English

# Universitetet i Bergen
## Det matematisk-naturvitenskapelige fakultet

| | | |
|---|---|---|
| Exam in | : | INF226 Software Security |
| Semester | : | Spring 2025 |
| Time | : | 09:00 – 12:00, 2nd of March 2025 |
| Number of pages | : | 8 |
| Permitted aids | : | none |

- This exam counts for 60% of your final grade in this course.
- Points on the exercises indicate *approximate* percentage weight on the total final grade.
- Give justifications for your answers, unless otherwise specified.
- Use precise language and make sure to read the exercises carefully.

## Exercise 1 (10 points)

Answer the questions below with one or two sentences on each.

a) [1.5 points] What is the purpose of Address Space Layout Randomization (ASLR)?
b) [1.5 points] What is the principle of least privilege and why is it important?
c) [1.5 points] What is the Same-Origin Policy and what does it protect against?
d) [1.5 points] Why should passwords never be stored in plaintext in a database?
e) [1.5 points] Why should validation of user input be done on the server side?
f) [2.5 points] Name three kinds of vulnerabilitites that can be caused by concatenating user input into strings.

## Exercise 2 (12 points)

In this exercise, we'll examine a vulnerability in InShare, the note-sharing web application which you are familiar with from the assignments. Below is an excerpt from the application showing how notes are created and displayed.

First, we have the view template (viewNote.html using Thymeleaf):

```
<main class="note-view">
    <h1 th:text="${note.name}">Note Title</h1>
    <p th:text="'Created: ' + ${#dates.format(note.created
                                              ,'dd MMM yyyy, HH:mm')}">
    </p>
    <p th:text="'Author: ' + ${note.author.username}"></p>
    <div class="note-content">
        <p th:utext="${note.content}"></p>
    </div>
</main>
```

And the relevant controller code:

```
@PostMapping("/create")
@Transactional
public String createNote(@RequestParam("name") String name,
                         @RequestParam("content") String content) {
    final Authentication authentication
            = SecurityContextHolder.getContext()
                                   .getAuthentication();
    if (authentication != null && authentication.isAuthenticated()
            && (authentication.getPrincipal() instanceof User)) {
        final User user = (User)authentication.getPrincipal();
        final Note newNote = new Note(user, name, content)
            .withUserPermission(user, Note.Permission.READ)
            .withUserPermission(user, Note.Permission.WRITE)
            .withUserPermission(user, Note.Permission.DELETE);
        newNote.save(jdbcTemplate);
        return "redirect:/note/edit/" + newNote.id.toString();
    }
    return "redirect:/";
}

@GetMapping("/view/{id}")
public String showViewForm(@PathVariable("id") UUID id, Model model) {
    Note note = Note.load(jdbcTemplate, id);
    model.addAttribute("note", note);
    return "viewNote";
}
```

a) [2 points] Identify and explain the cross-site scripting (XSS) vulnerability in this code. Be specific about where the vulnerability occurs and why it exists.

b) [2 points] Describe two distinct threats that an attacker could accomplish by exploiting this vulnerability. For each threat, explain what malicious JavaScript could do (no need to provide the actual code).

c) [2 points] What is the standard method for preventing XSS when outputting data to a webpage?

d) [1.5 points] The note content is intended to support HTML formatting (like <b>, <i>, <ul>, etc.) for rich text editing. Explain why this requirement makes it challenging to use the standard XSS prevention method you described in (c).

e) [1.5 points] Propose a solution to prevent XSS in this application while maintaining the requirement for HTML formatting in notes.

Content Security Policy (CSP) is a security feature that restricts the sources from which a web page can load content such as scripts, styles, and images. This allows website owners to control and limit how external resources are loaded and executed. It is enforced via an HTTP header or a tag, defining which domains are allowed to provide specific types of resources. CSP is deny-by-default: content from unapproved sources is blocked, and violations can be reported. In particular, inline JavaScript is denied.

Imagine that InShare has been modified to use CSP. All JavaScript code has been moved from inline scripts to a single file called dashboard.js, and the application uses the Quill rich text editor from a CDN. Here's the current CSP:

```
Content-Security-Policy:
    default-src 'none';
    script-src 'self' https://cdn.quilljs.com;
    style-src 'self' https://cdn.quilljs.com;
    img-src 'self';
    connect-src 'self';
    form-action 'self';
    frame-ancestors 'none';
```

f) [3 points] Explain how this CSP configuration makes it more difficult for an attacker to exploit the XSS vulnerability you identified in part (a). Be concrete and explain why the threats from (b) would be mitigated by this CSP.

