Security in WEB Applications, Definitions, Risks and Tools

Alejandra Santoyo-Sanchez[1], RediCórdova-Arbieto[2], and Carlos De Jesús-Velásquez[3]

[1]Department of Computing Science, Universidad de Guadalajara–CUCEI, Guadalajara, Jalisco, México

[2] Facultad de Ingeniería de Sistemas e Informática, Universidad Nacional Mayor de San Marcos–UNMSM, Lima, Perú

[3]Compatibility Validation, Intel Tecnología de México S.A., Tlaquepaque, Jalisco, México

Phone (33) 1378 5900   E-mail: alejandra.santoyo@cucei.udg.mx

Abstract. Security in WEB applications has become a major concern for the scientific and business communities today. An increasing amount of money is being spent for handling information security. . Therefore, giving the proper importance of handling information security, the paper focuses on: definitions of software security, vulnerabilities and risks, dealing with various threats and vulnerabilities, the risk ranking created by OWASP (Open Web Application Security Project) and describes different tools that can be used for security within a Web application using a test, some of them are Zed Attack Proxy (ZAP), BeEF (The Browser Exploitation Framework), Burp Suite, PeStudio, Xenotix XSS Exploit Framework, Lynis, Recon-ng, Suricata, WPScan, and O-Saft (OWASP SSL Advanced Forensic Tool).

Keywords: WEB Application, Reliability, Risk, Testing, Security Standard.

1 Introduction

Meanwhile the world is more connected, the need for security in the procedures used to share information become more important. Since it has generated an evolution of information systems using the Internet. According to [1-4] information systems are a mechanism that helps to collect, store, organize and disseminate the data contained therein; its purpose is to help users to get some kind of value of information that is in the system, regardless of the type of information that is stored and the type of desired value. The influence of the Internet in all processes and communication stages of current media covers the registration, handling, storage and distribution of information, whether the form of text, still and moving images, sound, etc.

From the perspective of software developers, the need for users to access information from anywhere on any device, they are frequently asked to fill registration forms, by entering the information required to specific tasks to operate. However, registration can be done by non-automated software and user data can be compromised or corrupted.

It is crucial for WEB applications to consider a defense system against various types of threats and attacks. Among the first lines of defense are the CAPTCHA images [5]. Its function is to identify if the user of the WEB application is a human user/or automated software designed to gain access to the WEB application. This software is named as Optical Character Recognition (OCR).

In order to guarantee safety aspects in WEB applications involve performing a variety of tests in a complex set of possible scenarios. Therefore, it is difficult to remove all security flaws and performance during the development of WEB applications.

According to [6], software vulnerabilities can be grouped in three categories: the design, development and implementation, and operation. Unfortunately, in the practice, analysts and designers of WEB applications do not specify security requirements, and often do not provide assessment for vulnerabilities in their design; where most security risks occur during the implementation stage [7]. So, this paper focuses on analyzing the characteristics of the most common risks defined by OWASP during the implementation phase, and the use of security tools to help in this complex task.

The paper is organized as follows; the second section contains the basics definitions about risks on WEB applications, security, vulnerability, and threats. It also shows the list of the 10 most common vulnerabilities according to OWASP taxonomy [7], and the guide vulnerabilities proposed in [8]. The third section presents the analysis of existing software tools under GNU-GPL license for the risks described in section two, which are based on questionnaires given in the guidance document in [8]; through descriptive pictures. The fourth section illustrates the use of the tools discussed in the implementation of a case study. Finally, conclusions and future work are presented in the fifth section.

2 Background

Definition 2.1. *A WEB application* consists of a set of WEB pages and components that interact to form a system running using WEB server (s), network (s), Hyper Text Transport Protocol (HTTP), and a browser, where its state changes according to the information users provide or request [8-9].

The issue of proper handling information security has been raised in various working groups around the world, such as IEEE, ISO and NIST. As result from those working groups, here some definitions based on the ISO/IEC17799 standard are presented[9] to provide adequate support for reading topics.

