English

Universitetet i Bergen
Det matematisk-naturvitenskapelige fakultet

Exam in	:	INF226 Software Security
Semester	:	Spring 2021
Time	:	09:00 – 12:00, 17th of February 2021
Number of pages	:	6
Permitted aids	:	Open-book exam

- 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 (12 points)

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

a) How can non-executable stack prevent an attacker from exploiting a buffer overflow using shell code?
b) How can the *SameSite* flag on a cookie prevent a cross-site request forgery attack?
c) Why should untrusted data not be inserted directly into the HTML code on a website?
d) Why do we add salt to a key derivation function when implementing password based authentication?
e) Which characters must be escaped in order to prevent XSS when inserting data into an HTML element content? For example in: `<div>DATA</div>`
f) What would be the consequence if capabilities fail to be unforgeable (in a capability based access control system)?

Exercice 2 (14 points)

A small factory producing specialised parts for the fishing industry has hired you as a consultant for helping them imporove their IT systems. One of the issues Susan, the manager, brings up is that their access control system is too tedious.

In the factory every worker has an access card which is used when entering the building, using certain machines or getting supplies. The same kind of key cards are used also for the cleaning people who come in after hours to clean.

The rules for the access cards use an *access control list* (ACL), which Susan updates. You can see the permissions list in Figure A on the next page.

For instance, it says that Alice, because she is a trainee, can only use Machine C. Ayla, however, is a regular worker who can use all three machines. All workers and trainees can enter during regular hours – not out-of-hours. But cleaning people, such as Matthew, can only enter out-of-hours. And so on...

Susan's complaint is that it is very tedious to update the list with a lot of permissions every time someone comes or leaves – not to mention going through the permissions of all employees whenever they get a new machine which some of the workers need access to.

When you hear the manager's complaint, you immediately think of *role-based access control* (RBAC), and suggest that it may help making her job of updating the list less tedious.

a) Explain the difference between role-based access control and access control lists.
b) Reading from the access control list (Figure A), who has access to the machine supply room, but not the office?
c) Find a set of access control roles for the factory. Use the access control list given, and re-write it using the roles you have found – assigning roles to people and permissions to roles. *Hint: Five roles suffices.*
d) Explain the advantage of using RBAC as opposed to ACL using the operations below as examples. Could these simplifications improve security? If so, how?

   - Adding a new regular worker.
   - Giving all workers access to a new machine.
   - Upgrading a trainee to a regular worker.

Person	Permission
Alice	Enter factory in hours
Alice	Use Machine C
Ayla	Enter factory in hours
Ayla	Use machine A
Ayla	Use machine B
Ayla	Use machine C
Ayla	Access machine supply room
Bob	Enter factory in hours
Bob	Use Machine C
Chaitanya	Enter factory in hours
Chaitanya	Use machine A
Chaitanya	Use machine B
Chaitanya	Use machine C
Chaitanya	Access machine supply room
Espen	Enter factory in hours
Espen	Use machine A
Espen	Use machine B
Espen	Use machine C
Espen	Access machine supply room
Lene	Enter factory in hours
Lene	Use machine A
Lene	Use machine B
Lene	Use machine C
Lene	Access machine supply room
Lene	Enter office
Mark	Enter factory in hours
Mark	Enter office
Matthew	Enter factory out of hours
Steve	Enter factory out of hours
Susan	Enter factory out of hours
Susan	Enter factory in hours
Susan	Enter office
Susan	Access machine supply room
Xing	Enter factory in hours
Xing	Use machine A
Xing	Use machine B
Xing	Use machine C
Xing	Access machine supply room

Figure A: The access control list for the factory.

Exercice 3 (12 points)

Imagine that you are working as developer for an environmental organisation collecting sightings from the public of endangered species. The system has been in use for many years and the back-end is written in C, but have developed a new front end in Java.

During testing, one of your testers reports that the system crashes when given large input to one of the fields. You trace the error to the back-end, and in particular the following function:

```
struct datapoint {
   char species[1024];
   int datecode;
}

(...)

void read_data_point() {
   datapoint p;
   gets(p.species);
   scanf("%d", &p.datecode);
   save_datapoint(p);
}
```

a) Explain why the above code could lead to a vulnerability.

Imagine an attacker would want to exploit this vulnerability using a shell code exploit.

b) What parts would the attack payload consist of?
c) Where would the attacker want to point the function return pointer to?
d) What is the purpose of the NOOP-sled?
e) How can stack canaries make it more difficult for an attacker to exploit this vulnerability?

Exercice 4 (12 points)

You are hired as a security consultant for a small business, who has a website where customers fill in orders.

The website is developed and maintained by two in-house IT experts, who wrote it using Java Server Pages, served by Tomcat. While proficient at programming, they have no special competence in software security.

After some initial investigation you run static analysis on their code base, and it points out the following code in the handling of the form where users fill out orders:

```
String query = "INSERT INTO order (userid,itemid,address) "
             + "VALUES(" + currentUser + "," + itemId + ","
                         + deliveryAddress + ");"
```

```
            + "SELECT orderid FROM order WHERE userid='"
            + currentUser+"';";

Statement stmt = connection.createStatement();
stmt.execute(query);
ResultSet rs = stmt.getResult();
(...) // Code which displays all the current orders to the user
```

a) What kind of vulnerability is likely to occur in the code above?
b) How would you test wheather there is a vulnerability in the code above? Be as specific as you can.

By further investigating the logs, you discover that a malicious user has used the vulnerability to download the entire order history of the business, including the personal information of the business' customers.

b) According to the General Data Protection Regulation (GDPR), what are the responsibilities of the business after the breach?

After you pointed out the problem in the above code, the two in-house developers write a new version:

```
String query = "INSERT INTO order (userid,itemid,address) "
            + "VALUES(" + currentUser + "," + itemId + ","
                        + deliveryAddress ");"
            + "SELECT orderid FROM order WHERE userid='"
            + currentUser+"';";

PreparedStatement stmt = connection.prepareStatement(query);
stmt.execute();
ResultSet rs = stmt.getResult();
(...) // Code which displays all the current orders to the user
```

d) Is the new version secure? Why/why not?
e) Write a secure version of the code above. (Remember to format the code in your answer, to make it readable.)

Exercise 5 (10 points)

1. Explain in short what the Common Vulnerabilities and Exposures (CVE) database is. What information is stored there? Why is it useful?

2. Which other security resources are usually referenced when discussing a software vulnerability?

`sudo` is a program which allows authorized users to execute commands as a different user on Linux. Typically this is used by adminstrative users to gain *root* priviledges, for instance to install new software on the system.

In January a vulnerability was discovered in sudo, with identifier **CVE-2021-3156**.

3. Look this identifier up online in the National Vulnerability Database. **Explain in your own words what you understand the vulnerability in `sudo` to be. Describe its severity and how it was fixed.**