## Exercise 3 (11 points)

In this exercise, we will explore operating system security features. We will focus on processes, and how they are isolated from each other. We will use Linux as an example.

a) [3 points] What prevents a process from reading the memory of another process directly via pointers?

Programs often need to communicate and share resources with each other. But at the same time the operating system must ensure the security of the system.

By default, processes on linux of the same UID are quite exposed to each other. For instance, a process can read the memory of another process of the same UID using the /proc filesystem: For each process, there is a file called /proc/<pid>/mem, with mode rw−−−−−−−− (mode 600). There is also the ptrace system call that allows a process to attach to another process and inspect its memory.

b) [4 points] Explain the threats posed by these features, in the event of an exploitable buffer overflow vulnerability in, for example, a web browser.

Seccomp can be used to filter system calls made by a process. This can be used to for instance prevent a process from making the ptrace system call, or opening new files.

c) [4 points] Suggest how using multiple processes and privilege dropping can be used to limit the damage of an exploitable buffer overflow vulnerability in a web browser, while still enabling the web browser to function properly, and for instance access the network and downloads folder – or even upload arbitrary user-chosen files when requested by the user.

## Exercise 4 (13 points)

Windizor, an offshore wind power company, operates large wind farms off the western coast of Norway. Their turbines are monitored and controlled through a web-based control panel that allows remote operators to adjust critical parameters of individual turbines.

Below is an excerpt from their remote control interface, showing the control panel for a single turbine:

```
<form action="/api/turbine/247/control" method="POST">
    <h2>Wind Turbine Control Panel – Unit #247</h2>

    <!-- Yaw Control -->
    <fieldset>
        <legend>Yaw Control</legend>
        <label>Target Heading (degrees):
            <input type="number"
                   name="yawTarget"
                   value="175"
                   min="0"
                   max="360">
        </label>
        <label>Auto-tracking:
            <select name="yawMode">
                <option value="auto">Enabled</option>
                <option value="manual">Manual Control</option>
            </select>
        </label>
    </fieldset>

    <!-- Pitch Control -->
    <fieldset>
        <legend>Blade Pitch Control</legend>
        <label>Collective Pitch (degrees):
            <input type="number"
                   name="pitchAngle"
                   value="0"
                   min="-5"
                   max="90">
        </label>
        <label>Operation Mode:
```

```
        <select name="pitchMode">
            <option value="normal">Normal Operation</option>
            <option value="storm">Storm Protection</option>
            <option value="maintenance">Maintenance</option>
        </select>
    </label>
</fieldset>

<!-- Brake Control -->
<fieldset>
    <legend>Brake Systems</legend>
    <label>Mechanical Brake:
        <select name="mechBrake">
            <option value="released">Released</option>
            <option value="engaged">Engaged</option>
        </select>
    </label>
</fieldset>

<!-- Power Generation -->
<fieldset>
    <legend>Power Generation</legend>
    <label>Cut-in Wind Speed (m/s):
        <input type="number"
               name="cutInSpeed"
               value="3.5"
               min="3"
               max="5"
               step="0.1">
    </label>
    <label>Cut-out Wind Speed (m/s):
        <input type="number"
               name="cutOutSpeed"
               value="25"
               min="20"
               max="30"
               step="0.1">
    </label>
</fieldset>

<!-- Emergency Controls -->
<fieldset>
    <legend>Emergency Controls</legend>
    <button type="submit"
            name="action"
```

```
                value="emergency"
                class="emergency">
            Emergency Shutdown
        </button>
    </fieldset>
</form>
```

Wind turbines of this size must be carefully controlled to prevent damage. Several combinations of settings can lead to catastrophic failures:

- At wind speeds above 25 m/s, blades must be "feathered" (rotated to 90°) to minimize load. Operating with blades at normal pitch (0°) in storm conditions subjects the blades to extreme forces, potentially potentially breaking them.
- Yaw control (turbine direction) is also sensitive. A 3.5MW turbine was destroyed in Denmark in 2008 when its yaw brake failed during a storm, causing the nacelle to spin uncontrollably. Rapid yaw adjustments while operating at high wind speeds creates torsional loads on the tower and bearings. A turbine facing 90° to the wind direction in storm conditions experiences maximum lateral force on the tower structure.
- The mechanical brake system is the last line of defense against rotor overspeed. Operating beyond the cut-out speed of 25 m/s without engaging brakes can lead to rotor speeds exceeding 18 RPM, risking generator burnout and blade separation.