Definition 2.2. *Information security* is the preservation of confidentiality, integrity and availability of information, where: *confidentiality* is defined as ensuring that information is accessible only to those who are authorized to access. *Integrity* is defined as safeguarding the accuracy and completeness of information and processing methods. And *availability* is defined as ensuring that authorized users have access to information and associated assets when required.

Moreover, from the viewpoint of software developers, the information security involves a set of methodologies, practices and procedures that seek to protect information as a valuable asset and thereby reduce threats and risks to ensure that resources system to be used the way it was decided (namely *security*). Security cannot be a product, it is a process, and it is an activity that attempts to reduce risks. In order to clarify the terms used herein, the definitions "threat", "vulnerability" and "risk" adapted from [9-12] are listed next.

Definition 2.3. *Threat* is any circumstance or event with the potential to adversely impact a system through unauthorized access, destruction, disclosure and/or data modification or denial of service.

Definition 2.4. *Vulnerability* is a flaw or weakness in system's design, implementation, or operation and management that could be exploited to violate the system's security policy.

Definition 2.5. *Risk* is a function between the possibility of the occurrence of a potentially disastrous event (namely *threat*) and the inability to smoothly absorb the effects of the event (namely *vulnerability*).

In [8] a set of guidelines based on standards: ISO 9126:1991, IEEE 1233, IEEE 6190, IEEE 830, ISO / IEC 9646-1 is proposed (see references [10-14]). The guide is based on questionnaires, which indicate at each stage of development of a WEB application, what security issues are found in the application during each stage. The guide indicates which subjects have been validated. Some questions contained in the guide are: Is there a security protocol to avoid outside attacks and intrusions from hackers? Do you avoid program names and directories being seen,? Does it protect the integrity of their programs and data? Are the services offered are made via secure transaction channels?

Just to mention some questions you can give a general idea of the aspects to be taken into account for attaining security within a WEB application. During the *test* phase, you should check the requirements, analyze the design and perform code review. In case of errors or new requirements in WEB development, requires to determine the cost, time and impact of a change in the existing product. Also, to document the changes and verify the consistency of the changes. Some questions in this phase are: Which threats exist? What problems are caused? Where are they? Does the occurrence of a threat can trigger other? How often were presented threats? What was the damage? What elements of the application are most exposed? What tools are available to us to measure vulnerabilities? What capacity has to continuously assess threats?

These questions reflect, and it becomes more clear, that efforts for eradicate or treatment of threats don't need to be limited, but also is required a safety plan has to be started during all the development cycle of an application from initial design to maintenance phase. The guide give a good basis to implement security in WEB applications, in particular, this work focuses on the available tools to complement the security process.

The OWASP in [15] has released an updated ranking for the most critical risks affecting the WEB applications. This ranking, comes from the consensus and debate among experts [7], which is described in Tables 1 and 2. Moreover, in order to assess risks, three steps are defined: (1) Assessment of the threat, (2) Analysis of vulnerability, and (3) The risk estimate as a result of linking the two parameters above. A template that can help you document the assessment is presented in Table 3.

The intrusion is the most difficult kind of threats, and there are more issues to consider and deal with. Attackers have a range of capabilities and motivations, where threats agents can arise from many groups of people. These potential attackers will also have a wide range of capabilities, resources, organizational support, and motivations. The following is a brief list of potential attackers/threats agents.

- **Hackers**, who have computer resources, knowledge, dedication and find it funny and challenging.
- **Employees**, who have inside knowledge, easy access, and do it on/without purpose.
- **Insiders, contractors, and competitors**, who have access to confidential information, and possess knowledge of operations and default passwords.
- **Traders**, who have computer skills and could financially gain.
- **Foreign governments**, which have resources (system expertise, computers, cryptographers, money), intelligence agencies, and an interest in military or cause economic damage.
- **Organized crime, extremist groups, and terrorist**, who have dedication, computer skills, to financially gain and harm groups they oppose.
- **Alliances of the above**, with a combination of reasons.

