tools assume that there is no knowledge of the application code. Instead of using the application code, the tester, like a regular user, uses the application with a browser. A black-box tool first examines different parts of the application to find all possible injection vectors. Whatever way through which the attacker can enter the application is called attack vector like URL parameters, HTML form parameters, cookies, HTTP headers etc. After identifying injection vectors to the application, inputs are given to recognize vulnerabilities. This process is called fuzzing. In fuzzing, the type of injection vector and its use duration can be different in black-box tools. Finally, these tools examine http and html responses related to fuzzing, and if successful, report it as vulnerability.

Among advantages of white-box over black-box tools, one can point to independence of the application code and less false positive. Given that a black-box tool can only identify vulnerabilities that run their related attack, one of its disadvantages is that it cannot guarantee to identify all the vulnerabilities of the application.

Grey-box tools, as their name implies, are a combination of black-box and white-box. These tools use static white-box analysis techniques to identify vulnerabilities. Then they try to really attack identified vulnerabilities to confirm them. If this step is successful, the vulnerability is reported. Grey-box tools can find vulnerabilities in all the application paths with low false positive but, like white-box tools, they depend on a specific language or framework [9].

3.1. Web vulnerabilities

Given that applications are highly vulnerable, invaders use different methods and paths to damage various organizations. These vulnerabilities can be very simple or complex. In complex ones, the discovery and exploitation become very difficult for invaders. According to the characteristics of its business environment and associated risks, especially threatening factors, each organization identifies implemented security controls and their impact on business and financial matters of the organization. Owasp annually publishes a list of ten common vulnerabilities. The last list published in 2013 includes the following vulnerabilities [36]:

1- Injection: Injection flaws, such as SQL, OS, and LDAP injection occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.
2- Broken Authentication and Session Management: Application functions related to authentication and session management are often not implemented correctly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities.
3- Cross-Site Scripting (XSS): XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation or escaping. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.
4- Insecure Direct Object References: A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, or database key. Without an access control check or other protection, attackers can manipulate these references to access unauthorized data.
5- Security Misconfiguration: Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server,

Copyright (c) 2014 Advances in Computer Science: an International Journal. All Rights Reserved.

and platform. Secure settings should be defined, implemented, and maintained, as defaults are often insecure. Additionally, software should be kept up to date.

6- Sensitive Data Exposure: Many web applications do not properly protect sensitive data, such as credit cards, tax IDs, and authentication credentials. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data deserves extra protection such as encryption at rest or in transit, as well as special precautions when exchanged with the browser.

7- Missing Function Level Access Control: Most web applications verify function level access rights before making that functionality visible in the UI. However, applications need to perform the same access control checks on the server when each function is accessed. If requests are not verified, attackers will be able to forge requests in order to access functionality without proper authorization.

8- Cross-Site Request Forgery (CSRF): A CSRF attack forces a logged-on victim's browser to send a forged HTTP request, including the victim's session cookie and any other automatically included authentication information, to a vulnerable web application. This allows the attacker to force the victim's browser to generate requests the vulnerable application thinks are legitimate requests from the victim.

9- Using Components with Known Vulnerabilities: Components, such as libraries, frameworks, and other software modules, almost always run with full privileges. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications using components with known vulnerabilities may undermine application defenses and enable a range of possible attacks and impacts.

10- Invalidated Redirects and Forwards: Web applications frequently redirect and forward users to other pages and websites, and use untrusted data to determine the destination pages. Without proper validation, attackers can redirect victims to phishing or malware sites, or use forwards to access unauthorized pages.

3.2. Black-box web vulnerability scanners

In recent years, automated or semi-automated application scanning has been addressed to find vulnerabilities. To perform an automatic penetration test, there are many commercial and open source scanners. Next, we introduce some commercial and open source scanners and then review research on the web penetration test. Articles in this category have considered three issues: comparing scanner available tools, developing a new scanner and designing a vulnerable application.

3.2.1. Open-source vulnerability scanners

Table 2 presents some open source scanners and their developer companies, used technology and platform. Then Table 3 compares them in terms of use scale, initial scan method and outputs of each one.

Table 2-Open-source scanners overview [39]

	scanner	Company	Technology	Platform
1	**Websecurify**	GNU CITIZEN	JavaScript (General)	Windows, Mac, Linux
2	**Skipfish**	Micheal Zalewski	C (General)	Linux, FreeBSD, Mac OS X, Windows (Cygwin)
3	**Wapiti**	Informatica Gesfor	Python (2.6.x)	Unix/Linux, FreeBSD, Mac OS X, Windows
4	**Parosproxy**	MileSCAN	Java 1.4x	Linux, Mac OS X, Windows
5	**Arachni**	Tasos Laskos	Ruby (1.9.x)	Windows
6	**Opent Acunetix**	John Martinelli	Java 1.6	Windows
7	**Grendel-Scan**	David Byrne	Java1.6	Linux, Mac OS X, Windows
8	**W3af**	W3af developers	Python (2.5.x)	Linux, Mac OS X, Windows
9	**WebScarab**	OWASP	Java (1.5.x)	Linux, Mac OS X, Windows
10	**SQLmap**	SQLmap developers	Python 2.6	Linux, Mac OS X, Windows
11	**ZAP**	OWASP	Java 1.6x	Linux, Mac OS X, Windows
12	**Andiparos**	Compass Security AG	Java 1.5x	Linux, Mac OS X Windows
13	**Watabo**	Andreas Schmidt	Ruby 1.8x	Linux, Mac OS X, Backtrack, Windows

Copyright (c) 2014 Advances in Computer Science: an International Journal. All Rights Reserved.

