Project course Security Testing 2024/2025

Vulnerabilities identification and exploitation

In the course, we have seen that testing the security aspects of an application is not always an easy task but that, by applying secure coding rules, developers can increase the security degree of the developed applications. We have also seen that the support of automated tools is fundamental for security testing, aiming at early revealing bugs leading to security issues. Tools that support static and dynamic analysis can be quite effective in discovering vulnerabilities but have specific limitations. For instance, static analysis can produce a quite large number of false positive alerts, while dynamic analysis can explore only partially the application.

(1) Context of the project

We are considering a scenario in which we are part of a software house with two teams:

a) DEVELOPERS team – the team composed by junior developers that develop Web applications by using Java Servlets;
b) SECURITY TESTERS team - the team ("participants to the Security Testing course 2024/2025") composed of testers that test security aspects of the applications developed by DEVELOPERS and give advices on how to improve the security degree of the developed applications.

Request: max two persons per team
Report: The composition of the team needs to be declared

(2) Objective of the project

To conduct a realistic experience on vulnerabilities identification and exploitation.

(2) Task and steps

Given a web application developed by the DEVELOPERS team, the task consists in installing it locally on your PC and, by using the testing tools (we have seen in the course), identify and exploit the vulnerabilities of the applications. The following the steps need to be applied:

- (step 1) Fill out the pre-task questionnaire (see below the pre-task questionnaire section)
- (step 2) Install the application (App)
- (step 3) Test the application App with the support of selected testing tool(s)
  - (step 3.a) Apply the selected testing tools and get the list of the identified potential vulnerabilities
  - (step 3.b) Inspect the results obtained from the tools, apply code review (manual code review or eventually code review supported by other specific tools), for checking the identified potential vulnerabilities and classifying them us TRUE or FALSE POSITIVE cases (i.e., real and non-real vulnerabilities among the ones suggested by the tool/s).
- (step 4) Test the application App manually, to look for vulnerabilities undetected by the tool(s)
- (step 5) Fill out the post-task questionnaire (see below the post-task questionnaire section)

See below for the list of App and Testing Tool(s) to use.

Report: The adopted testing must be clearly presented (in terms of process steps, sequence, input and output of each step) and motivated in the project report.

NOTE: It is not required to copy and paste in the report (or in the XLS file) the results of the tool. It is not enough to have a positive evaluation for the project.

NOTE: Please, remember that the quality of the conducted analysis and of the reports produced is more important than just listing a high number of vulnerabilities. It is not required to inspect and identify all vulnerabilities in the code.

Request:

(1) Run at least two testing tools (see below the mandatory tools)
(2) document all the vulnerabilities identified by the tools;
(3) inspect in detail at least up to 5 (different types) of those vulnerabilities, by classifying them as True/False Positive and listing possible exploitations (i.e., attack vectors showing the vulnerabilities);
(4) conduct an extensive manual test of the applications to identify not discovered vulnerabilities (document it even if no new vulnerabilities are discovered)

Minimum accepted:

(1) run two testing tools (see below which one is mandatory to apply);
(2) document only the vulnerabilities identified by the tool that we have seen in the course (e.g., SQL injection, XSS, etc);
(3) inspect 3 (different types) of those vulnerabilities, by classifying them as True/False Positive and listing at least one possible exploitation (i.e., attack vectors showing the vulnerabilities) for each of the three vulnerabilities;
(4) conduct a manual test of the applications to identify not discovered vulnerabilities (document the activities even if no new vulnerabilities are discovered)

(3) Applications

The Eclipse project of Web application target of the task are provided by the teacher via Moodle.

Together with the Eclipse project, instructions on how to import the project in Eclipse and install the application are also provided.

The target application is a Web application really developed by junior developers (university students of another university): **Tourism**.

Note. Please, note that names and details of developers can refer to existing person, so be carefully in using this information. However, be aware that the application are distributed in GitHub. However, please, use the provided Eclipse projects for the task.

(4) Testing tools to use

As testing tools, all tools used in the course Security Testing 2024/2025 can be used, in particular, the request is to conduct both static and dynamic analysis:

- Static tool, e.g., **SpotBugs (+ Find-Sec-Bugs)**
  - Instructions and examples about how to install and use it can be found in the Moodle of the course.
- Dynamic tool, e.g., **Zap web scanner** (with plugins such as **Fuzzer** and **FuzzDBFiles** and **FuzzDBOffensive**)
  - Instructions and examples on how to install and use it can be found in the Moodle of the course.

Any other tools can be used as supporting tool (= only if **SpotBugs+Find-Sec-Bugs** and ZAP have been used). Extra tools, i.e., tools not used in the course can be used only for step 4 ("Manual Analysis") of the section "(2) Task and steps", and must be introduced and motivated in the report.

Report: Each tool used must be documented (briefly presented) and motivated in the project report.

Request: apply SpotBugs+Find-Sec-Bugs and ZAP (& manual inspection) to the tested application
Minimum accepted: use of only the static tool, SpotBugs+Find-Sec-Bugs, (& manual inspection)

(5) We expect to receive (output produced)

We expect to receive:

- a **PDF document**, that reports and comments on the conducted testing activity and that is written according to the given guidelines (see below); and
- a supporting **XLS file** with data referenced and commented in the document itself.

The expected document must contain the following sections (a *PDF template* is provided):

- Brief introduction to the task
  - Presentation of the conducted activities
  - Presentation of the tested app
  - Presentation and motivation of the used tool(s), with details on their configuration
- Explanation of the adopted testing process,
  - Introduction of the overall adopted testing process
  - Details of each step conducted, of the tools used in each step etc.
  - Explaining how the activities have been divided among the testing team members
- Summary of the collected data (results)
  - Brief summary of the results: potential vulnerabilities identified by each used tool and by the manual analysis, types of vulnerabilities identified, report on TRUE/FALSE POSITIVE (overall and by vulnerability types), etc.
  - Total and average time spent in the tool running, in the vulnerability analysis and exploitation, etc.
  - The whole set of collected data are then fully reported in the related XLS file
- Discussion of the collected results and of missing vulnerabilities
  - Brief evaluation of the security degree of the tested application
  - Feedback on the used tools
  - Feedback on the experience

The XLS file must contain the following data (an *XSL Template* is provided):

- Pre-task questionnaire filled
  - Questions on experience degree in programming / testing /security
- Results
  - report of the results: used tool, vulnerability suggested by the tool, True or False positive, exploitation / attack vectors
- Post-task questionnaire filled
  - Feedback about the experience done