A Model for Conducting Security Assessment within an Organisation

Nor Fatimah Awang
Faculty of Defence Science and Technology
National Defence University of Malaysia
Kuala Lumpur, Malaysia
e-mail: norfatimah@upnm.edu.my

Azizah Abd Manaf
Advanced Informatics School (UTM AIS)
UTM International Campus
Kuala Lumpur, Malaysia
e-mail: azizaham.kl@utm.my

Abstract—**Security Assessment is widely used to audit security protection of web applications. However, it is often performed by outside security experts or third parties appointed by a company. The problem appears when the assessment involves highly confidential areas which might impact the company's data privacy where important information may be accessed and revealed by the third party. Even though the company and third party might have signed a non-disclosure agreement, it is still considered a high risk since confidential information on infrastructure and architecture are already exposed. It is important to keep the confidential information within the project team members to protect the data used by the system. Therefore, this paper proposes a model to conduct internal security assessment to ensure all organisational assets are protected and secured. The main objective of this paper is to discuss the activities and processes involved in conducting the security assessment.**

Keywords-Web application; vulnerability; security testing; security assessment; penetration testing.

I. INTRODUCTION

Today, more than one billion people worldwide use the Internet in their daily routine for a variety of reasons, such as communicating with others, conducting research, shopping, banking and electronic commerce [1]. Due to the high usage of the Internet in today's highly competitive world, more organisations are relying solely on web-based applications and the Internet to change their daily manual activities to online-based activities. Most of the organisations have shifted to the Internet to make more profits and at the same time to increase the efficiency of their activities such as customer support services, data transactions and quality of information supply [2]. From businesses, industries, governments to non-profit organisations, the Internet has simplified a lot of business processes and activities. The growth of internet applications gave a high impact and created business opportunities to the organisations. However, the Internet has also brought unintended consequences, such as criminal activities, spamming, credit card frauds, online fraud, theft of sensitive information, phishing and other related cyber-crimes [3][4]. According to surveys conducted by the security firm McAfee and the Center for Strategic and International Studies, millions of dollars have been lost due to cyber-crime attacks [3]. In fact, Symantec Group reported, attacks against web applications have increased in 2010 by 93% compared to 2009. Another report showed that, almost 150,000 new sites are registered per day on the internet, which has the potential to introduce around two billion serious vulnerabilities [5].

There are numerous researches that focused on the issues of web application security and vulnerability. Many of the studies provide models, methodologies and technologies to enhance the security in web applications. One important step to ensure web application security is to conduct security assessment periodically. Security assessment is a process to search for potential loopholes or vulnerabilities contained in a system. Through the security assessment, an organisation can then assure that systems and applications are operating effectively in providing appropriate product or service confidentiality, integrity and availability [6]. The assessment is important to make sure all systems are secure and all vulnerabilities are discovered before any system is being deployed [7][8]. Some companies choose to use consultants or outsource security assessments to third-parties. Outsourcing security assessment is mandatory in security audit for banking and online business industries, therefore a software industry for any related business can just concentrate on developing their system and let the third-party evaluate their product before releasing it to the market. However, according to a study conducted by Corwill and Nasimmbeni, there are some security issues involved when using external party to conduct an assessment [10][11]. Even though a non-disclosure agreement has been signed by both parties to prevent them from divulging information, it is still considered a high risk as the third-party already has the confidential infrastructure and architecture information. It is therefore important to keep the internal information within project members to protect the confidential data used by the system.

This paper discusses a model for conducting security assessment and detecting vulnerabilities that exist in web applications. Security assessment is a process to find potential security loopholes or vulnerabilities in target systems. Using this model, many organisations will have the opportunity to perform security assessment internally without having to outsource it to third-party security experts.

The structure of this paper is as follows. Section II briefly describes the background of web application architecture and discusses related techniques which are commonly used in detecting vulnerabilities in web applications. Next, Section III discusses in detail the proposed model. Section IV discusses the results and finally, Section V presents the conclusion.