Table 3-Open-source scanners compare [39]

	A	B	C	D	E	F	G	H	I	J
1	✓	V.S	V.S	V.ST	F	✓	×	×	×	✓
2	×	S	C[1]	ST	F	✓	×	✓	✓	✓
3	×	C	C	FG	F	✓	×	×	✓	✓
4	✓	V.S	V.S	UST[3]	SL[2]	✓	✓	×	✓	✓
5	✓	S	C	ST	F	✓	✓	×	×	✓
6	✓	S	S	ST	F	✓	✓	×	×	×
7	✓	V.S	S	ST	SL	✓	✓	×	✓	✓
8	✓	C	C	FG[4]	SL	✓	✓	×	✓	✓
9	✓	V.S	V.S	V.ST	F	✓	×	×	×	✓
10	✓	C	S	ST	SL	×	✓	✓	✓	✓
11	✓	V.S[7]	V.S	V.ST[6]	F[5]	✓	✓	×	✓	✓
12	✓	V.S	S[8]	ST	F	✓	✓	×	✓	✓
13	✓	V.S	V.S	UST	F	×	✓	×	✓	×

A: GUI
B: Configuration
C: Usability
D: Stability
E: Performance
F: Spider
G: Manual crawl
H: File analysis
I: Logging
J: Report

3.2.2. Commercial vulnerability scanners

Commercial vulnerability scanners are developed by various organizations. Although they cost too much, they have fewer security bugs compared to open source scanners. Because of competition between different organizations for developing scanners, their limitations are not disclosed. Table 4 introduces some popular commercial scanners.

[1] Complex
[2] Slow
[3] UnStable
[4] Fragile
[5] Fast
[6] Very Stable
[7] Very Simple
[8] Simple

Table 4- Commercial scanners overview [40]

	Scanner	Features
1	**Acunetix**	Low false positive rate, possible reform application security
2	**N-Stalker**	Set of security programs to evaluate the application security, possibility of define evaluation policies
3	**Netsparker**	Focus on reducing the false positive, first scanner without false positive
4	**Burp Scanner**	An integrated platform for attacking and testing web applications, operate in passive/active mode, manual/live scan mode
5	**Rational AppScan**	advanced web application security scanning, results with the Results Expert wizard
6	**HP WebInspect**	fast scanning capabilities, broad security assessment coverage, accurate web application security scanning result
7	**NTOSpider**	identifies application vulnerabilities, ranks threat priorities, graphical HTML reports

Table 5- Commercial scanners compare [38]

Scanner	A	B	C	D	E	I	J
1	✓	V.S	V.S	V.ST	F	✓	✓
2	✓	V.S	V.S	V.ST	V.F	×	✓
3	✓	V.S	V.S	V.ST	F	✓	✓
4	✓	V.S	V.S	ST	V.F	✓	✓
5	✓	S	S	V.ST	F	✓	✓
6	✓	V.S	V.S	S	F	✓	✓
7	✓	V.S	V.S	S	F	✓	×

3.3. Academic research about web scanners

Articles in the field of penetration testing are analyzed in three groups. A group of articles compared commercial and open source scanners and tested them against some vulnerability. A second group is articles that provide a new method or tool for automatic penetration tests. Moreover, for assessment of security methods and tools, we require vulnerable applications whose vulnerabilities are clear. The third group of studies is devoted to these applications.

3.3.1. Comparison of existing tools and methods

Numerous articles reviewed and compared web scanners. In some of these papers, the comparison of tools is the main goal, and some others compared tools with the goal of providing a new tool or technique. Most vulnerabilities considered in these comparisons are SQL and XSS injection. Many of these articles suggest that available tools cannot identify all vulnerabilities and some others concluded that false positive is high.

Copyright (c) 2014 Advances in Computer Science: an International Journal. All Rights Reserved.

Table 2 and Table 3 present the scanners tested in each article, vulnerabilities and the test environment.

In [10], the name of tools has not been mentioned for commercial reasons and the neutrality of the article.

In [11], results indicate that different tools produce different results, cannot identify many vulnerabilities and have about 20% to 70% false positive.

The comparison by McAllister et al. (2008) shows that tools have a low ability to detect vulnerability, however, using the proposed technique, more vulnerabilities were found [12].

