USING OPEN SOURCE SOFTWARE FOR WEB APPLICATION SECURITY TESTING

Ksenija Živković, Ivan Milenković, Dejan Simić

Faculty of Organizational Sciences, University of Belgrade, Belgrade, Republic of Serbia

ksenija.zivkovic@mmklab.org, ivan.milenkovic@fon.bg.ac.rs, dejan.simic@fon.bg.ac.rs

Case Study

DOI: **10.7251/JIT1602086Z** UDC: **004.738.5.056**

Abstract: Web applications are a standard part of our everyday lives. Their purpose can vary significantly, from e-banking to social networks. However, one thing is similar - users have generally high expectations from different web applications. To assure such high expectations, proper web application testing is necessary. Non-functional testing is an important part of web application testing. As technology advances and requirements become more complex, the importance of non-functional application aspects becomes even greater. It is necessary to identify non-functional requirements of web applications which are important to users, implement those requirements and test them.

Keywords: non-functional testing, web applications, testing tools.

INTRODUCTION

Software errors in most cases can cause trivial problems that have no greater impact on business and can be easily solved. However, software errors in flight control or medical equipment can not be allowed, because consequences may be disastrous. For companies, software errors could bring grave financial damage. Below are examples of problems that errors caused in the past.

In 1998 NASA launched the Mars Climate Orbiter, a robotic space probe for studying climate, atmosphere and surface changes on Mars. Instead entering the Martian atmosphere, it was destroyed because a navigation error caused it to miss its target altitude. Main cause of this problem was a bad translation of English to metric units. This way a project worth 327 million dollars was destroyed [9].

In 2003, the Amazon UK web site had to be closed because of error in pocket computer prices, which were sold for 7 instead of 192 pounds. As a result, all orders had to be cancelled [12].

In 2014, Sony suffered an enormous attack where hackers erased data from the system, stole and published movies that were not yet published, private and sensitive data [10]. Before that, in 2011, hackers have accessed data of 77 million users of PlayStation network. Company has lost millions of dollars because website was not working for over a month [8].

In February this year, Volvo had to recall 59,000 cars over software fault that could cause temporary shutdown of an engine while the car is in motion. This error was reported by drivers of new Volvos, who experienced a brief absence of steering and braking. The error could have caused traffic accidents and could be life threatening for drivers, and at the same time negatively affect reputation and financial stability of a company [12].

National Institute of Standards and Technology (NIST) concluded a research study in 2002. The study concluded that software errors cost American economy 59 billion dollars per year, and that with

better testing there could be saved 22.2 billion dollars [9].

In order to avoid problems caused by software omission, errors should be identified and amended on time, before they even appear. Process of identifying errors is called software testing. Testing is an important activity in software development and it will be described in detail in the next chapters.

Section 2 of this paper describes the process of application testing. In section 3, two tools for non-functional application testing are described. Analysis of a case study where described tools are applied is given in section 4. Section 5 contains conclusions and suggestions for future improvements.

APPLICATION TESTING

Testing is a method for software quality control and an important activity in software development, and its purpose is identifying errors. It represents check if software is implemented according to user requirements. In a broader sense, testing is a process of quality control, during which, besides checking software, contains checking of its components and characteristics.

There are two basic types of testing: [11]:
- Functional testing and
- Non-functional testing.

Functional testing checks if application meets all necessary functional requirements. It checks if application does what is made for. This type of testing will not be further discussed in this paper.

Non-functional testing checks behavior and readiness of an application. Focus is on the aspects of software that refer to software quality, not functionality.

Quality aspects of software that are determined by non-functional testing are:
- Performance,
- System behavior under heavy load,
- Reliability,
- Security.

Despite big differences between functional and non-functional testing, there are common basic concepts, team roles and activities in testing teams.

Non-functional application testing

Types of non-functional testing are [1]:
- Load testing,
- Security testing,
- Configuration testing and
- User interface testing.