II. BACKGROUND AND RELATED WORK

A. Web Application Architecture

Since a web application runs in the dynamic and distributed environment that is different from the traditional programmes, hence more vulnerability exists. This section gives some explanations on the architecture of web applications and several common vulnerabilities which exist in web applications. In general, a web application has three tier constructions as shown in Figure 1 [12][13]. Figure 1 describes the architecture of a web application. The architecture of a web application consists of web browser, web server, web application and database server. In Tier 1, web server receives input and interacts with clients through web browser by using http or http protocol. The web applications are developed using different programming language such as Active Server Page (ASP), Common Gateway Interface (CGI), Ruby or Java in Tier 2. Generally, the web server will manage the page requested from the web client by sending the request to the application server and the application server constructs codes dynamically and passed the codes back to the web server. The flow of data amongst the tiers gives rise to input validation problem for the web application server; it must check and/or modify incoming the input before processing it further or incorporating the input into output that it passes to other tiers to execute. Failure to check or sanitise the input appropriately can compromise the web application's security [14]. Similarly, Tier 3 is responsible for the access of authenticated users and rejection of malicious users from the database.

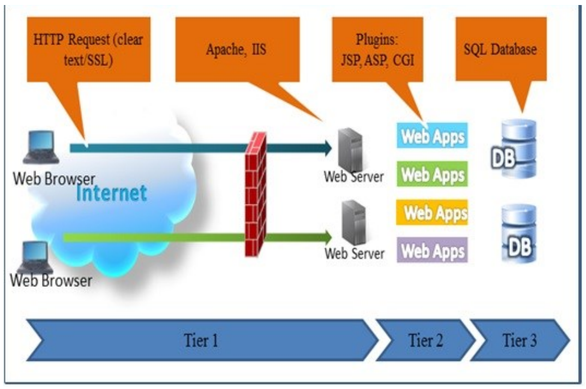


Figure 1. Web Application Architecture

B. Web Application Vulnerabilities

In this paper, the definition of web application vulnerabilities follows the definition from the Open Web Application Security Project (OWASP) [18], which defines vulnerability as a hole or a weakness in the application, which can be a design flaw or an implementation bug that allows an attacker to cause harm to the application.

There are a few web application vulnerability databases available on the Internet, e.g., OWASP Top 10 Web Application Vulnerability [18], SANS Top 20 2007 Security Risks, and WASC Threat Classification [15]. These databases classify and identify all known web application vulnerabilities and attacks, and they are continuously updated and maintained. The public security knowledge databases are very useful to testers for self-education and used as test references. With lots of vulnerabilities appearing in web applications, it is more difficult for system or network administrators to protect core assets such as personal information, confidential data and customer credit card numbers. The most common and popular vulnerabilities exploited by attackers are SQL injection and cross site scripting [16]. These are due to improper sanitisation in input validation fields. The researcher in [17] highlighted some potential vulnerabilities that will help security tester or assessor to understand possible vulnerabilities in login page that would be useful for security assessment.

The model of this study aims to identify potential vulnerabilities which exist in web applications. The goals of testers are to mimic the possible techniques commonly used by the attacker to attack the system, identify possible vulnerability based on the functionality, identify test cases to the system and find out how to exploit these attacks to improve the web application security. The testers can obtain information that help them to understand what are the function and vulnerability that are commonly used by the attacker to exploit the system by analysing the intentions of functionality and vulnerability. This paper extends the results presented in [8] and [9].

C. Techniques to Detect Vulnerability

There are many techniques for detecting vulnerabilities during the process of software development life cycle such as static code analysis, dynamic analysis and security assessment, and penetration testing. Static Analysis consists of analysis on the application source code [19]-[21]. It is performed on the source code without executing the application. This can be done manually or by using code analysis tools such as FORTIFY, Ounce or Pixy [22]. Reports are generated and presented to the developer team after reviewing the source code. Generally, it helps to catch implementation structural bugs early and it is important to know that static analysis cannot solve all security problems. There are different tools available now for this kind of test but it is not easy to find mature and well tested tools to discover all the security defects in an application. The problem is that code analysis may be difficult and may not find all security flaws because of the complexity of the code [23].

Dynamic Analysis, also known as Dynamic Testing is used to test a program by executing it in real-time [24]. Dynamic Analysis test will communicate with a web application through the web browser in order to identify potential security vulnerabilities and architectural weaknesses in the web application. The objective is to find security errors in the web application while it is running. This technique can be performed either manually or by using automated tools [25]. Automated tool provides an automatic way to search for vulnerabilities by avoiding the repetitive and tedious task of doing hundreds or even thousands of tests manually for each vulnerability type [26].

III. The Proposed Model

This section describes the phases and activities of the proposed model as shown in Figure 2. The three main phases are Data Gathering, Attacks, and Reporting. Each phase comprises of several major activities together with their flows and stages.