In [13], the authors, due to the detection rate of vulnerabilities, claimed that their test suit is effective for different tools. They also stated that none of the tools are capable of identifying level 2 or higher vulnerabilities.

Shelly (2010) concluded that the use of a test environment with secure and insecure versions is a good way to find the reasons of producing false positive and false negative by tools. The conclusion declared on the quality of tools indicates that tools can detect simple XSS and SQL injection. But to identify non-simple XSS and SQL injection, session management flows, running malicious files and buffer overflow, more work is needed to improve techniques and tools [14].

In [15], it is noted that tools require a greater understanding of active contents and scripting languages Like SilverLight, Flash, Java Applet and JavaScript.

In [16], a new tool called CIVS-WS[9] is developed with a new method to identify SQL/XPath injection. It came to the conclusion that the implemented tool has the 100% coverage power and 0% false positive.

The paper [17] is based on the results obtained in [10]. The author proposed a method to identify SQL injection and developed a tool called VS.WS. To test this method, test [10] was repeated. All tools were executed against 262 public web services and Java implementation from 4 web services specified By TPC-APP benchmark. It concluded that the implemented tool performance in terms of coverage and false positive is better than commercial tools.

In [18], it is concluded that crawling in an advanced web application is a serious challenge for penetration test tools and they require supporting technologies such as Flash and Java, more advanced algorithms for performing deep crawling, and tracking the state of the application under test and more studies on automating the identification of vulnerabilities in the application logic.

In [19], the authors concluded that even when scanners are taught to exploit vulnerabilities, they cannot detect stored SQL injection. They also state that improving some of the functionality of black-box scanners like state full scanning, input selection by field name and tag, the novelty of attack vector, server response analysis and post scanning can improve the discovery rate.

Reference [20], which is a generalization of [15, 18], studied three scanners and concluded that scanners cannot detect vulnerabilities because of weaknesses in the third phase.

In [21], it is concluded that Iron WASP, NetSparker community edition, OWASP ZAP, Vega, N-Stalker and W3AF identified highest to lowest number of vulnerabilities and had the lowest to highest false negative, respectively.

Based on the results of [22], W3AF, Archani and Skipfish were chosen as the best. It was also shown that the most difference in scanners is in identifying injection vulnerabilities, cross-site scripting, session management and broken authentication.

Table 2-Scanners evaluation summery1

[]	Scanners	Used test suits
[10]	HP WebInspect, IBM Rational AppScan, Acunetix Web Vulnerability Scanner	
[11]	3 commercial scanners.	MyReferences, BooksStore Online
[12]	Brup Spider, w3af, Acunetix Web Vulnerability scanner free edition	Django-basic-blog, Django-forum

[9] Command Injection Vulnerability Scanner for Web Services

Copyright (c) 2014 Advances in Computer Science: an International Journal. All Rights Reserved.

[13]	4 commercial and open-source scanner	An online banking application
[14]	Grendel-Scan, Wapiti, w3af, Hailstorm, N-stalker, Netsparker, Acunetix Web Vulnerability Scanner, Brup Scanner	BuggyBank , Hokie Exchange
[15]	Acunetix Web Vulnerability Scanner, Cenzic Hailstorm Pro,HP WebInspect, IBM Rational AppScan, Mcafee SECURE, N-Stalker QA Edition, QualysGuard PCI, Rapid7 NeXpose	Drupal, phpBB, WordPress, customized test bed
[16]	CIVS-WS[10], IBM Rational AppScan, Acunetix Web Vulnerability Scanner, VS.BB	A prototype tool with 9 services with multiple operations
[17]	VS.WS	Repeat [16] evaluation test
[18]	Acunetix Web Vulnerability Scanner , IBM Rational AppScan, Burp scanner, Grendel-Scan, Hailstorm, Milescan, N-Stalker , NTOSpider, Paros, w3af , HP WebInspect	WackoPicko
[19]	Acunetix WVS, IBM Rational AppScan Enterprise , QualysGuard Express Suite	PCI, WackoPicko, MatchIt
[20]	OWASP ZAP, N-Stalker WVS, Acunetix WVS,IBM Rational AppScan	PCI, WackoPicko, SimplifiedTB
[21]	Iron WASP ,W3AF ,N-Stalker , NetSparker Community Edition ,Vega and OWASP ZAP	WackoPicko
[22]	IronWASP, ZedAttackProxy(ZAP), SQLmap, W3AF,arachni,Skipfish, Watobo, VEGA, Andiparos, ProxyStrike, Wapiti , ParosProxy, GrendelScan, PowerFuzzer,Oedipus, UWSS(UberWebSecurityScanner), Grabber, WebScarab, MiniMySQLat0r, WSTool, Crawlfish, Gamja, iScan, DSSS(DamnSimpleSQLiScanner), Secubat, SQID(SQLInjectionDigger), SQLiX, Xcobra	

