Poznan University of Technology
Faculty of Computing and Telecommunications

michal.apolinarski[at]put.poznan.pl

Course: Application Security – laboratories

Lecturer: Michał Apolinarski, Ph.D.

Topic: Security audit – black-box approach

Duration (on site): 180 min.

PREREQUISITES:

Completion of previous laboratories involving the design and implementation of a web-based content service (e.g. Meme Service or a similar application), including authentication, session management, and role-based access control. Basic knowledge of web applications, databases, HTTP, and common web security issues.

GOALS:

The goal of this laboratory is to perform a security audit using a **black-box approach** on a selected web application. Students will:

- test an application **without access to its source code**,
- identify security vulnerabilities that may lead to unauthorized access, data exposure, or malfunction,
- apply automated and manual testing techniques,
- document findings in a structured security audit report.

INSTRUCTIONS (tasks for groups of up to 2 persons):

PART A – Target selection

1. Choose an application as the target of the security audit:
   - an application developed by another student group during this course,[1]
   - ~~an open-source application used in previous laboratories,~~
   - ~~your own application (only if none of the above options are available).~~

[1] Mutual testing between groups is prohibited ☺

2. The selected application must be approved by the lecturer.

PART B – Testing (manual and automated)

1. Perform a security audit of the selected web application using a black-box approach, without access to the source code or internal documentation.
2. Testing must combine manual techniques and automated tools, focusing on observable application behavior, input handling, access control, and error responses.
3. **Testing scope** should cover, where applicable:
   - authentication and session management behavior,
   - role-based access control and privilege separation,
   - ownership checks (e.g. deleting or modifying other users' content),
   - input validation and injection vulnerabilities (XSS, SQL injection),
   - file upload handling and content processing,
   - search functionality and parameter handling,
   - error handling and information disclosure.
4. **Manual testing** should include attempts to:
   - access restricted functionality without proper authorization (testing unauthorized actions),
   - perform actions as another user (broken access control),
   - manipulate request parameters and identifiers,
   - bypass client-side restrictions,
   - basic XSS, injection attempts, parameter tampering,
   - submit malicious input to comments, search fields or upload forms.
5. Students should rely on browser developer tools, HTTP request inspection, and controlled modification of requests.
6. **Automated tools** may be used to support the testing process, such as:
   - reconnaissance and analysis tools (e.g. Nmap),
   - web application scanners (e.g. OWASP ZAP, Burp Suite),
   - injection testing tools (e.g. SQLMap, XSStrike),
   - containerized or locally installed security testing tools.
7. Automated results must be manually reviewed. False positives should be clearly identified and explained in the report.
8. Students may perform testing using **testing environment and tools** like:
   - any standard operating system (Windows, macOS, Linux),

- browser-based testing environments,
- Kali Linux (native or virtualized),
- containerized tools (e.g. via Docker).

REPORT:

- Include a title page with full details of the student's group, course and exercise.
- Should be carefully edited and provide evidence of the completion of all exercises (screenshots, answers, and conclusions).
- A complete report must be submitted to the lecturer at least two days before the next class in which it will be presented.