A. Data Gathering – Phase 1

This is the first stage in the model. There are six major activities involved in this stage. The first three activities are basically planning focused activities. In this phase, there are some items that should be highlighted and prepared such as identifying which target system that should be tested to detect vulnerability, and what type of potential threat or vulnerability that commonly exists in web applications. Additionally, questions such as how long the testing will be carried out, which methodology will be used and what restrictions or limitations need to be applied must be tackled. The test plan should also outline the tools needed to conduct the tests, as well as exploring opportunities for automated testing. Next is to find other test planning criteria as shown in Table I. The tools used for the assessment are combinations of both commercial and open source software. At least two different tools are used to perform the test to ensure accuracy of the result. Table II lists the tools used during this assessment.

The other three activities, as discussed below, are more hands-on and mostly based on the first three activities in data gathering and findings.

Scanning - This phase is more on mapping of the potential vulnerabilities detected by scanners with main system components. This activity uses the vulnerability scanner to scan the services in identifying potential loop holes and vulnerabilities in web applications.

Discovery Scanning Analysis - In this activity, results produced by different tools are compiled for further analysis.

Risk Rating - In this activity, discovery analysis findings are used as the main source and subject in risk rating. The risk rating outcomes or results are more specific to the assessed system. The findings are then categorised in Section IV into three risk levels such as high, medium and low in order to indicate the level of severity. The severity levels are based on the guidelines from OWASP and recommendation from tools. This rating is used throughout this assessment to provide common understanding of the risk.

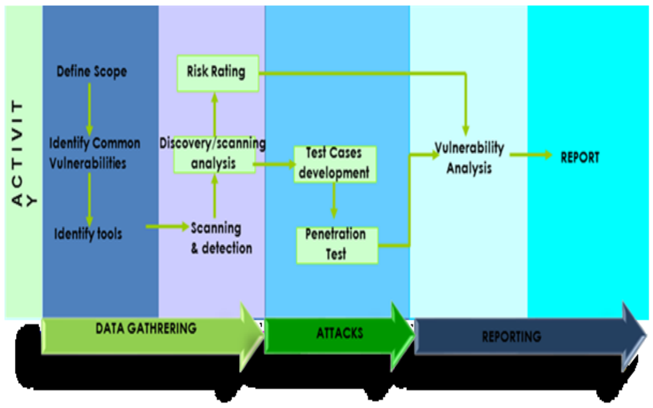


Figure 2. Model for detecting vulnerability in web applications

TABLE I. TEST PLANNING CRITERIA

Criteria	Planning Detail
No. of Security Tester and Qualification	To get the number of certified security tester and the tester unified qualification.
Type of Tools	To see if they use open source tools available on the net or commercial tools.
Number of Server	How many servers will be involved in this assessment?
Test Time Frame	How long is the duration for this assessment?

TABLE II. LIST OF TOOLS

Tools	Testing Activities
Zenmap	To get banner grabbing for server
Nessus Nexpose Burp Suite Free Edition Acunetix Web Vulnerability Scanner	During Scanning phase
Test Case generator Attack Generator	During Attack phase

B. Attacks – Phase 2

This is the second stage in the model. As the name suggests, it is responsible for performing the attacks on the system. The attacks are performed on the vulnerabilities that have been discovered during the data gathering phase. The attack phase is executed in a cascaded manner where every successful attack leads to obtaining more privileges and system information. There are two major activities involved in this phase, which are test cases development and penetration testing.

Test Cases - Structured test cases are developed based on the OWASP testing guidelines [17]. In this study, information on existing known vulnerabilities are collected and analysed to generate attack test cases. In this phase, a tool called Test Case Generator was developed to generate attack test cases. There were 1600 test cases generated to perform SQL Injection in the vulnerable web applications. Table III presents some samples of test cases which were generated by the Test Case Generator. These test cases were used to perform the attack during the penetration testing phase.