Table 3- Scanners evaluation summery2

[]	Checked vulnerabilities
[10]	SQL injection, Xpath injection, Code execution, Possible parameter based buffer overflow, possible username or password disclosure, possible server path disclosure
[11]	SQL injection, XSS
[12]	stored XSS, reflected XSS
[13]	XSS, SQL injection, blind SQL injection, file inclusion
[14]	SQL injection, XSS, Buffer overflow, Session management flaws, Malicious file execution
[15]	Different type of XSS, Different type of SQL injection, Cross-Channel Scripting, session management flaws, CSRF, SSL/Server Configuration, Information Leakage
[16]	SQL injection, XPath injection
[17]	SQL injection
[18]	Different type of XSS, SQL injection, command-line injection, file inclusion , file exposure
[19]	Different type of SQL injection(specially stored SQL injection)
[20]	Stored XSS
[21]	reflected SQLI, stored SQLI, reflected XSS, stored XSS, reflected XSS behind JavaScript, reflected XSS behind flash , predictable session ID, command line injection, file inclusion, file exposure, parameter manipulation, directory traversal, logic flow, forceful browsing , weak passwords
[22]	OWASP TOP 10

[10] Command Injection Vulnerability Scanner for Web Services

In this section, we reviewed several papers that compared different scanners. As seen in Table 7, vulnerabilities addressed in Owasp Top 10 were considered in these papers. To enhance the value of assessments, most articles used popular scanners Like Acunetix Web Vulnerability Scanner, IBM Rational AppScan, and Burp scanner. Table 8 summarizes the number of using each scanner in articles.

Table 4- Frequency of used scanners in papers

Scanners	Used in papers
(1) Acunetix Web Vulnerability Scanner	8
(2) IBM Rational AppScan	6
(3) w3af, (4) N-stalker	5
HP WebInspect , Brup Spider, Grendel-Scan, Hailstorm, Wapiti, OWASP ZAP	3
Netsparker, VS.BB, Paros, Iron WASP, Vega	2
other	1

Table 5 lists attacks to different vulnerabilities and Table 8 lists scanners that were investigated in the compared articles more than others and shows that they cover what percentage of cases in Table 6(those not listed in the table are covered by all scanners) [37].

Table 5- Attacks

#	Attack	#	Attack
1	Error based SQL injection	**18**	Format String Attack
2	Blind SQL injection	**19**	Code Injection
3	Server Side Java Script injection	**20**	XML Injection
4	Reflected Cross Site Scripting	**21**	Expression Language Injection
5	Persistent Cross Site Scripting	**22**	Buffer Overflow
6	DOM Cross Site Scripting	**23**	Integer Overflow

Copyright (c) 2014 Advances in Computer Science: an International Journal. All Rights Reserved.

7	JSON Hijacking	**24**	Source Code Disclosure
8	Path Traversal & Local File Inclusion	**25**	Old, backup and Unreferenced Files
9	Remote File Inclusion	**26**	Padding Oracle
10	Command Injection	**27**	Forceful Browsing/ Authentication Bypass
11	Unrestricted File Upload	**28**	Privilege Escalation
12	Open Redirect	**29**	Xml External Entity
13	CLRF injection	**30**	Weak Session Identifier
14	LDAP Injection	**31**	Session Fixation
15	Xpath injection	**32**	Cross Site Request Forgery
16	SMTP/IMAP/EMAIL Injection	**33**	Application Denial of service
17	Server-Side Includes Injection		

Table 6- Compare scanners

	3	**5**	**6**	**7**	**11**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	
(1)				×					×	×		×	
(2)	×												
(3)	×			×							×	×	
(4)	×	×	×	×	×	×	×	×	×	×	×	×	
	21	**22**	**23**	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**	**%**	
(1)		×	×			×		×				75.7	
(2)	×			×								90.9	
(3)	×		×	×		×	×		×			69.6	
(4)	×	×	×	×	×	×	×	×	×	×	×	30.3	

Since XSS and SQL injection vulnerabilities are among common vulnerabilities, they were reviewed in the majority of articles.

3.3.2. New methods and tools

This section reviews a number of papers that designed a new scanner.

In [23], a tool was design with static analysis capabilities called Pixy to identify vulnerabilities in web applications. Pixy is an open source tool and its main purpose is to identify cross-site scripting vulnerability in PHP scripts. PHP was selected because it is widely used in designing web applications and by a large number of security consultants in PHP applications. Vulnerability detection was based on data flow analysis. To evaluate this tool, six open source PHP applications were used. In PhpNuke, PhpMyAdmin and Gallery, 36 known vulnerabilities were recreated with 27 false positives. In Simple PHP Blog, Serendipity and Yapig, 15 unknown vulnerabilities were identified with 16 false positives. A restriction of Pixy is that it does not support 'OO' (object orientation).

