NTNU

Department of (Computer and Information Science)

Examination paper for (TDT4237) (Software Security)

Academic contact during examination: Jingyue Li

Phone: 9189 7446

Examination date: 13-May-2019

Examination time (from-to): 9.00

Permitted examination support material: D

Other information:

Language: English

Number of pages (front page excluded): 7

Number of pages enclosed: 1

Introduction

In this course, the written exam will count 70% of the final grade, and the remaining 30% of the final grade comes from the compulsory exercises.
So, your final grade of this course will be:
(Points you get from this written exam) * 70% + your grade of compulsory exercises.

If you feel that any of the problems require information that you do not find in the text, then you should

- Document the necessary assumptions
- Explain why you need them

Your answers should be brief and to the point.

Problem 1 – (40 points)

1) (2 points) Explain why prepared statement and bind variables can defend against SQL injection attacks?

2) (3 points) Explain the three possible ways to store session tokens and compare their advantages and disadvantages.

3) (4 points) Explain how to identify if a web site is vulnerable to CSRF attacks.

4) (4 points) Explain what a clickjacking attack is and how to defend against the clickjacking attack.

5) (4 points) Explain why using the ECB model in block cipher is insecure and how to deal with it.

6) (4 points) Explain digital signature, Certification Authority (CA), and how they are related to the SSL/TLS handshake process.

7) (4 points) Explain the Biba model and why it can help improve integrity.

8) (3 points) Explain the components and process of Single Sign-On (SSO).

9) (4 points) Explain how files are encrypted in iOS.

10) (4 points) Explain the proof-of-work consensus model of the blockchain, what security attack such a model can defend against, and the disadvantage of the model.

11) (4 points) Explain what web application firewall of Azure is and why it can only partially defend against the session fixation attack.

Problem 2 – (30 points in total)

For each of the code snippets listed below, your task is to:

- Identify all security vulnerability and their locations in the code (Note: you may find more than one vulnerabilities in one code snippet. You need to list and identify all of them.)
- Explain why these are security vulnerabilities/issues and how to exploit the vulnerabilities to attack the app
- Fix the code (You may use pseudo-code for this. Remember to explain your solution).

To present your answsers in a structured manner, you may format your answers like:
Code Snippet 1: vulnerability 1: ... (in Line ...); It is vulnerable because ... and the method to exploit the vulnerability is ...; The fix could be ....
...
Code snippet 2: ...

Code snippet 1

Source: guyrutenberg.com

```
1. PREFIX = '/home/user/files/'
2. full_path = os.path.join(PREFIX, filepath)
3. read(full_path, 'rb')
```

Code snippet 2

Source: lets-be-bad-guys project

```
1. users = {
```

```
2.
3.      '1': {
4.          'name': 'Foo',
5.          'email': 'foo@example.com',
6.          'admin': False,
7.      },
8.
9.      '2': {
10.         'name': 'Bar',
11.         'email': 'bar@example.com',
12.         'admin': True,
13.     }
14. }
15.
16. def user_profile(request, userid=None):
17.
18.     env = {}
19.
20.     user_data = users.get(userid)
21.
22.     if request.method == 'POST':
23.
24.         user_data['name'] = request.POST.get('name') or user_data['name']
25.
26.         user_data['email'] = request.POST.get('email') or user_data['email']
27.
28.         env['updated'] = True
29.
30.     env['user_data'] = user_data
31.
32.     env['user_id'] = userid
33.
34.     return render (request, '/profile.html', env)
35.
```

/profile.html

```
1.  {% block content %}
2.
3.  {% if updated %}
4.
5.  <p>Updated your user profile, thanks!</p>
6.
7.  {% endif %}
8.
9.  <form action="{% url ' code-profile' user_id %}" method="POST">
10.
11. {% if user_data.admin %}
12.
13. <p>You are an admin! Use it wisely.</p>
14.
```

```
{% endif %}

<p><label for="name">Name:</label>

    <input type="text" name="name" id="name" value="{{ user_data.name }}"></p>

<p><label for="email">Email:</label>

    <input type="email" name="email" id="email" value="{{ user_data.email }}"></p>

<p><label for="password">Password:</label>

    <input type="password" name="password" id="password"></p>

<p><input type="submit"></p>

</form>

{% endblock %}
```

/urls.py

```
url(r'^folder/users/(?P<userid>\d+)$',
exercises.user_profile, name='code-profile'),
```

Note: "**folder" is the folder which profile.html locates**

Code snippet 3

Source: wiki.sei.cmu.edu

```
import java.io.*;

class DeserializeObj {

  public static Object deserialize(byte[] buffer) throws IOException,
ClassNotFoundException {

    Object ret = null;

    try (ByteArrayInputStream bais = new ByteArrayInputStream(buffer)) {

      try (ObjectInputStream ois = new ObjectInputStream(bais)) {

        ret = ois.readObject();

      }

    }

    return ret;
```

```
20.
21.   }
22.
23. }
```

Code snippet 4

Source: CWE-643

XML doc

```
1. <users>
2.    <user>
3.       <login>john</login>
         <password>abracadabra</password>
         <home_dir>/home/john</home_dir>
4.    </user>
      <user>
5.       <login>cbc</login>
         <password>1mgr8</password>
         <home_dir>/home/cbc</home_dir>
6.       </user>
7. </users>
```