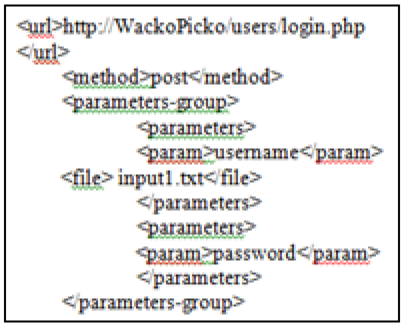
Penetration Testing – Usually, in this stage penetration testing is manually performed by a security expert to confirm the vulnerabilities detected during the scanning phase (to check false positive of the vulnerabilities). In the study model, the attack generator tool which was developed is used to automatically perform penetration testing. The attack test cases generated from the previous stage were utilised to inject and detect vulnerabilities. The fundamental objective of this Section is the design of the model that covers all steps to automate the injection attack process. This tool injects abnormal input to input parameter and discovers unexpected defects and vulnerabilities. The Attack Generator starts processing a set of target URL and target parameter. Some manual works are still required before automating the attack generator process. A Tester is needed to identify the target URL and target parameter. The test cases generated in the previous stage will be used as an input in this stage. In order to extract HTTP response and injecting them to the target system automatically, the model was developed using Apache HTTP Client API. The Attack Generator component uses input.xml (Figure 3) file to attack the target system by using POST or GET method, and it also identifies which parameter is chosen to inject the test cases. The target URL, HTTP method and parameter are chosen by the researchers.

TABLE III. TYPES OF TEST CASES

Type of Vulnerability	Test Cases
SQL Injection	' or 1=1-- " or 1=1-- ' or 1=1 /* or 1=1-- ' or 'a'='a " or "a"="a %27+OR+%277659%27%3D%277659 %22+or+isnull%281%2F0%29+%2F* a' ORDER BY 1;#
Cross Site Scripting	<script>alert("TEST");</script> <script>alert("HELLO");</script> <SCRIPT SRC=http://ha.ckers.org/xss.js></SCRIPT>

C. Reporting - Phase 3

This final stage in the model concludes the assessment from the combination of the two main phases – data gathering and attacks. Vulnerability analysis result is based on the results of two activities from these two different phases. Mapping the risk rating conducted in Phase 1 and validation of penetration testing in Phase 2, are the major sources of vulnerability analysis. Once completed, the report will provide the identification of all security vulnerabilities found. Each finding will be assigned a risk rating based on certain criteria, together with remediation recommendations to resolve the vulnerability. This phase analyses all the vulnerabilities based on http response collected after the injection of attacks to the target application. The results are then categorised into three classes as shown in Table IV.



```
<url>http://WackoPicko/users/login.php
</url>
    <method>post</method>
    <parameters-group>
            <parameters>
            <param>username</param>
    <file> input1.txt</file>
            </parameters>
            <parameters>
            <param>password</param>
            </parameters>
    </parameters-group>
```

Figure 3. Sample of input.xml

TABLE IV. CLASSES OF RESULT

Example of attack string	Example of HTTP Response	Classes of result
‘	You have an error in your SQL syntax	SQL Error
“	The username/password combination you have entered is invalid	No error
' OR '1'='1	ID: ' OR '1'='1 First name: admin Surname: admin ID: ' OR '1'='1 First name: Gordon Surname: Brown	Bypass Application

IV. RESULTS AND DISCUSSION

This Section presents the results of tests carried out to verify the model of study. Three vulnerable websites were chosen for this experiment; WackoPicko is an online photo sharing website that allows users to upload, comment and purchase pictures, while Peruggia is a website which is similar to WackoPicko. The third website is Damn Vulnerable Web Application (DVWA), normally used as an aid for security professionals to test their skills and tools in a legal environment, and help web developers understand better the processes of securing web applications. All these websites are designed with a number of vulnerabilities, such as cross-site scripting and SQL injection. Usually, the vulnerable websites were selected by researchers to test, investigate and verify their methods or approaches [27][28]. This experiment focuses only on SQL injection vulnerability. Our model was deployed by setting up the Eclipse development environment with Java Program. The Apache HTTP Client API library was installed in the machine to extract HTTP header from response pages. In the test case generation phase, 1600 attack test injections were generated for SQL injection attack. We ran the model with attack test cases, and the results are summarised in Table V and Table VI. The response results will indicate the vulnerability if error messages and bypass authentication results appeared in the HTML document header. Based on results of the test, it could be concluded that all input forms are vulnerable to the website. Due to some constraints, Acunetix was the only tool available to us at the time of writing this paper. The results of running the scanner against vulnerable web applications in the scanning phase are shown in Table V. SQL vulnerabilities were discovered in WakcoPicko and DVWA websites, but not in the Peruggia website. The function of the scanning tool is to find weaknesses in the application. The examples in Table V, when the tool inserted attack string 1 ' ", and the database error generated with SQL error in the message status, the tool will indicate that there are vulnerabilities. The tool provides only an overview of SQL error without explicitly detailing the weaknesses in the system. If seen randomly, SQL error does not give any meaning to a new tester (not an expert security tester). Thus, our model can solve problems found in Phase 2. Usually in penetration testing, the security tester will verify manually whether an attack can be executed or otherwise. Usually the attack used to verify whether the attack is successful or not (for the login form) is by using the attack string ' OR 1 = 1--.