Saner is a tool developed by combining static and dynamic techniques to identify errors in applications in PHP language [24]. It uses static techniques to identify inputs in each path with the help of string modeling methods. The second technique is dynamic analysis which operates bottom-up. It first recreates codes to identify inputs and then uses a large collection of malicious inputs to identify exploitable flows. The dynamic phase aims to test all the application paths that were identified suspicious in the static phase. Since the static analysis phase is prudent, it is possible to produce false positive which must be manually evaluated which is a boring process. The purpose of automatic dynamic analysis is to automate this process or at least automate detection of what vulnerability input should be used. For assessment, this tool was evaluated on five applications of Jetbox, MyEasyMarket, PBLGuestbook, PHP-Fusion and Sendcard which are developed in PHP. Although this tool has not a very good execution performance, the time required to analyze most applications is good.

In [25], the most common input validation vulnerability model called Tainted Mode Model is used to identify internal vulnerabilities. This paper improves the classical tainted mode model to investigate internal data flows. It also introduces a new method using information obtained from dynamic analysis for doing the automated penetration test. Given a larger view it gives from the application, its accuracy will be more, and the accuracy of input validation procedures can be tested. Using improved tainted mode model, applications were modeled as follows:

W: (Scheme, Req × State →DDG × Resp × {query } × State)

Where scheme is a set of relational data schemes which shows the application database. Req is the http request sent to the application. State, on the left, refers to the application state which includes contents of the application environment (such as a database, system files and LDAP). DDG = (V, E) is a data

Copyright (c) 2014 Advances in Computer Science: an International Journal. All Rights Reserved.

dependency graph which indicates the execution path and data flow obtained by the application to process the received request. Resp is the response returned by the application. {query} is a set of queries from the database generated by the application when processing a request. State, on the right, is the state to which the application transfers.

The implemented approach has three main components: dynamic analysis module that gathers the effects of application execution. Analyst that produces DDGs for gathered effects and the penetration test module that enters normal or malicious inputs into the application. For assessment, three applications in Python (Test application, Spyce and Trac) were used. In the presented results, false positive rate is zero.

In [26], a scanner is proposed to identify injection vulnerabilities. This system analyzes websites, aiming at automatically finding SQL and XSS injection vulnerabilities. The proposed system consists of two components: spider and scanner. The spider is used to navigate the site and find input points. The scanner initiates injection test and response analysis and consists of two parts: response analyst and author of rules. The system was run in VMware work station ACE with two hosts, one for the defense server and the other for the web server. The system was designed with PHP5 and MySQL and used the cURL module to execute attacks. Seven applications from National Vulnerability Database (NVD)[11] were selected for assessment. Finally, the designed scanner was compared with some other scanners. It concluded that this system is effective, and vulnerability detection based on input points definitely can find vulnerabilities.

BLOCK is a black-box approach designed in [27] to identify state violation attacks based on the WebScrab tool. In this paper, the application is considered as a stateless system, and the application behavior model is obtained from the interaction between client and application, i.e. the relationships between web requests, responses and session variables. BLOCK has two key phases to identify state violation attacks: in the training phase, the desired behavior model is obtained by observing web request/response sequence, and the variable values corresponding to the session during its running without attacks. In the identification phase, the obtained model is used to evaluate each incoming web request and outgoing response, and any violation is identified. To assess this tool, the applications of Scarf, Simplecms, BloggIt, WackoPicko and OsCommerce were used. Results show that this method is effective in identifying state violation attacks and is incurred little overhead. Since this method is independent of the application code and can be used for a large number of web applications with different frameworks, it is of particular importance.

In [28], a mealy state machine is used to model web applications for management of requests that change the application mode. To detect a state change in this model, difference in the returned response in case of identical sent requests was considered. To implement this method, htmlUnit and the fuzzer of w3af tool were used. In the state graph the tool generates, nodes represent states and edges represent the requests sent to the application. Using the graph coloring problem, this article combined similar states. For evaluation, this product was studied with wget, w3af and skipfish scanners on the Gallery application, two versions of PhpBB, Scraf, vanilla forums, WackoPicko and three versions of wordPress. The evaluation results indicate that the designed scanner not only can run more codes than web applications, but also it can identify vulnerabilities not identified by other scanners. Due to the use of HtmlUnit, despite using the w3af fuzzer tool, this tool has a less false positive than other tools. Among the limitations of this tool, one can point to the lack of support of AJAX and applications that can be used publicly because users may influence the state-change detection algorithm.

In [29], an automatic black-box tool is presented to identify reflected XSS and stored XSS vulnerabilities in web applications. The tool uses user interactions to do the test more effectively. First, user interactions are recorded, then changes are made on these interactions for attacks and finally, this transaction is re-executed on the system. To evaluate the

[11] A set of standards based on vulnerability management data owned by the US government which uses the Security Content Automation Protocol (SCAP).

Copyright (c) 2014 Advances in Computer Science: an International Journal. All Rights Reserved.

performance, the tool was compared with Spider, Brup Spider, w3af and Acunetix on three applications of the Django framework from different aspects. Results show that the proposed method can identify more bugs than the listed commercial and open source tools.

