Department of Computer Science and Engineering

Master Degree in Computer Science

Project Report:
Befloral web application security analysis

Matteo Beltrami
Student ID: 248333

Academic year 2023/2024

Submission date: February 14, 2024
Name of filled XLS: Beltrami_Matteo_248333.xlsx

1. Introduction to the task

This report is drawn up for the project of the 2023/2024 Security Testing course offered by the University of Trento. The objective is to analyze and test the aspects related to IT security of a real web application: the idea of the project is in fact to apply the attack and risk analysis methodologies of the IT world seen in lessons and to apply further tools in a realistic situation.
As seen extensively in the course lessons, analyzing the IT security of an application is absolutely not a trivial task and for this reason, in professional companies in the sector and in real situations, automated analysis tools are often used which make identification of vulnerabilities easier. These tools are very useful to developers and testers for writing and reviewing good code but we must remember the limitations that this type of tool entails, such as the possible fallacy of static analysis tools (they can provide false positives) and the little application coverage tested for dynamic analysis tools.
The application under analysis is *Befloral* (https://github.com/exSnake/Befloral), an e-commerce website for the sale of personalized flowers, plants, vases and bouquets developed by computer science students. In an e-commerce web application it is essential to first ensure that the features do not introduce basic errors or bugs accessible to any user so as not to undermine the integrity and reputation of the site. However, it is even more important to analyze in depth the specifications and the implementation at code level of the various functions to prevent a more expert and IT-aware user from exploiting weaknesses in the application to carry out malicious actions towards the owners of the site and towards other users.

Tools and Application installation

There are many vulnerability identification tools available for use and to conduct a complete analysis it is good practice to apply different types of tools, such as static analysis, dynamic analysis and those based on machine learning techniques. In particular, in this project SpotBugs and Find-Sec-Bugs were used as static analysis tools, and ZAP with the related plugins for the dynamic analysis part.

A virtual machine with the Ubuntu 22.04 operating system was used as the development and analysis environment. In order to simulate the activity of the backend server and perform the necessary analyses, it was necessary to install Eclipse IDE, a multi-language and multi-platform integrated development environment. In order to work with the proposed project, it was necessary to set up mySQL, the Apache Tomcat server and import the *Befloral* project with all the necessary procedures. Although the installation and setup of mySQL and tomcat were quite immediate and linear, I encountered some unpleasant problems when importing and using the *Befloral* web application. In particular, every time the Virtual Machine was restarted, the application no longer appeared within the eclipse IDE, although it was present in the current workspace. It was therefore necessary to remove the *Befloral* project and re-import and install it every time the VM was restarted. However, while it was

an unpleasant task to repeat, there were no further problems when using the web application and vulnerability analysis tools. I therefore believe that this problem was due to some specific setting of the virtual machine.

As regards the analysis tools used, SpotBugs is a program that uses static analysis to look for errors and/or bugs within Java code. It is a project built on top of FindBugs and can be used via integration with the eclipse development environment. What makes it a static analysis tool is the fact that it works by searching for "bug patterns" within the code that probably lead to unwanted behavior of the application. Installation within eclipse is made simple by the integrated marketplace, while the "Find-Security-Bugs" plugin, used in this analysis, must be installed via the web from the appropriate site. Since SpotBugs offers a wide range of insights into the analyzed code, the tool has been set to only report bugs related to "Malicious code vulnerability" and "Security" with a minimum confidence to report set to medium.
The dynamic analysis tool is ZAP, or a vulnerability scanner. Typically a vulnerability scanner is a detection tool used to carry out penetration testing and vulnerability assessments and the advantage in their use is that they do not depend on the programming language used and do not require knowledge of the internal structure of the code or of the bugs present, given that they operate by launching attacks on an application and observing the responses.

In particular, ZAP (Zed Attack Proxy) operates in three phases, Explorer, attack and report, and functions as an intermediary "man-in-the-middle" proxy. It scrutinizes messages traversing between the browser and the web application, employing a dual-spider mechanism: one for traditional link discovery and the other tailored for handling asynchronous requests. By meticulously recording and scrutinizing requests and responses, ZAP unearths potential vulnerabilities, constructing a comprehensive map of the application's architecture.
Leveraging a blend of passive and active scanning techniques, ZAP serves as a robust asset in automated vulnerability management. However, it's crucial to complement automated scans with manual assessments to ensure thorough investigation and remediation of identified vulnerabilities.
Among the plugins used is "Fuzzer", designed to address the challenge of generating program executions that lead to the discovery of new vulnerabilities. Using fuzzing techniques, the plugin generates random or model-based inputs, while also using knowledge and evolution approaches to guide input generation more intelligently.

2. Testing process

In order to carry out consistent and comprehensive application testing, it is necessary to adopt an effective testing activity. In the case under analysis here, the testing activity was divided between different parts, each of which contributed to revealing bugs or possible vulnerabilities in the application under examination. The simplest testing activity is that conducted by any user who navigates an application in a "standard" way; this type of activity, although it might seem trivial, is of fundamental importance since it can help reveal the vulnerabilities (if present) that are more easily attacked and which could therefore cause more damage to the application.