TABLE V. DETECTION RESULT AT SCANNING PHASE

Application	**Target parameter**	**Attack String**	**Result Analysis**
Wacko Picko	username/ password	1'"	SQL Error
Peruggia	username/ password	None	None
DVWA	userid	1'"	SQL Error

TABLE VI. DETECTION RESULT AT ATTACK PHASE

Application	**Target parameter**	**Attack String**	**Result Analysis**
WackoPicko	username/ password	‘	SQL error
		1'"	SQL error
		' OR '1'='1	Bypass application
		' order by 1 #	Bypass application
		1 ORDER BY 1	No error
		"a' OR database() LIKE '%A%';#	Bypass application
Peruggia	username/ password	‘	No error
		1'"	No error
		' OR '1'='1	No error
		' order by 1 #	No error
		1 ORDER BY 1	No error
		"a' OR database() LIKE '%A%';#	Bypass application
DWVA	userid	‘	SQL error
		1'"	SQL error
		' OR '1'='1	Bypass application
		' order by 1 #	No error
		1 ORDER BY 1	Bypass application
		"a' OR database() LIKE '%A%';#	Bypass application

Table VI shows a list of attack strings which successfully bypass the application and entered the application. As seen in Table VI, WackoPicko and DVWA are the easiest to bypass the application. By simply using the simple attack string ' OR 1 = 1, one can easily enter into the application. On the other hand, Peruggia requires advance test cases to enter the application. Acunetic scanner tool could not detect any vulnerability found in the Peruggia website. The result of this study proves that the study model successfully detects vulnerability even though it cannot be detected during the scanning phase.

V. CONCLUSION

This paper aims to provide a web security assessment model for in-house self-assessment exercise which will help to identify the weaknesses and potential vulnerabilities of web applications. OWASP Top Ten vulnerabilities classification is used as the main reference or guidelines to seek security holes in the web applications and simulate hackers' actions via specific test cases to validate the real existence of vulnerabilities. The overall methodology is relatively straightforward, but the existing method was extended with newly generated test cases and analysed http response with three different classifications; SQL error, no error and bypass application. After conducting the security assessment in selected web applications, the result shows that the model has successfully detected vulnerability in the web applications even though it cannot be detected during the scanning phase. The result is then categorised based on three classifications and it was found that the class with bypass applications is with critical vulnerabilities and requires immediate action to mitigate risks. There is intention of implementing other attack types such as cross site scripting and parameter manipulation attack to replace the SQL injection in future.

ACKNOWLEDGMENT

This work was supported by the Advanced Informatics School (AIS), University Technology of Malaysia and National Defence University of Malaysia.

REFERENCES

[1] G. B. shelly and M. E. Vermaat, "Discovering Computers 2009: Living in Digital World, Complete," Cengage Learning Course Technology, 2009.

[2] A. Al-dahoud and C. Universitaria, "E-Government : Benefits , Risks and a Proposal To Assessment Including Cloud Computing and Critical Infrastructure," 2013.

[3] P. Katsumata, J. Hemenway and W. Gavins, "Cybersecurity risk management," Military Communications Conference, 2010 - MILCOM 2010 , vol. Oct. 31 2010-Nov. 3 2010, no., pp.890-895.

[4] K. Francis, B. Andoh and K. O. Bryson, "Exploring the characteristics of Internet security breaches that impact the market value of breached firms," Expert Systems with Applications, Volume 32, Issue 3, April 2007, pp. 703-725, ISSN 0957-4174.

[5] G. Jeremiah, "The State of Website Security," Security & Privacy, IEEE , vol.10 no.4, 2012, pp.91-93.

[6] J. D. Meier, A. Mackman, M. Dunner, S. Vasireddy, R. Escamilla and A. Murukan, "Improving Web Application Security: Threats and Countermeasures," Microsoft Corporation, http://msdn.microsoft.com/en-us/library/aa302420.aspx, 2003 [retrieved: July, 2015].

[7] A. Ahmad, S. R. Ahmad, N. F. Awang and M.Z. Ali, “Web Vulnerability Assessment: Outsource dilemmas,” Electrical Engineering and Informatics (ICEEI), 2011 International Conference , vol., no., 2011, pp.1-6.