In [30], a tool called MiMoSA is presented based on static analysis for PHP applications. The paper divides multi-module into two groups (data flow and workflow) and input states into two categories (server side and client side). The analysis intended to MiMoSA has two phases: intra-module which reviews each module of the application on its own, and inter-module which considers the whole application. The intra-module analysis aims to summarize each module of the application by defining preconditions, post-conditions and sinks. All links in each module are also extracted. This phase is dependent on the programming language. This information is then used for inter-module analysis to obtain the future state of the application workflow. In the inter-module analysis phase, results from intra-module analysis are put together in a single graph which models the future work flow of the whole application. Then a model survey technique is used to identify data flow vulnerabilities and future workflow defects. This phase is independent of the programming language used in the application development. To assess the tool, three applications of Aphpkb, BloggIt, MyEasyMarket, Scarf and SimpleCms, developed in the PHP language, were used. Evaluation results show that MiMoSA can identify all known vulnerabilities and also discover some new vulnerabilities. In evaluating the tool, only one false positive was seen. The number of states the tool considers is more than the number of actual states in the application code. This problem occurs for two main reasons: first, in MiMoSA, states may be produced correspond to paths that cannot be run in the application and second, the possibility of repetitive states in the tool with aligned but different conditions.

In [31], first the dynamic analysis and observing the application performance are used to understand the application's behavioral characteristics. Then the found characteristics are filtered to reduce false positive. The symbolic evaluation model is used on inputs to identify the application paths which violate these conditions. This paper focuses on logical vulnerabilities. The provided tool is Waler which is used for servlet-based application developed in Java. To assess the tool, 12 applications were used. According to this article, Waler is the first tool that can detect logical processes in applications automatically without human intervention.

There are many different tools for implementing each of the penetration test steps. In [32], for saving time, some of these tools were combined including theHarvester, Metagoofile, ZAP, NMAP, Nessus and Metasploit. The language used in this article is Python. The function of proposed method can be explained in three phases: information gathering, analysis of the information obtained and the use of this information to find possible vulnerabilities. First, the tools theHarvester, Metagoofile and NMAP are run as information gathering tools and Nessus and ZAP as two scanners, and information obtained by each tool is stored in a separate file. Then the results of ZAP and Nessus are analyzed and the resulting output file is used for implementing the attack phase on Metasploit.

The section addresses 10 studies that proposed a new method or tool for finding vulnerabilities in web applications. Each study is based on static or dynamic analysis, or a combination of the two. A summary of these characteristics is given in Table .

Table 11- papers proposed tools and methods summery

[]	Year	Static analysis	Dynamic analysis	Description
[23]	2006	✓		cross-site scripting detection
[24]	2008	✓	✓	identify errors in applications by path input detection
[25]	2008		✓	Use TDM to identify internal vulnerabilities
[26]	2010		✓	XSS and injection vulnerabilities detection
[27]	2011	✓	✓	identify state violation attacks based on the WebScrab tool
[28]	2012	✓	✓	Use state machine to model web application for detect more vulnerabilities
[29]	2008		✓	reflected XSS و stored

Copyright (c) 2014 Advances in Computer Science: an International Journal. All Rights Reserved.

				XSS detection
[30]	2007	✓		Code analysis to models the future work flow of the application
[31]	2010		✓	logical vulnerabilities detection
[32]	2013	✓		Combine different tools to save time

3.3.3. Design of test environments

Vulnerable web applications are used to web vulnerability scanners evaluation. In this part we will have a review on some of this applications like Damn Vulnerable Web App(DVWA)[33], OWASP WebGoat[34], WackoPicko[18] and BodgeIt[35] that are used to training courses and numerous articles.

Damn Vulnerable Web App (DVWA) is a PHP/MySQL web application that is damn 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. Brute Force Login, Command Execution, CSRF, File Inclusion, SQL Injection, Upload Vulnerability and XSS are the vulnerabilities exist in DVWA[33].

Webgoat is a J2EE based web application designed by OWASP to introduce common security flaws in the web applications. The following categories are available in Webgoat : Access Control Flaws, AJAX Security, Authentication Flaws, Buffer Overflows, Code Quality, Concurrency, Cross-Site Scripting (XSS), Improper Error Handling, Injection Flaws, Denial of Service, Insecure Communication, Insecure Configuration, Insecure Storage, Malicious Execution, Parameter Tampering, Session Management Flaws, Web Services, and Admin Functions [34].

WackoPicko is a vulnerable website that designed by Adam Doupé and used in [18] for the first time. Vulnerabilities of this application are reflected XSS, stored XSS, SessionID vulnerability, stored SQL injection, reflected SQL injection , Directory Traversal, multi-step stored XSS, forceful browsing, Command-line Injection, File Inclusion, Parameter Manipulation, Reflected XSS Behind JavaScript, Logic Flaw, Reflected XSS Behind a Flash Form and Weak username/password.

