Project Phase 1 (2025)

Teams

Throughout this semester, you will perform and document a technical security review of OpenEMR (https://www.open-emr.org/). This open-source system provides electronic healthcare functionality for medical practices. Security and privacy are concerns for both medical practices and patients.

Software:

This year's project will involve an extensive analysis of OpenEMR 7.0.3. Each student will need to download the project's source code so that you can see the code, run tools against the code, edit the code, and run the code. You will also need to install OpenEMR. This document may be helpful for some guidance on installation in a Docker image.

We are using 7.03, a development branch (at the suggestion of OpenEMR), which updates daily. Use this source code (downloaded 1/24/25), so we are all using the same source code.

Deliverables:

Submit one PDF for all parts of the assignment in Gradescope. **Only one submission per team.**

0. Black Box Test Cases

For Parts 2 and 3 (ZAP), you will write black box test cases. For each test case, you must specify:

1. A unique test case ID that maps to the ASVS, sticking to Level 1 and Level 2. Provide the name/description of the ASVS control. The ASVS number should be part of the one unique identifier.
2. Detailed and repeatable (the same steps could be done by anyone who reads the instructions) instructions for how to execute the test case
3. Expected results when running the test case. A passing test case would indicate a secure system.
4. Actual results of running the test case.
5. Indicate the CWE (number and name) for the vulnerability you are testing for [which should be in the ASVS document].

1. Vulnerable Dependencies (40 points)

In this assignment, you will run two secure compositional analysis (SCA) tools on OpenEMR.

Pick two of the following tools:

1. GitHub’s checker
2. Snyk

3. OWASP Dependency Check (see Workshop #8)

Instructions on how to run these tools: Running Secure Compositional Analysis tools

Run two of these tools (your choice) on OpenEMR.

(18 points per tool) Report the results for each tool. The results should contain:

1. (2 points) The number of total vulnerable dependencies. Download and include the report from the tool.
2. (1.8 points per dependency) For up to 10 different vulnerable dependencies (report all for tools that report 10 or less, randomly choose 10 for tools that report more than 10), for each of the 10 dependencies, provide the following:
   - (1 point) The list of CVEs.
   - (.5 point) Determine if the dependency is **direct** or **transitive** to OpenEMR. A direct dependency is a dependency that is directly accessed by a project. e.g., through declaration in the pom.xml file. A transitive dependency is a dependency of any direct dependency (or the dependency of a dependency's dependency), e.g., not declared in the pom.xml file. However, a build tool like Maven can still determine and download these dependencies for a successful build.
   - (.3 point) Determine if the vulnerable dependency has a safer version available and provide the version number.

(4 points) Explain why you think the results differ among the two tools and write a comparison report. What do you think are the strengths and weaknesses of each tool from both technical and usability standpoints.)

Run two vulnerable SCA tools on OpenEMR

(18 points per tool)

- (2 points) The number of total vulnerable dependencies and a download of all the output from the tool.

For each of the 10 dependencies, the results should contain

- (1 point) The list of CVEs
- (.3 points) Whether the dependency is a **direct** or **transitive** dependency.
- (.5 points) The version number for a safer version, if available. “Not available” if no safer version is available.

Task 2 (4 points)

(2 points) Explain why you think the results differ among the tools.

(2 points) What do you think are the strengths and weaknesses of each tool, 1 point per tool

~~(2 points) Overall: Your report may explain why a certain tool missed a vulnerable dependency that another tool has detected. What do you think are the strengths and weaknesses of eac~~h tool)

2. ZAP Client-sode bypassing (30 points)

[See Workshop 7 for a reminder] … also see this example

1. Plan 5 black box test cases (using the format provided in Part 0 above) in which you stop user input in OpenEMR with ZAP and change the input string to an attack. (Consider using the strings that can be found in the ZAP rulesets, such as jbrofuzz.) Use these instructions as a guide.
   - In your test case, be sure to document the page URL, the input field, the initial user input, and the malicious input. Describe what "filler" information is used for the rest of the fields on the page (if necessary). Be repeatable!
   - Remember, the expected result for a passing test case would be that OpenEMR is secure.
   - Run the test case and document the actual results.