Load testing is performed in order to determine system's behavior under extreme conditions and discover its endurance limits [3]. Result of testing can be time, amount of used memory, etc. Types of load testing are:
- Performance testing and
- Stress testing.

Performance testing checks how the system functions under normal load [3]. Time for performing an action is analyzed. Factors that affect application performance are: platform, infrastructure, number of users, etc. Each of the above factors needs to be considered during testing and acceptable results need to be defined. Some of the tools for performance testing are: FireBug, JMeter, Grinder, etc.

Stress testing checks how system functions under extreme conditions [3]. Goal of stress testing is finding limits of application's endurance. It can be tested how the system behaves with greater amount of users or a large database.

Security testing is a type of software testing which purpose is detecting system's vulnerabilities. It checks whether the system data is protected from unauthorized access, during which data could be altered or deleted.

Areas to be considered during security testing are:
- Network security,
- Software security,
- Security on client's side,
- Security on server's side.

Main goal of security testing is assuring that application is safe and there are no exploitable weak-

nesses. It is performed in order to protect data and application functionality when system is attacked. Security testing includes confidentiality testing, integrity testing and authentication and authorization testing [2].

Success of application depends also on interaction between user and graphic interface. In the early stages of application development, these tests are used for accelerating its development and enhancing its quality. Testing can be performed manually or automated, by using tools for user interface testing. UI testing checks quality of interface and ease of use [3]. Interface quality refers to application appearance, and ease of use means that user has no difficulties while using the application and that there are no complicated actions. It is necessary to check whether application looks the same in the environment in which it will be used as in development environment. Application appearance needs to be concise regardless of device and web browser.

Tools for non-functional application testing
OWASP Zed Attack Proxy

OWASP Zed Attack Proxy (ZAP) is one of the most popular free security testing tools and one of the most active OWASP projects, maintained by hundreds of international volunteers. Main goal of this project is ease of use, so everyone could benefit from it. It can help with automated finding security vulnerabilities in web applications during development and testing phase.

Main characteristics:
- Free, open source tool,
- Can be used on Linux, Windows and Mac operating systems,
- Easy to use,
- Completely documented,

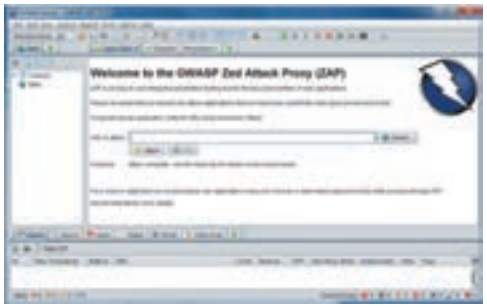

Figure 1. ZAP's interface

- Works well in combination with other software testing tools.

Some of the security problems are impossible to find with automated testing, but ZAP also has manual testing functionality.

Interface of ZAP contains:
- Menu,
- Toolbar which includes buttons for commonly used features**,**
- Window which displays the sites tree and the scripts tree,
- Workspace window,
- Information window,
- Footer, which displays a summary of the found alerts and their status.

Quick Start enables easy web application testing, which allows entering an URL that ZAP will attack. ZAP uses spider which crawls the application and passively scans all discovered pages. In the meantime, active scanner is used for attacking all found pages.

Spider is a tool which automatically finds links by examining the HTML in application responses. Using this tool it is possible to find hidden web application pages. List of found URLs depends on where spider starts with crawling.

The spider is very fast but not always effective when exploring AJAX applications. AJAX spider is more effective in this case, despite being slower, because it explores the application by invoking browsers which follow the links that have been generated.

Types of scanning ZAP does are:
- Passive and
- Active scanning.

Passive scanner identifies problems by analyzing requests and responses discovered via the spiders and it is safe to use on any web site or application, because it does not use attacks or changes the responses in any way.