BodgeIt is a web based vulnerable application designed by Simon Bennett with learning goals for pentesters. This web application consist of cross site scripting, sql injection , hidden(but unprotected) content, cross site request forgery, debug code, insecure object references و application logic vulnerabilities[35].

Table 7- Test bed vulnerabilities

	DVWA	WebGoat	WackoPicko	BodgeIt
Brute Force	✓		✓	
CSRF	✓			✓
File Inclusion	✓		✓	
SQL Injection	✓	✓	✓	✓
Access Control Flaws		✓		
XSS	✓	✓	✓	✓
Buffer Overflows		✓		
Logic vulnerabilities			✓	✓
AJAX Security		✓		

4. Conclusion

The present paper reviewed studies in the field of penetration test, especially web penetration test. Manual penetration test is not effective in terms of time and money, so its automatic version is considered. For performing the automatic web penetration test, web scanners are used. They first crawl the target, then attack to the results of the previous phase and finally report vulnerabilities in the target. In this paper, we examined research in the field of web penetration test in three categories: articles that compared and analyzed available scanners, articles that proposed a new method or tool for penetration test and articles that proposed a test environment to test different tools. According to papers that analyzed various scanners, the Acunetix Web Vulnerability Scanner and IBM Rational AppScan scanners and the SQL injection and XSS vulnerabilities were considered more than others. We also reviewed 10 studies that proposed a new tool or method for penetration test, some of which were based on the dynamic analysis, some on the static analysis and some on a combination of the two. To

Copyright (c) 2014 Advances in Computer Science: an International Journal. All Rights Reserved.

evaluate any method or tool in the field of penetration test, we require test environments. Four test environments were introduced in the final section.

The problems in existing scanners include the lack of support of attacks like stored sql and stored XSS that need to several steps to complete the attack, the lack of support of new technologies and vulnerabilities related to application logic flows. It is hoped that future work will consider these items.

References

[1]http://searchnetworking.techtarget.com/tutorial/Network-penetration-testing-guide

[2] Samant, N. (2011). Automated penetration testing (Doctoral dissertation, San Jose State University).

[3]http://www.coresecurity.com/comparing-security-testing-options

[4]http://www.c2networksecurity.com/

[5] Engebretson, P. (2013). The Basics of Hacking and Penetration Testing: Ethical Hacking and Penetration Testing Made Easy. Elsevier.

[6] Scarfone, K., Souppaya, M., Cody, A., & Orebaugh, A. (2008). Technical guide to information security testing and assessment. NIST Special Publication, 800, 115.

[7] Alisherov, F., & Sattarova, F. (2009). Methodology for penetration testing. InInternational Journal of Grid and Distributed Computing.

[8] Herzog, P. (3). The Open Source Security Testing Methodology Manual. 2010.ISECOM.[Links].

[9] Doupé, A. L. (2014). Advanced Automated Web Application Vulnerability Analysis (Doctoral dissertation, UNIVERSITY OF CALIFORNIA Santa Barbara).

[10] Vieira, M., Antunes, N., & Madeira, H. (2009, June). Using web security scanners to detect vulnerabilities in web services. In Dependable Systems & Networks, 2009. DSN'09. IEEE/IFIP International Conference on (pp. 566-571). IEEE.

[11] Fonseca, J., Vieira, M., & Madeira, H. (2007, December). 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 (pp. 365-372). IEEE.

[12] McAllister, S., Kirda, E., & Kruegel, C. (2008, January). Leveraging user interactions for in-depth testing of web applications. In Recent Advances in Intrusion Detection (pp. 191-210). Springer Berlin Heidelberg.

[13] Fong, E., Gaucher, R., Okun, V., & Black, P. E. (2008, January). Building a test suite for web application scanners. In Hawaii International Conference on System Sciences, Proceedings of the 41st Annual (pp. 478-478). IEEE.

[14] Shelly, D. A. (2010). Using a Web Server Test Bed to Analyze the Limitations of Web Application Vulnerability Scanners (Doctoral dissertation, Virginia Polytechnic Institute and State University).

[15] Bau, J., Bursztein, E., Gupta, D., & Mitchell, J. (2010, May). State of the art: Automated black-box web application vulnerability testing. In Security and Privacy (SP), 2010 IEEE Symposium on (pp. 332-345). IEEE.

[16] Antunes, N., Laranjeiro, N., Vieira, M., & Madeira, H. (2009, September). Effective detection of SQL/XPath injection vulnerabilities in web services. InServices Computing, 2009. SCC'09. IEEE International Conference on (pp. 260-267). IEEE.

[17] Antunes, N., & Vieira, M. (2009, September). Detecting SQL injection vulnerabilities in web services. In Dependable Computing, 2009. LADC'09. Fourth Latin-American Symposium on (pp. 17-24). IEEE.