Java code used to retrieve the home directory based on the provided credentials

```
1. XPath xpath = XPathFactory.newInstance().newXPath();

2. XPathExpression xlogin = xpath.compile("//users/user[login/text()='" +
   login.getUserName() + "' and password/text() = '" + login.getPassword() +
   "']/home_dir/text()");


3. Document d =
   DocumentBuilderFactory.newInstance().newDocumentBuilder().parse(new
   File("db.xml"));
4. String homedir = xlogin.evaluate(d);
```

Code snippet 5

Source CWE-601

```
1. public class RedirectServlet extends HttpServlet {

2.    protected void doGet(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException {

3.       String query = request.getQueryString();

4.       if (query.contains("url")) {

5.          String url = request.getParameter("url");
```

```
6.              response.sendRedirect(url);
7.          }
8.      }
9.  }
```

Problem 3 – (30 points in total)

Case description:

Company COOL wants to make a SocialUber app to facilitate sharing of the private autonomous car as a taxi and to mitigate possible safety risks. The app will have both web and mobile versions.

Three kinds of stakeholders will be the users of the app, namely the autonomous car owners, the autonomous car users, and friends/relatives of the autonomous car users.

- The autonomous car owners want to earn money by renting out their autonomous car as a taxi when they do not use the car. They will use the app to register their information and the information and availability of the car. They also want to use the app to receive payment from the car user.
- The autonomous car users want to order and use an autonomous car as a taxi. They will use the app to order the autonomous car, to see the location of the ordered car, and to make payment after using it.
- When start using the autonomous car, a car user can use the app to inform his/her friends/relatives about the location of the car in the whole journey, so that his/her friends can trace where the car is and can help inform police or ambulance, if an incident happens.

To use the app, the autonomous car owner needs to register and fill in the following information:

- Username and password (mandatory)
- Real name (mandatory)
- Email address (mandatory)
- Phone number (mandatory)

- Home address (mandatory)
- Information of the car (e.g., registration ID, size of the car, and price to be used as a taxi) (mandatory)
- A bank account to receive the payment (mandatory)

The car owner needs to log in the system to change the information and to update the availability of the car.

The autonomous car user needs to register and fill in the following information:

- Username and password (mandatory)
- Real name (mandatory)
- Email address to receive a receipt (optional)
- Phone number (mandatory)
- Credit card information, if the user wants to use one-click payment (optional)
- The username of the friend/relative he/she wants to link to (optional). When filling such information, an SMS will be sent to the phone of the friend/relative and for approving the link. The information will be saved to the app, only if the friend/relative approves the link.

The autonomous car user needs to log in the system to order the car.

A friend/relative of the car user needs to register and fill in the following information:

- Username and password (mandatory)
- Phone number (mandatory)

The friend/relative of the car user needs to log in the system to see the location of the car in use.

All filled-in information by car owners, car users, and their friends will be encrypted and saved at the server side of the app.

A simplified usage scenario is as follows:

- A car user opens the app and types in the starting and ending address of a journey.

- The app searches for available autonomous cars nearby and returns a list of their possible arrival time and prices.
- The user chooses an autonomous car from the list.
- The app sends SMS to the car owner and asks the car owner to approve the request of using the autonomous car.
- The autonomous car owner approves the request.
- The app sends a confirmation code to the car user.
- The autonomous car leaves home and arrives at the starting address of the journey.
- The car user opens the door of the autonomous car using the confirmation code he/she receives from the app.
- The car user sits in the car.
- The care user can open the app to choose from the list in the app one friend/relative to monitor the journey.
- In the whole journey, the app will keep updating the location of the car to the friend/relative of the car user. The friend/relative can see the car location after logging in.
- The car arrives at the destination.
- The car user leaves the car, closes the door, and clicks the button "journey completed" in the app. The friend/relative of the car user will get informed in the app that the journey is complete if he or she is observing the journey.
- The app will calculate and show the price to the user. The user can then click the "one-click payment" button to pay the journey if the credit card information is saved in advance. The user can also pay the journey through filling in credit card information. If the user does not make the payment, he or she could not use the app to make a new order. After the payment, 0.1% of the payment will be transferred to Company COOL as the cost of using the SocialUber app.
- After the payment, the app notices car owner about the payment.
- The app will send a receipt to the user as email if his/her email address to receive a receipt is saved.

Your task is to make a risk-based assessment of this application based on the RMF (Risk Management Framework).

Your tasks include:

- Task 1: Identify business goals, business assets, and business risks (5 points).

- Task 2: Identify at least 10 technical risks related to the SocialUber app using threat modeling and explain how threat modeling is performed (12 points). (Note: You do not need to draw the threat modeling graphs. However, you need to explain briefly how the threat modeling, e.g., misuse cases and attack trees, are applied to help you identify the technical risks.)

- Task 3: Derive security requirements from each technical risk identified, and design and describe black-box penetration test cases (including test steps and expected results of each step) to verify each derived security requirement (10 points)

- Task 4: This SocialUber app must be compliant with the General Data Protection Regulation (GDPR). Discuss how to address the privacy issue of this app (3 points)

To present your answsers in a structured manner, you may format your answers like:
Task 1: ...
Task 2: ...
Task 3: ...
Task 4: ...