Active scanning attempts to find vulnerabilities by using known attacks and it should be used only

on application where testers are allowed to perform such testing. With this kind of scanning it is not possible to identify logical errors, which means automated testing is not enough. Some of the rules of active scanning in ZAP are: Buffer Overflow, SQL Injection and Cross Site Scripting.

In order for active scanning to begin, URL should be listed in Starting point field. In Input Vectors tab target elements could be chosen, in Technology tab used technology should be listed and in Policy tab rules of active scanning could be modified.

Vulnerabilities that could not be detected by active scanners can be identified by using Fuzzer. Fuzzing is a black box software testing technique, which means that only inputs and outputs are tested. It involves providing invalid, unexpected or random data to the inputs of a web application, in order to identify implementation errors [9].

Forced browsing is an attack where the aim is to access resources that are not referenced in application. Attacker can use technique known as Brute Force in order to find sensitive data while searching through directory content.

ZAP generates reports including all found vulnerabilities of an application. Reports contain advices for avoiding these kinds of vulnerabilities and links to more information about security threats and methods to prevent them.

All of the potential vulnerabilities are shown in the Alerts tab. Alerts are connected with application requests and contain information about risks and solutions regarding those vulnerabilities. One request could have more than one alert. Types of alerts are shown on the next picture.

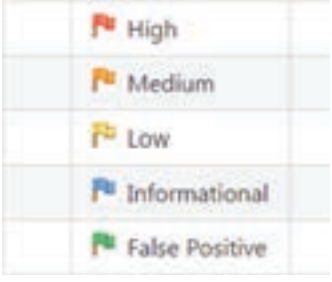

Figure 2. Types of alerts in ZAP

VEGA

Vega is a free open source scanner and security testing platform for web applications [4]. It is written in Java and runs on Linux, Windows and OS X operating systems. It is developed by Subgraph, company founded in Canada.

Figure 3. Vega's interface

Vega can be used as:

- Automated scanner or
- Intercepting proxy.

Automated scanner crawls the web application and analyzes pages content. It is capable of identifying system vulnerabilities such as XSS and SQL injection, unintended publishing of sensitive information and other vulnerabilities.

Vega as an intercepting proxy is used for tactical inspection. It is situated between the browser and the web server hosting the application which is tested. The proxy can read all requests that come from the browser and all responses that are returned from the server.

There is also an option to alter requests and responses before being passed on. In order to use this functionality, it is necessary to configure web browser.

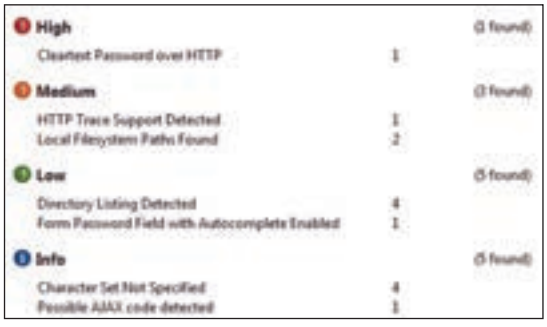

Figure 4.Types of alerts in Vega

During scanning, Vega identifies vulnerabilities and marks them with alerts. Types of alerts are shown in the picture below.

Scan Info contains data about URL where problem could occur and means of solving it.

Figure 5. Scan Info in Vega

CASE STUDY

In order to demonstrate how these tools work, security of a web application will be tested. Application is written by using next technologies:

- HTML,
- CSS,
- PHP,
- Flight framework and
- MySQL database.

For initializing, an application on local server WampServer is used. Database is also created on MySQL WampServer. Flight framework was used for creating application paths.