The second analysis activity of the application conducted is the most laborious and least trivial, given that it is assumed that the average user who uses an application does not go and test it in detail with specific security tools. This testing phase in fact involves the use of code analysis tools (static analysis) and behavioral analysis of the application in certain conditions (dynamic analysis). An in-depth analysis of this type helps to discover those vulnerabilities that are difficult for the average user of an application to access, but which a computer security expert could find and exploit for his possible malicious intentions. Therefore it is of fundamental importance to carry out a detailed analysis of this type to avoid damage to the application and to all users who could fall victim to malicious cyber attacks.

Static Analysis
As regards the static analysis part of the application, the tool used is SpotBugs, which is a means of searching for errors and/or bugs in the java code of an application. After conveniently installing it via the eclipse ide marketplace and after configuring the related "FInd-security-bugs" plugin, this tool analyzes the code in search of "code patterns" attributable to vulnerabilities or security problems (as set during its configuration). Just noting the presence of a (possible) vulnerability, however, is not enough. The objective is in fact to verify whether the vulnerability is actually present (given that these static analysis tools can return false positives), understand what vulnerability it is, look for possible attack vectors or exploitation actions that aim to exploit the vulnerability, find the code snippet responsible for the vulnerability and possibly understand how to fix it.
All of these steps are of fundamental importance in order to effectively analyze the level of security of an application and were implemented during the testing phase of the *Befloral* application.

In particular, SpotBugs revealed 4 vulnerabilities considered critical at the level of static code analysis, all 4 of which are part of the OWASP top 10, a standard awareness document that lists the 10 most critical IT security vulnerabilities in web applications. These reported vulnerabilities are: Potential Path Traversal (A01 - Broken access control), JDBC Injection (A03 - Injection), Unvalidated Redirect (A10 - Server Side Request Forgery), XSS in Servlet (A03 - Injection (A07:2017-XSS)).

Dynamic Analysis

As regards the dynamic analysis part of the application, the tool used is ZAP, which is a means of finding behavioral flaws within the application and is often used to perform penetration testing and brings with it the advantage that it does not require knowledge about the code or internal structure of the program under analysis. Even in this case, simply noting the reported vulnerability is not enough to be able to classify it as such and it is therefore necessary to verify it, understand it better, find the piece of code responsible and understand how a weakness of this type could be prevented.

The installation of the ZAP scanner and the use of the program and its plugins are simple and intuitive and, as in the case of static analysis tools, report the vulnerabilities found with a certain confidence.

In particular, there are 6 vulnerabilities identified through the use of ZAP: Absence of Anti-CSRF Tokens (A01 - Broken access control), Content Security Policy (CSP) header not set (A05 - Security misconfiguration), Cross-Domain misconfiguration (A01 - Broken access control), Missing Anti-clickjacking header (A05 - Security misconfiguration), Session ID in URL rewrite (A01 - Broken access control), Vulnerable JS Library (A06 - Vulnerable and Outdated components).

3. Collected data and results

The analysis conducted using the application's analysis tools reported a large number of possible vulnerabilities present, however not all the results received confirmed an actual vulnerability. This may be due to the fact that it is possible that project analysis tools give rise to vulnerabilities that cannot actually be attacked by users, or that I do not have the knowledge and skills necessary to fully exploit the present vulnerability.
Here is summarized the most useful information regarding the vulnerabilities identified with the techniques applied.

Collected data summary

Type of analysis	Vulnerability	OWASP Top-10	CWE	Exploitation time	True/False positive
Static: SpotBugs	Potential Path Traversal	A01	CWE-35	few minutes	False
Static: SpotBugs	JDBC Injection	A03	CWE-89	~30 minutes	True
Static: SpotBugs	Unvalidated Redirect	A10	CWE-601 or CWE-938	/	/
Static: SpotBugs	XSS in Servlet	A03	CWE-79	few minutes	False
Static: SpotBugs	Potential injection	A03	CWE-20	~20 minutes	True
Dynamic: ZAP	Absence of Anti-CSRF Tokens	A01	CWE-352	Automatic	True
Dynamic: ZAP	Content Security Policy (CSP) header not set	A05	CWE-693	Automatic	True
Dynamic: ZAP	Cross-Domain misconfiguration	A01	CWE-264	Automatic	True
Dynamic: ZAP	Missing Anti-clickjacking header	A05	CWE-1021	Automatic	True
Dynamic: ZAP	Session ID in URL rewrite	A01	CWE-200	Automatic	True

Dynamic: ZAP	Vulnerable JS Library	A06	CWE-829	Automatic	True
Static + Dynamic	SQL Injection	A03	CEW-03	Automatic	True