Based on the above' s descriptions (today's and tomorrow's situations), the following threats may be evident for WEB applications:

- **External intrusion** by unauthorized access or by customers who do more that they are allowed to do. Also, an intruder may interfere with the system users, such that the user cannot access to the system and use the service as they expected (denial of service).
- **Internal intrusion** by employees, it could be with or without the intention to do harm.

Furthermore, "Intel's McAfee WEB Protection" [16] considers the following threats to WEB applications.

- **Balancing risk and usability**. Usability and security in WEB applications cannot be necessarily mutually exclusive. Some measures taken to increase security often affect usability. It is necessary to consider intruders as "users", and to use security measures to identify if the user who interacts with the application if it is or not and intruder. For instance, the case of a user name and password used for registration that are expected by the procedures.
- **Data tracking**. The scope is to establish the "origin point" of the information (or data). The idea is based on a point-to-point view of data transfer and communication security; in the WEB application context, a security scheme needs to provide the capability and means to control access to the data, and to control its use. Thus, it is possible determinate if the request of a user are or are not validate.

- **Filter Check**. Through this process, data is validated. If to ensure that data are properly filtered to inside, it is possible “to eliminate” or “minimize” the risk of contamination or malicious data, which are used to cause undesired operation in the application (for example App Store prevent the error).

Table 1. OWASP 2013 Top 10 of Security WEB applications risk, first part.

RISK	DESCRIPTION
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, 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.

Table 2. OWASP 2013 Top 10 of Security WEB application risk, second part.

RISK	DESCRIPTION
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.

Table 3. Risk Evaluation Template

Threat Agent	Attack Vectors	Prevalence of Weaknesses	Defectivity weaknesses	Technical impact	Business Impact
Application-specific	Easy	Diffused	Easy	Severe	Specific by Application / Business
	Average	Common	Average	Moderate	
	Hard	Uncommon	Hard	Less	

- **Escape exits**. It is the process of verifying if escape strings were properly implemented and its counterpart to encode/decode special characters. So, its original meaning is preserved.

3 Tools for Security Testing

The following list of tools for security testing, which has analyzed in this work (see references [17-26]).

- **OWASP Zed Attack Proxy (ZAP)** is an easy to use integrated penetration testing tool for finding vulnerabilities in WEB applications. It is designed to be used by people with a wide range of security experience and is highly recommended for developers and functional testers who are new to penetration testing. ZAP provides automated scanners as well as a set of tools that allow you to find security vulnerabilities manually.

- **BeEF (The Browser Exploitation Framework)** is a penetration testing tool that focuses on the WEB browser. It allows the professional penetration tester to assess the actual security posture of a target environment by using client-side attack vectors.
- **Burp Suite** is an integrated platform that combine advanced manual techniques with automated tools, some of its components are: an intercepting Proxy, an application-aware Spider, an advanced WEB application Scanner, an Intruder tool, and a Sequencer tool.
- **PEStudio** is a tool for analyzing unknown and suspicious executable files. The goal of PEStudio is to detect malicious behavior, provide indicators and score the trust for the executable being analyzed. Since the executable file being analyzed is never started, it is possible inspect any unknown or malicious executable with no risk. \
- **OWASP Xenotix XSS Exploit Framework** is an advanced XSS vulnerability detection and exploitation framework. It provides Zero False Positive scan results with its unique Triple Browser Engine (Trident, WebKit, and Gecko) embedded scanner. It has more of 1500 different Payloads for effective XSS vulnerability detection and WAF Bypass.
- **Lynis** is an open source security auditing tool. Primary goal is to help users with auditing and hardening of UNIX and Linux based systems. The software is very flexible and runs on almost every UNIX based system (including Mac). It includes searching for installed software and determines possible configuration flaws.
- **Recon-ng** is a full-featured WEB Reconnaissance framework written in Python. It is designed exclusively for web-based open source reconnaissance.
- **Suricata** is a high performance Network IDS, IPS and Network Security Monitoring engine. Open Source and owned by a community run non-profit foundation, the Open Information Security Foundation (OISF). This is highly scalable, i.e. it can run one instance and it will balance the load of processing across every processor on a sensor Suricata is configured to use.
- **WPScan** is a vulnerability scanner which checks the security of WordPress installations using a black box approach, which is made in Ruby.
- **O-Saft (OWASP SSL Advanced Forensic Tool)** is an easy to use tool to show information about SSL certificates and tests the SSL connection according to given list of ciphers and various SSL configurations. It's designed to be used by penetration testers, security auditors or server administrators. The idea is to show the important information or the special checks with a simple call of the tool. However, it provides a wide range of options so that it can be used for comprehensive and special checks by experienced people.