The most severe scenarios combine multiple unsafe settings. During Storm Eunice in 2022, a competitor's turbine suffered catastrophic failure after operating for just 3 minutes with:

- Released mechanical brakes
- Blades pitched at 15° (partial power generation position)
- Nacelle yawed 75° from wind direction
- Wind speeds of 32 m/s

The resulting failure scattered debris across a 600m radius and required a complete turbine replacement costing 12.8 million euro.

Remote operators use this control panel throughout their 12-hour shifts while monitoring weather conditions, maintenance schedules, and power production statistics through various browser tabs and windows. During storm conditions, operators must monitor multiple data sources including weather radar, marine forecasts, and grid stability metrics.

a) [3 points] Assuming no additional security measures, what vulnerability may HTML forms, such as this remote control panel be open to, in the event that the operator visits other web sites using the same browser? Why?

b) [4 points] How would an attacker exploit this to damage the wind turbines? Be as concrete as you can.

c) [3 points] Explain how to prevent the vulnerability you discovered in a).

d) [3 points] What impact does the SameSite=Lax attribute have on the vulnerability? Would it be different if the form was transmitted using the GET method?

## Exercise 5 (14 points)

PHP is a popular server-side scripting language. It is used to create dynamic web pages and web applications. PHP allows inserting variables into string literals using double quotes (“). When a variable is placed inside double quotes, PHP automatically replaces it with its value. For example the following code will output”Hello, Alice!”

```
$name = "Alice";
echo "Hello, $name!";
```

However, when variables are directly inserted into SQL queries without proper escaping of the input, they can introduce SQL injection vulnerabilities.

The WordPress plugin “Search & Replace” allows administrators to search for specific text in the database and replace it with new text. This functionality is useful during website migrations or content updates.

A simplified version of the plugin’s code is shown below:

```
global $wpdb;

if (   isset($_GET['search'])
   && isset($_GET['replace'])
   && isset($_GET['select_tables'])) {
     $search = $_GET['search'];
     $replace = $_GET['replace'];
     $tables = $_GET['select_tables'];

     foreach ($tables as $table) {

          $query = "UPDATE $table
                    SET post_content = REPLACE( post_content
                                              , '$search'
                                              , '$replace')";
          $wpdb->query($query);
     }
}
```

a) [3 points] Explain why this code is vulnerable to SQL injection.

b) [2 points]This vulnerability was discovered in 2024 and assigned the CVE number CVE-2024-4145. Explain what CVE numbers are used for.

A normal, valid URL for using this plugin would look like:

```
https://wp.example.org/wp-admin/tools.php?page=search-replace
                                          &search=old-domain.com
                                          &replace=new-domain.com
                                          &select_tables[]=wp_posts
                                          &select_tables[]=wp_options
```

c) [3 points] Construct a malicious URL (starting with https://wp.example.org/) that an attacker could use to delete an important WordPress table (e.g., wp_users or another critical table). Explain how your exploit works.

### CVSS 3.1 Score

The Common Vulnerability Scoring System (CVSS) 3.1 measures security flaws based on severity. Consider the CVSS base score metrics and fill out the vector string for this SQL injection vulnerability.

Consider these metrics:

- **Access Vector (AV)**: How is the vulnerability accessed?
  - Network (N): Remotely exploitable
  - Adjacent (A): Requires local network access
  - Local (L): Requires local access
  - Physical (P): Requires physical access
- **Attack Complexity (AC)**: How difficult is it to exploit?
  - Low (L): No special conditions needed
  - High (H): Special conditions required
- **Privileges Required (PR)**: What level of user privileges are needed?
  - None (N): No privileges required
  - Low (L): Basic user privileges required
  - High (H): Administrative privileges required
- **User Interaction (UI)**: Does the attack require a user to take action?
  - None (N): No user interaction needed
  - Required (R): User interaction required
- **Confidentiality (C), Integrity (I), Availability (A)**: for eac of the CIA triad, the impact is rated:
  - None (N): No impact
  - Low (L): Limited impact
  - High (H): Significant impact

d) [3 points] Write a CVSS vector for CVE-2024-4145 on the format below, and explain your reasoning:

AV : [ ? ] / AC : [ ? ] / PR : [ ? ] / UI : [ ? ] / C : [ ? ] / I : [ ? ] / A : [ ? ]

e) [3 points] Explain what you think the severity of this vulnerability is, and why.