[18] Doupé, A., Cova, M., & Vigna, G. (2010). Why Johnny can't pentest: An analysis of black-box web vulnerability scanners. In Detection of Intrusions and Malware, and Vulnerability Assessment (pp. 111-131). Springer Berlin Heidelberg.

[19] Khoury, N., Zavarsky, P., Lindskog, D., & Ruhl, R. (2011, October). An analysis of black-box web application security scanners against stored SQL injection. InPrivacy, security, risk and trust (passat), 2011 ieee third international conference on and 2011 ieee third international conference on social computing (socialcom) (pp. 1095-1101). IEEE.

[20] Alassmi, S., Zavarsky, P., Lindskog, D., Ruhl, R., Alasiri, A., & Alzaidi, M. An Analysis of the Effectiveness of Black-Box Web Application Scanners in Detection of Stored XSSI Vulnerabilities.

Copyright (c) 2014 Advances in Computer Science: an International Journal. All Rights Reserved.

[21] Suteva, N., Zlatkovski, D., & Mileva, A. (2013). Evaluation and Testing of Several Free/Open Source Web Vulnerability Scanners.

[22] SAEED, F. A., & ELGABAR, E. A. (2014). ASSESSMENT OF OPEN SOURCE WEB APPLICATION SECURITY SCANNERS. Journal of Theoretical and Applied Information Technology, 61(2).

[23] Jovanovic, N., Kruegel, C., & Kirda, E. (2006, May). Pixy: A static analysis tool for detecting web application vulnerabilities. In Security and Privacy, 2006 IEEE Symposium on (pp. 6-pp). IEEE.

[24] Balzarotti, D., Cova, M., Felmetsger, V., Jovanovic, N., Kirda, E., Kruegel, C., & Vigna, G. (2008, May). Saner: Composing static and dynamic analysis to validate sanitization in web applications. In Security and Privacy, 2008. SP 2008. IEEE Symposium on (pp. 387-401). IEEE.

[25] Petukhov, A., & Kozlov, D. (2008). Detecting security vulnerabilities in web applications using dynamic analysis with penetration testing. Computing Systems Lab, Department of Computer Science, Moscow State University.

[26] Chen, J. M., & Wu, C. L. (2010, December). An automated vulnerability scanner for injection attack based on injection point. In Computer Symposium (ICS), 2010 International (pp. 113-118). IEEE.

[27] Li, X., & Xue, Y. (2011, December). BLOCK: a black-box approach for detection of state violation attacks towards web applications. In Proceedings of the 27th Annual Computer Security Applications Conference (pp. 247-256). ACM.

[28] Doupé, A., Cavedon, L., Kruegel, C., & Vigna, G. (2012, August). Enemy of the State: A State-Aware Black-Box Web Vulnerability Scanner. In USENIX Security Symposium (pp. 523-538).

[29] McAllister, S., Kirda, E., & Kruegel, C. (2008, January). Leveraging user interactions for in-depth testing of web applications. In Recent Advances in Intrusion Detection (pp. 191-210). Springer Berlin Heidelberg.

[30] Balzarotti, D., Cova, M., Felmetsger, V. V., & Vigna, G. (2007, October). Multi-module vulnerability analysis of web-based applications. In Proceedings of the 14th ACM conference on Computer and communications security (pp. 25-35). ACM.

[31] Felmetsger, V., Cavedon, L., Kruegel, C., & Vigna, G. (2010, August). Toward automated detection of logic vulnerabilities in web applications. In USENIX Security Symposium (pp. 143-160).

[32] Haubris, K. P., & Pauli, J. J. (2013, April). Improving the Efficiency and Effectiveness of Penetration Test Automation. In Information Technology: New Generations (ITNG), 2013 Tenth International Conference on (pp. 387-391). IEEE.

[33] http://www.dvwa.co.uk/

[34]https://www.owasp.org/index.php/Category:OWASP_WebGoat_Project

[35] https://code.google.com/p/bodgeit/

[36]https://www.owasp.org/index.php/Top_10_2013-Top_10

[37] http://www.sectoolmarket.com/

[38]http://www.sectoolmarket.com/general-features-comparison-unified-list.html#Glossary

[39] Khari, M., Singh, N. (2014, May). An Overview of Black Box Web Vulnerability Scanners. In Computer Science and Software Engineering, 2014 International Journal of Advanced Research.

[40] Shelly, D. A. (2010). Using a Web Server Test Bed to Analyze the Limitations of Web Application Vulnerability Scanners (Doctoral dissertation, Virginia Polytechnic Institute and State University).

[41] Arefzadeh, A.(2012). Representing a penetration test plan for web-applications , based on RUP test plan(Master thesis, Department of Information Technology Engineering, Maleke-ashtar University of Technology)

[42] Marvari, S. (2012). Presentation An Improved Structure of Methodology of Penetration Test. (Master thesis, Department of Information Technology Engineering, Maleke-ashtar University of Technology

Copyright (c) 2014 Advances in Computer Science: an International Journal. All Rights Reserved.