From the description of tools for security testing above, table 4 achieve an overview from the tools and their relations with top security risk defined by OWASP, where “X” inside one box of the matrix indicates if the tool include the risk in its security test. However, determining the necessary actions to eliminate hazards found, it is responsibility of test developer, since the tools only indicate the vulnerability.

4 Results

By using the guidelines proposed in [8] and the tools mentioned in the third section, the results illustrated in Figure 1 were obtained. Those were performed in 6 WEB massive use applications available today.

During the analysis of each tool, an important factor in determining the number of risks in the application is the number of breakpoints present. This refers to the ability of the WEB application to perform an action if detects a threated from the attacker (attack test in our analysis), forcing to take another route or scope (more tedious) to see if it can undermine the application, the more breakpoints found lower risks.

TABLE 4: ANALYSIS OF TOOLS VS TOPS SECURITY WEB RISK

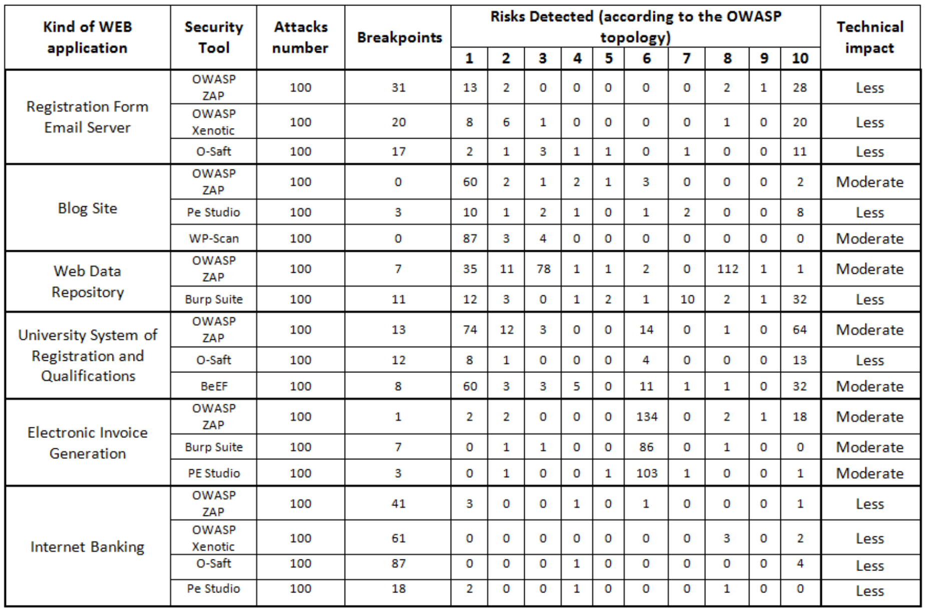
TOOLS	**RISK RANKING CREATED BY OWASP**										**PLATFORMS**	**LICENCE**
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**		
OWASP Zed Attack Proxy (ZAP)	X	X	X	X	X	X	X	X	X	X	Windows/Unix	GNU/UNIX
BeEF (The Browser Exploitation Framework)	X	X		X	X	X	X		X	X	IExplorer Google Chrome Mozilla Firefox	Free Project
Burp Suite	X	X					X		X	X	Windows/Unix/ Web Browser	Free Software
PeStudio	X	X	X	X		X	X		X	X	Windows	Free Software
OWASP Xenotix XSS Exploit Framework			X					X			Windows / Unix	GNU/UNIX
Lynis	X	X	X	X	X	X	X	X	X	X	Unix/Linux	GNU/UNIX
Recon-ng	X	X		X		X			X		Windows	GNU/UNIX
Suricata	X	X	X	X	X	X	X	X	X	X	Windows / Web Browser	Open Source GNU/UNIX
WPScan	X	X	X								IExplorer Google Chrome Mozilla Firefox	Free Software
O-Saft (OWASP SSL Advanced Forensic Tool)	X	X	X							X	Windows/Unix	GNU/UNIX