Client-side bypassing: For each of 5 test cases (6 points per test case):

- (1 points) A unique test case id that maps to the ASVS, sticking to Level 1 and Level 2. Provide the name/description of the ASVS control. Only one unique identifier is needed (as opposed to the example in the lecture slides). The ASVS number should be part of the one unique identifier.   Find ASVS controls that relate to user input.
- (2 points) Detailed and repeatable (the same steps could be done by anyone who reads the instructions) instructions for how to execute the test case

  * 1 point Anyone who ran the test case would do exactly the same thing ... exactly. Specific test input and steps are provided. You do not get points if generalized instructions are provided that indicate what should be done, but different people might do slightly different things.

  * 1 point Instructions demonstrate a solid understanding of the vulnerability that involves user input that could be changed in ZAP.
- (1 point) Expected results when running the test case. A passing test case would indicate a secure system.
- (1 point) Actual results of running the test case.
- (1 point) Indicate the CWE (number and name) for the vulnerability you are testing for. Hint: The CWE number is provided in the ASVS document.

3. ZAP Fuzzing (30 points)

[See Fuzzing in Workshop 7 for a reminder]

1. Use the five client-side bypassing test cases from the client-side bypassing (Part 2) for this exercise.

2. Use the jbrofuzz rulesets to perform a fuzzing exercise on OpenEMR with the following vulnerability types checked: Injection, Buffer Overflow, XSS, and SQL Injection.
        - Take a screenshot of ZAP information on the five test cases.
3. Report the ruleset you chose for each vulnerability type, along with the results and what you believe the team would need to do to fix any five true positive vulnerabilities you find. If you don't find any vulnerabilities, provide your reasoning as to why that was the case, and describe what mitigations the team must have in place such that there are no vulnerabilities.

- (3 points per test case) Take a screenshot of the ZAP output for running the fuzzing on the 5 test cases.

This part assumes you found 5 true positive vulnerabilities.

- (3 points per each of the 5 true positive vulnerabilities)
    - (1 point) Report the ruleset (Injection, Buffer Overflow, XSS, SQL Injection) that yielded the vulnerability
    - (2 points) The results and instructions on how to conduct an attack using the vulnerability

This part is if you ran ZAP on all five test cases with the Injection, Buffer Overflow, XSS, SQL Injection checkboxes checked; and the tool did not provide you evidence of 5 true positive vulnerabilities.

- (10 points or 2 points*number of vulnerabilities less than 5 you are justifying can't be found) If you find less than 5 true positive vulnerabilities, provide your reasoning strongly as to why that was the case, and what mitigations strategy (explain well) the team must have in place such that there are no (or few) vulnerabilities. You would need to justify why you should get the points -- particularly given that other teams may have no problem finding 5.

4. Team member work distribution

Provide a table with each team member's name and which part(s) of the submission they worked on.

It is essential that every team member does their part in completing the project iterations. Your grade may be multiplied by the percentage of effort you put into the project relative to how much you should have been for all the work to be equally distributed. For example, if it is determined that you did 10% of what you should have done to pull your weight with the team, your grade on the iteration will be (.1)(the grade your team earned) such as (.1)(100) = 10%. The message is -- do your part, for that is how you learn, and that is how you are a good team member. Let's keep it simple, and everyone on the team do their part.

Note: For any part of this course-long project, you may not directly copy materials from other sources. You need to adapt and make it unique to OpenEMR. You should provide references to your sources. Copying materials without attribution is plagiarism and will be treated as an academic integrity violation.

Laurie Williams | Email: lawilli3@ncsu.edu

Last updated 2024