A22652 *No Calculator permitted in this examination*

UNIVERSITY OF BIRMINGHAM

School of Computer Science

First Year – MSc Computer Science
First Year – UG Aff Computer Science/Software Engineering
Fifth Year – MEng Electronic and Software Engineering with Industrial Year
First Year – MSc Computer Security

06 20010

Secure Programming

Summer Examinations 2013

Time Allowed: 1:30 hours

[Answer ALL Questions]

No calculator

Note: question 11 is worth 25% of the marks.

1. Give a definition of "vulnerability" and "exploit". **[3%]**

2. Suppose that you are using the search functionality on the School's web site. When you search for `O'Reilly` you notice that the results contain a MySQL error: there likely is a SQL injection vulnerability. You consider running more manual tests or even an automated vulnerability scanner to validate your hypothesis. Would this be appropriate? If not, what would be an acceptable course of action? **[2%]**

3. Statistics show that vulnerabilities are prevalent in today's software. Discuss two reasons for the wide-spread presence of vulnerabilities. **[4%]**

4. (a) Explain and discuss the security principle of "Complete mediation." **[4%]**
   (b) Present an example of a system of your choice that applies this principle, and one in which this principle is violated. **[6%]**

5. Suppose the School is concerned that students may try to pass exams without actually knowing the material, and it hires you to defend against this threat. You propose to analyze the security of the existing system by using the attack tree methodology: students are seen as potential attackers whose goal is to pass the exam without knowing the material. Present the attack tree you develop for this scenario. **[10%]**

6. Suppose a customer asks you to identify vulnerabilities in one of his web applications. He provides you with the URL of the application's main page, but he does not give you access to its source code. Describe the steps you take to perform this analysis. **[10%]**

7. (a) Describe SQL injection vulnerabilities **[2%]**

   (b) You are looking through the logs of your SQL server when you notice the following sequence of queries:

```
SELECT id FROM users WHERE username = "foo" and password = "secret"
SELECT id FROM users WHERE username = "foo" and password = "bar" union
select if(substring(user(), 1, 1) = 'r', SLEEP(10), 0)
SELECT id FROM users WHERE username = "foo" and password = "bar" union
select if(substring(user(), 1, 1) = 'o', SLEEP(10), 0).
SELECT id FROM users WHERE username = "foo" and password = "bar" union
select if(substring(user(), 1, 1) = 'o', SLEEP(10), 0).
SELECT id FROM users WHERE username = "foo" and password = "bar" union
select if(substring(user(), 1, 1) = 't', SLEEP(10), 0).
```

   What is likely happening? If the log contains an attack, name the attack and discuss the specific technique being used.

   [3%]

8. (a) In the context of shellcode, describe what a *GetPC* mechanism is, and discuss its use. **[2%]**

   (b) Provide one concrete example of a GetPC mechanism. **[3%]**

   (c) What is the reason for using NOP sleds in shellcode? **[3%]**

9. (a) In the context of memory corruption vulnerabilities, what is a *canary*? **[2%]**

   (b) Draw the stack layout for a function frame that is protected by a canary (in particular, define the position of the canary with respect to local variables and control data, such as the saved return address). **[4%]**

   (c) Describe two types of canaries. **[3%]**

10. (a) Consider the following program:

```
1       void foo(char *s) {
2           char *buf1, *buf2;
3           buf1 = malloc(1024);
4           buf2 = malloc(1024);
5
6           strcpy(buf1, s);
7
8           free(buf1);
9           free(buf2)
10
11          return 0;
12      }
```

   Assume that the string s passed as a parameter to foo can be as large as 4,096 characters. What vulnerability does the code contain? **[3%]**

   (b) Propose a fix for the vulnerability. **[3%]**

(c) Assume you want to exploit this vulnerability: draw a layout of the heap before the first `free` is invoked. What are the key control structures that you need to overwrite to successfully perform this exploit? **[8%]**

11. BuggyFriend is an up-and-coming startup company rivaling with Facebook. Before the official launch of their new web application, they have hired you to perform a full source code audit of its code. Identify the vulnerabilities contained in the BuggyFriend's code (use the line numbers to specify the location of each vulnerability) *and* briefly propose a fix for each of them. **[25%]**

(*The code is listed on the following page.*)

```
1   public class BuggyFriend extends HttpServlet {
2     Connection conn = ... /* initializes the DB connection */;
3     String sanitizeSql(String str) {
4         String cleanStr = str.replace("SELECT", "_SELECT_");
5         cleanStr = cleanStr.replace("OR", "_OR_");
6         return cleanStr;
7     }
8     protected void processRequest(HttpServletRequest request,
9       HttpServletResponse response)
10    throws ServletException, IOException, SQLException {
11        PrintWriter out = response.getWriter();
12
13        String action = request.getParameter("action");
14        // search for a friend whose name contains the given string
15        if ("search".equals(action)) {
16            Statement st = conn.createStatement();
17            ResultSet rs = st.executeQuery("SELECT * "
18              + "FROM friends WHERE name LIKE '%"
19              + sanitizeSql(request.getParameter("q"))
20              + "%'");
21            while (rs.next()) {
22                String f = rs.getString("NAME");
23                out.write(f);
24            }
25            st.close();
26        // view the privacy policy in the language specified by the
27        // user (lang parameter)
28        } else if ("view_privacy_policy".equals(action)) {
29            String lang = request.getParameter("lang");
30            if (lang.matches("[a-z]{2}")) {
31              BufferedReader f = new BufferedReader(
32                new FileReader("policies/privacy/" + lang));
33              String s;
34              while ((s = f.readLine()) != null) {
35                  out.write(s + "\n");
36              }
37            }
38        // send a friendship invitation via email to the address
39        // specified by the user (to parameter)
40        } else if ("email".equals(action)) {
41            String to = request.getParameter("to");
42            String[] cmd = {"/bin/bash", "-c",
43                "cat invitation.txt | "
44                + "mail -s \"Friendship invitation\" " + to};
45            Runtime.getRuntime().exec(cmd);
46        // invalid action
47        } else {
48            out.write("Error: invalid action: " + action);
49        }
50    }
51  }
```