Note also that each tool uses different methods to detect risks, so some despite having few breakpoints have fewer vulnerabilities detected , since the application is not considered important to protect these techniques used by software , often because they know they have nothing to lose in that way. It can be inferred that with reference to ZAP OWAS tool is able to identify and mitigate as many risks.

5 Conclusions and Future Work

This paper contributes with an analysis of available tools that can be used for the identification of security risks in WEB applications, based on the basic concepts of risks WEB applications, security, vulnerabilities, and threats. We focused the analysis on 10 of the most critical vulnerabilities according to the OWASP taxonomy and guidance given in [8]. According to the results, OWAS ZAP was the security tool with better score reported. Those results can be used to determine what the weaknesses points of the application are. As a future work, we will propose a list of best practices during the development phase that might help to reduce the number of risks found using the automated security set of tools.

FIGURE 1: VULNERABILITY FOUND IN WEB APPLICATIONS.



Kind of WEB application	Security Tool	Attacks number	Breakpoints	Risks Detected (according to the OWASP topology)										Technical impact
				1	2	3	4	5	6	7	8	9	10	
Registration Form Email Server	OWASP ZAP	100	31	13	2	0	0	0	0	0	2	1	28	Less
	OWASP Xenotic	100	20	8	6	1	0	0	0	0	1	0	20	Less
	O-Saft	100	17	2	1	3	1	1	0	1	0	0	11	Less
Blog Site	OWASP ZAP	100	0	60	2	1	2	1	3	0	0	0	2	Moderate
	Pe Studio	100	3	10	1	2	1	0	1	2	0	0	8	Less
	WP-Scan	100	0	87	3	4	0	0	0	0	0	0	0	Moderate
Web Data Repository	OWASP ZAP	100	7	35	11	78	1	1	2	0	112	1	1	Moderate
	Burp Suite	100	11	12	3	0	1	2	1	10	2	1	32	Less
University System of Registration and Qualifications	OWASP ZAP	100	13	74	12	3	0	0	14	0	1	0	64	Moderate
	O-Saft	100	12	8	1	0	0	0	4	0	0	0	13	Less
	BeEF	100	8	60	3	3	5	0	11	1	1	0	32	Moderate
Electronic Invoice Generation	OWASP ZAP	100	1	2	2	0	0	0	134	0	2	1	18	Moderate
	Burp Suite	100	7	0	1	1	0	0	86	0	1	0	0	Moderate
	PE Studio	100	3	0	1	0	0	1	103	1	0	0	1	Moderate
Internet Banking	OWASP ZAP	100	41	3	0	0	1	0	1	0	0	0	1	Less
	OWASP Xenotic	100	61	0	0	0	0	0	0	0	3	0	2	Less
	O-Saft	100	87	0	0	0	1	0	0	0	0	0	4	Less
	Pe Studio	100	18	2	0	0	1	0	0	0	1	0	0	Less

6 Acknowledgements

This work was developed in the context of project and PAME-DUAL CO STUDENT MOBILITY period January to July 2014, between the University of Guadalajara, Jalisco Mexico and the National University of San Marcos, Lima, Peru.