Figure 6. Testing results in Vega

Results of testing can be seen in the next two pictures:

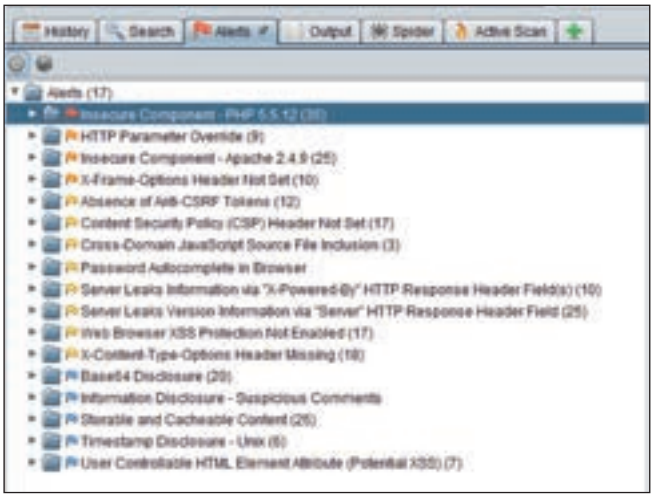

Figure 7. Testing results in ZAP

In Mozilla browser proxy was set to localhost on port 8888 which are default settings in Vega. This way proxy intercepting functionality was set so it could record all requests and responses from web application.

Alerts with highest level of risk are:

- Outdated version of PHP,
- Path traversal,
- Cleartext password over HTTP,
- SQL Injection.

During passive analysis, PHP version 5.5.12 was used, which is not supported anymore. Solution to this problem is using newer version, because web applications in older versions are susceptible to a higher number of security attacks [11].

Path traversal (dot-dot-slash, directory traversal) aims to access files and directories that are stored outside the web root folder. Attackers may manipulate variables that reference files with "dot-dot-slash" sequences and its variations to get to the files of interest [12]. One of the solutions to this problem is using white lists.

If a password is sent over HTTP, there is a danger it might be discovered by an attacker. Solution to this problem is using HTTPS instead of HTTP. That could be resolved by using SSL certificates.

SQL Injection is a technique of code injecting into web applications that are data driven. In this way attackers can get information even though they are not authorized. Web application was tested in order to test whether an attack during signing in is possible. Attack can be stopped by validating input. In case of working with database, it can be solved by preparing inputs for queries or escaping special characters.

Alerts with medium level of risk are:
- HTTP Trace Support detected,
- HTTP parameter override,
- ClickJacking attacks.

HTTP TRACE is a request method used for echoing back user inputs. If application is not secured from XSS attacks, this method can be used for getting confidential information or redirecting users to malicious sites.

On every page that contains a form possibility of overriding, HTTP parameters were detected. In order to stop this kind of attacks it is necessary to specify a name for form action, because if it does not, browser considers URL as an action name which means attackers can set values of parameters within URL, so user would not be redirected on the page it should be.

ClickJacking is an attack where attacker uses multiple transparent layers to trick the user into clicking on a page or button when they were intending to click on the other page. This way user can be redirected to a malicious page. X-Frame-Options header in HTTP responses must be set in order to prevent these attacks. This problem was reported on every page that contains a form.

Alerts with the lowest level of risk, that nevertheless should be considered are:
- Absence of Anti-CSRF token,
- Content Security policy header not set,
- Cross-Domain JavaScript source file inclusion,
- Web browser XSS protection not enabled,
- For password field with autocomplete enabled.

Other alerts are informational kind. In order to better estimate application security, manual tests should be performed. Those alerts are:
- Character set not specified,
- AJAX code detected,
- Information disclosure – suspicious comments that could help attackers.

From the above stated, it can be concluded that application is not entirely secured and that attackers can get to information from a database and files that are on the server. In order to stop attacks and save data, it is necessary to apply all techniques, i.e. if a web application is for a financial organization, loss would be enormous if attacks occur.

CONCLUSIONS

With turbulent advancement of technology and even faster development of web applications, testing becomes a real challenge. Instead of evaluating software after process of development, testing should be a part of development cycle, in order to prevent errors that could lead to problems in production. Therefore, many organizations choose to use agile software development methodologies.