Static Analysis data

From the table you can see how the vulnerabilities listed and analyzed by the project's static analysis tool "SpotBugs" are 5, belonging to different vulnerability categories. However, it is important to point out that SpotBugs actually reported a total of 214 possible bugs or security issues that give rise to 12 different possible vulnerabilities. Of these 11 vulnerabilities, 5 referred to those identified with a medium confidence and deemed "frightening", another 5 to those identified with a low confidence and deemed "troubling", while only one vulnerability was identified with low confidence and deemed "of concern". Of the total 11 potential vulnerabilities identified, 5 of them have been analyzed and attempts at exploitation have been made. Due to the complexity of the task and the presumed non-actual presence of the vulnerability in the indicated position, of the 5 alerts indicated, actual success in exploiting the vulnerability was only possible in 2 of them. In both of these cases the procedure was not trivial and penetration attempts were made based on the knowledge already possessed and on information sought online.
In particular, observing the XLS file delivered together with this report, it can be seen how vulnerability number 2 can be exploited by manipulating the string passed as an argument to the "sort" attribute of the reported URL (even if the attempt was unsuccessful) and this is a typical case of injection. Vulnerability number 5, on the other hand, is a classic example of Cross-site-scripting (XSS) in which an input parameter is not validated correctly and can cause the execution of a script by the user who falls victim to it.
Both of these vulnerabilities are therefore considered True positive cases (i.e. critical issues identified as true which effectively give rise to vulnerabilities) and on average approximately 30 minutes were spent identifying and attempting to exploit each one of them.

Dynamic Analysis data

Similarly, the vulnerabilities identified by the dynamic analysis tools were also numerous, however only those considered to have a level of criticality and confidence above a certain threshold were reported. There are therefore 6 potential relevant vulnerabilities, each of which was automatically identified by the ZAP web scanner tool. Due to the dynamic nature of this tool, all the vulnerabilities identified by it are to be considered true positives given that it does not look for potentially critical patterns within the code but analyzes the behavior of the application during its execution.
Most of these vulnerabilities have been identified in files internal to the web application server and are therefore not easily analyzable, while others are due to jsp files or tag files contained in the project.
For each of the potential vulnerabilities reported, specific online searches were carried out on the topic to understand how a possible exploitation action could be implemented or an attack

vector identified. In particular, for vulnerability number 10 reported in the XLS file, a penetration attempt was made which is based on the interception by an attacker of a user's session id and its reuse to carry out malicious actions through a simple substitution within the web application URL.

4. Conclusions

After having used and analyzed in detail the most important aspects regarding the IT security of the *Befloral* web application, it is possible to draw conclusions on the security level of the project.

Final considerations

If from a design and functional point of view the application is developed in an excellent way, with a good implementation of features and a good user experience, the security aspect is in my opinion *Befloral*'s weak point. I would define the security level of the application as medium-low level, given that, although there are no serious problems easily accessible by a novice web user, significant problems and vulnerabilities persist which put the integrity and availability of the application at risk. the application and the users who use it. This is probably due to a suboptimal procedure for defining security requirements, developing code and verifying requirements.
If I were the owner and responsible for managing the web application and I was aware of the critical issues found, I would not be comfortable keeping the site active, given that this could cause damage to the service and the users who use it. Although absolute security is not well definable, I would turn to an IT security expert to analyze the most relevant critical issues and resolve them in order to obtain an acceptable level of application security.

From a technical point of view, the data reported in the analysis section are not reassuring, given that they show how the application is subject to the risk of a SQL injection attack, cross site scripting, session hijacking etc..., although not for all vulnerabilities listed it was possible to find a working attack vector.
Although the value of the site's assets and resources may not be of great importance (for a marketplace, critical information in terms of security impacts the service and its reputation more than the users) even if only for the fact that for some vulnerabilities there is no that it took more than 30 minutes to exploit them and that potentially many others do, the site managers should consider the option of turning to experts in the sector.

Critical analysis

Regarding the analysis tools used, the use experience was good overall. The greatest difficulties were initially encountered on the virtual machine side, which sometimes gave problems in saving the internal state of the workspace and it happened that the project had to be reinstalled several times, a problem which was then resolved by updating system settings. On the analysis side, however, the setup of the Eclipse IDE development environment, the application and the analysis tools was simple and linear and their use, although complex at first glance, proved to be convenient. The usefulness of analysis tools such as SpotBugs and ZAP is undoubtedly high and certainly helps a security tester and maximizes the quality of the analysis conducted, despite not spending too much time.

Personal reflections

Before attending this course and undertaking this security analysis experience I had never given too much importance to many aspects also addressed during this report. This is also due to the fact that personal projects and previous development experiences were more focused on writing code for data analysis algorithms and the projects were not expected to be used by a large number of people.

However, now I pay much more attention to security aspects every time I interact with computer systems and ask myself questions about the implementation of certain features. I therefore found the entire experience of the course and the analysis undertaken very useful and educational and I really appreciated the fact that I addressed the topics both from a theoretical point of view, exploring a vast range of the most frequent vulnerabilities, and from a practical one, with the opportunity to use concrete and effective analysis tools first-hand.

In conclusion, I believe the entire experience was very useful for me for my academic career and for future work experiences and I make a personal commitment to further delve into the topics covered and the food for thought I received.