7 References

1. Y. Deshpande, S. Hansen, "*Web Engineering: Creating a Discipline among Disciplines*", Vol.8, No. 2, pp.82 – 87, Ed. IEEE Multimedia, 2001.
2. D. Andrés Silva, B. Mercerat "Construyendo aplicaciones web con una metodología de diseño orientada a objetos", *Magazine: Colombian Journal of Computation*, Vol. 2, pp. 79 – 95, 2001.
3. Peter Norton, "Introducción a la computación", sexta edición, Editorial Mc Graw Hill, ISBN 970-10-5108-4, 2006.
4. Piattini Velthuis, Mario Gerardo, "Tecnología y diseño de Bases de Datos", 1ª edición, Editorial AlfaOmega, ISBN: 9701512685, 2007.
5. M. Dailey, C. Namprempre, "A text graphics character CAPTCHA for password authentication", Proc. on IEEE Region 10 Conference TENCON, 21-24, 2004, Print ISBN 0-7803-8560-8, Vol. 2. pp 45 – 48.
6. "Desarrollo Seguro de Aplicaciones", en http://es.scribd.com/doc/54453491/6/Aplicaciones-inseguras, May 2011.
7. OWASP Breakers community, "Web Application Penetration Testing", OWASP Testing Guide Table of Contents, Ver. 4, Section 4-5, 2014.
8. A. Santoyo-Sanchez, C. De Jesús-Velásquez, L. I. Aguirre-Salas, "Security in Web applications", Research in Computing Science: Tendencias Tecnológicas en Computación, Vol. 66, 2013, pp. 161 – 174, ISSN 1870-4069, Ed. Instituto Politécnico Nacional, México D.F, México.
9. ISO/IEC 17799 (International Standard ISO/IEC 17799 Second edition 2005-06-15) http:/www.iso17799software.com/
10. ISO 9126: 1991 (Software Engineering Product Quality part 1, 2, 3), http://www.iso.org
11. IEEE Std 1233-1998, IEEE Guide for Developing System Requirements Specifications, IEEE STANDARDS, http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=741940
12. IEEE Std 610.12, (Standard Glossary of Software Engineering Terminology) 1990, http: www.swen.uwaterloo.ca/~bpekilis/public/softwareEnGlossary.pdf
13. IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications, IEEE, http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=720574
14. IEEE Standard 802.16 - IEEE Standard for Conformance to Part 2: Test Suite Structure and Test Purposes (TSS&TP) for 10-66 GHz WirelessMAN-SC Air Interface, http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1273457
15. The Open Web Application Security Project, "The Ten Most Critical Web Application Security Risks", The OWASP Foundation, 2013.

16. McAfee An Intel Company, "Web Security your way-SaaS. on-premises, or hybrid combination", in http://www.mcafee.com/au/resources/data-sheets/ds-web-protection.pdf , MacAfee Web Protection, Data Sheet,2012, Santa Clara, CA, USA, pp 1-2.
17. S. Sampath, V. Mihaylov, A. Souter, L. Pollock, "Composing a framework to automate testing of operational Web-based software", in Proc. 20th IEEE International Conference on Software Maintenance (ICSM), 2004, Chicago, IL, USA, pp. 104 – 113.
18. OWASP Zed Attack Proxy Project – OWASP, in https://www.owasp.org/index.php/ OWASP_Zed_Attack_Proxy_Project, consulted January/16/2014.
19. Burp Suite, in http://portswigger.net/burp/, consulted February/12/2014.
20. PESTudio, in http://www.winitor.com/, consulted march/01/2014.
21. OWASP Xenotix XSS \, https://www.owasp.org/index.php/ OWASP_Xenotix_XSS_Exploit_Framework, consulted January/16/2014
22. Lynis, http://cisofy.com/lynis/, consulted February/16/2014.
23. Recon-ng, https://bitbucket.org/LaNMaSteR53/recon-ng, consulted April/01/2014.
24. Suricata, http://suricata-ids.org/, consulted may/08/2014.
25. WPScan, https://github.com/wpscanteam/wpscan, consulted may/08/2014.
26. O-Saft, https://www.owasp.org/index.php/O-Saft, consulted January/16/2014.