Security testing is predicted to have a significant growth of importance in future. Estimated market value equals 594.7 million of dollars in this year, and it is expected to rise to 1.724 million in 2021. The rise of web application testing is correlated with the increase of hacker attacks.

In order for web application to satisfy all non-functional requirements, it is necessary to apply all kind of tests during application development. Testing should be executed by qualified and experienced testing team. Which tools, methodologies and testing strategies should be used depends on company's necessities. For example, if application contains confidential data, testing should be done with extreme care, because unidentified problems can have serious financial, legal or reputation consequences for organization.

ACKNOWLEDGMENT

This work is a part of the project Multimodal biometry in identity management, funded by the Ministry of Education and Science of Serbia, contract number TR-32013.

BIOGRAPHY

Ksenija Živković is a MSc candidate at University of Belgrade, Faculty of Organizational Sciences, Department for Information Technology. She currently works as Project Assistant at msgNetconomy, Belgrade. She published several research papers and her interests include software engineering, web security and biometrics.

Ivan Milenković is a PhD candidate at University of Belgrade, Faculty of Organizational Sciences, Department for Information Technology. He currently works as a researcher at Laboratory for multimedia communications in Belgrade, Serbia. He published several research papers and participated in several research and commercial projects in Information Technology area. His interests include biometrics, computer security and mobile technologies.

Dr Dejan Simić is a full professor at University of Belgrade, Faculty of Organizational Sciences, Department for Information Technology. He is engaged as a researcher at Laboratory for multimedia communications in Belgrade, Serbia. He published many research papers and participated in several research projects in Information Technology area. His interests include biometrics, computer security and e-commerce.

REFERENCES

[1] Chung L, and do Prado Leite J C S (2009) On non-functional requirements in software engineering. In Conceptual modeling: Foundations and applications (pp. 363-379), Springer, Berlin Heidelberg.

[2] Mervi J, Joonas M (2015) Non-functional testing: security and performance testing, Bachelor's Thesis, JAMK University of Applied Sciences, Available:

[3] http://www.theseus.fi/bitstream/handle/10024/110354/Joonas_Moilanen_Mervi_Jeskanen.pdf?sequence=1 [Accessed 5 August 2016].

[4] Myers G et all (2011) The art of software testing, John Wiley & Sons, ISBN 978-1118031964

[5] Official Vega software page, Available: https://subgraph.com/vega/ [Accessed 6 August 2016]

[6] OWASP - Definition of fuzzing, Available: https://www.owasp.org/index.php/Fuzzing, [Accessed 5 September 2016].

[7] Path traversal attack description, Available: https://www.owasp.org/index.php/Path_Traversal [Accessed August 2016]

[8] PHP supported version, Available: http://php.net/supported-versions.php, [Accessed 15 August 2016].

[9] PlayStation Network hackers access data of 77 million users, Available:

[10] https://www.theguardian.com/technology/2011/apr/26/playstation-network-hackers-data [Accessed 22 August 2016]

[11] Popović J (2014) Testiranje softvera u praksi, ISBN 978-86-7991-363-0, CET, Beograd

[12] Steinberg J (2014) Massive Security Breach At Sony, Available:

[13] http://www.forbes.com/sites/josephsteinberg/2014/12/11/massive-security-breach-at-sony-heres-what-you-need-to-know/#789e36ece9a5 [Accessed 22 August 2016]

[14] Tomic B, Vlajic S (2008) Functional Testing for Students: a Practical Approach, Inroads - ACM SIGCSE Bulletin, Vol. 40, No. 4, pp. 58-62, ISSN 0097-8418, DOI 10.1145/1473195.1473221

[15] Volvo recalls 59,000 cars over software fault, Available: http://www.bbc.com/news/world-europe-35622753 [Accessed 18 August 2016]

Submitted: November 24, 2016.
Accepted: November 30, 2016.