[8] P. Xiong and L. Peyton, “A model-driven penetration test framework for Web applications,” Privacy Security and Trust (PST), 2010 Eighth Annual International Conference on , vol., no., 2010, pp.173-180.

[9] Y.-W. Huang, S.-K. Huang, T.-P. Lin, and C.-H. Tsai, “Web application security assessment by fault injection and behavior monitoring,” Proc. twelfth Int. Conf. World Wide Web - WWW ’03, p. 148, 2003.

[10] C. Colwill, and A. Gray, “Creating an effective security risk model for outsourcing decisions,” BT Technology Journal, Vol. 25 No. 1, 2007, pp. 79-87.

[11] G. Nassimbeni, M. Sartor and D. Daiana, “Security risks in service offshoring/outsourcing: an assessment model based on the Failure Mode and Effect Analysis,” POMS 21st Annual Conference, Vancouver, Canada, 2010.

[12] J. G. Kim, “Injection Attack Detection Using the Removal of SQL Query Attribute Values,” Information Science and Applications (ICISA), 2011 International Conference on , vol., no., April 2011, pp.1-7, doi: 10.1109/ICISA.2011.5772411

[13] Z. Su and G. Wassermann, “The Essence of Command Injection Attacks in Web Applications,” In Conference Record of the 33rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, 2006, pp. 372-382.

[14] T. Scholte, D. Balzarotti, and E. Kirda, “Have things changed now? An empirical study on input validation vulnerabilities in web applications,” Comput. Secur., vol. 31, no. 3, pp. 344–356, 2012.

[15] Trustewave, The Trustwave 2012 Global Security 2012, https://www.trustwave.com/spiderlabs, 2012. [retrieved: July, 2015].

[16] S. Zanero, L. Carettoni and M. Zanchetta, “Automatic Detection of Web Application Security Flaws,” Black Hat Forum, 2005.

[17] N. F. Awang, A. A. Manaf and W. S. Zainudin, “A Survey on Conducting Vulnerability Assessment in Web-Based Application,” 2014, pp. 459–471.

[18] The Open Web Application Security Project: The Ten Most Critical Web Application Security Vulnerabilities. https://www.owasp.org/index.php/Main_Page:OWASP_Top_Ten_Project, [retrieved: July, 2015].

[19] N. Jovanovic, C. Kruegel and E. Kirda, “Static analysis for detecting taint-style vulnerabilities in web applications,” Journal of Computer Security, 2010, pp. 861-907.

[20] Y. Xie and A. Aiken, A, “Static detection of vulnerabilities in scripting languages,” Proc. 15th USENIX Security Symposium, 2006, pp. 179-192.

[21] N. Antunes and M. Vieira, “Comparing the effectiveness of penetration testing and static code analysis on the detection of SQL injection vulnerabilities in web services,” 2009 15th IEEE Pacific Rim Int. Symp. Dependable Comput. PRDC 2009, 2009, pp. 301–306.

[22] N. Ayewah, D. Hovemeyer, J. D. Morgenthaler, J. Penix and W. Pugh, “Using Static Analysis to Find Bugs,” IEEE Software, 2008, pp 22-29.

[23] M. Vieira, N. Antunes and H. Madeira, “Using Web Security Scanners to Detect Vulnerabilities in Web Services,” IEEE/IFIP Intl Conf. on Dependable Systems and Networks, DSN 2009.

[24] R. S. Basaval, “Web application vulnerability detection using dynamic analysis with penetration testing,” International Journal of Enterprise Computing and BusinessSystems, Vol 2, 2012.

[25] M. Curphey and R. Araujo, “Web Application Security Assessment Tools,” IEEE Security & Privacy, Published By The IEEE Computer Society, 2006.

[26] OWASP Testing Guideline, https://www.owasp.org/index.php/OWASP_Testing_Guide_v4_Table_of_Contents. [retrieved: July, 2015].

[27] R. Akrout, E. Alata, M. Kaaniche, and V. Nicomette, “An automated black box approach for web vulnerability identification and attack scenario generation,” J. Brazilian Comput. Soc., vol. 20, 2014, p. 4.

[28] Z. Djuric, “A black-box testing tool for detecting SQL injection vulnerabilities,” 2013 2nd Int. Conf. Informatics Appl. ICIA 2013, 2013, pp. 216–221.