A22652

Calculator permitted provided they are not capable of being used to store alphabetical information other than hexadecimal numbers.

UNIVERSITY OF BIRMINGHAM

School of Computer Science

MSc Advanced Computer Science
Final Year - MSci Computer Science
MSc Cyber Security
Final Year – MEng Computer Science/Software Engineering
Final Year – MSci Computer Science with Industrial Year
Final Year – MEng Computer Science/Software Engineering with Industrial Year

06 20010

Secure Programming

Summer May/June Examinations 2017

Time allowed: 1 hour 30 minutes

[Answer ALL Questions]

TURN OVER

1. **[Total: 30%]** Your task here is to investigate an application that is vulnerable to SQL-Injection. The main purpose of the applications are to receive log messages from a TCP connection and store them in an SQL database. Since the messages might contain arbitrary content, white listing approaches are not permitted here. To send messages to the application, you might use a command like: *echo message |nc localhost 7777* or alternatively the command *nc localhost 7777* to use the application interactively.

   You can find the code for this app, in Java, Python and PHP in the three figures below. Pick any language and answer the questions below. State clearly which language your answers relate to. You only need to answer the questions for one language, if you provide answers for more than one language you will be awarded the maximum marked achieved on any any of them.

   (a) Name all lines in which (possibly untrusted) input is read from external sources. **[10%]**

   (b) Name all lines in which SQL statements are sent to the database. **[8%]**

   (c) Name all lines in which SQL statements are built in an insecure way so that their structure might be altered by a malicious user. **[12%]**

Listing 1: sql-injection/java/App.java

```
1  public class App {
2
3      public static Connection connect() throws SQLException {
4          Connection con=DriverManager.getConnection( "jdbc:mysql://db
               :3306/bham","bham","bham");
5          return con;
6      }
7
8      public static void checkTables(Connection con) throws Exception {
9          Statement stmt=con.createStatement();
10         stmt.execute("create table IF NOT EXISTS logging (msg text not
               NULL, ts TIMESTAMP DEFAULT CURRENT_TIMESTAMP)");
11         stmt.close();
12     }
13
14     public static void log(Connection con, String msg) throws
           SQLException {
15         Statement stmt=con.createStatement();
16         stmt.execute("INSERT into logging (msg) values ('"+msg+"')");
17         stmt.close();
18     }
19
20     public static void showLastMessages(Connection con, PrintWriter pw
           ) throws SQLException {
21         Statement stmt=con.createStatement();
22         ResultSet rs=stmt.executeQuery("select msg from (select * from
               logging order by ts desc limit 10) as log order by ts");
23         while (rs.next()) { pw.println("> " + rs.getString("msg")); }
24         rs.close();
25         stmt.close();
26     }
27
28     public static void main(String[] args) throws Exception {
29         ServerSocket serverSocket = new ServerSocket(7777);
30         while(true) {
31             Socket connection = serverSocket.accept();
32             BufferedReader br = new BufferedReader(new
                   InputStreamReader(connection.getInputStream()));
33             PrintWriter pw = new PrintWriter(connection.
                   getOutputStream(), true);
34             pw.println("Welcome to the logging service");
35             Connection con = connect();
36             checkTables(con); showLastMessages(con, pw);
37             String l = br.readLine();
38             while (l != null) { log(con, l); l = br.readLine(); }
39             connection.close();
40         }
41     }
42 }
```

Listing 2: sql-injection/python/app/server.py

```
#/usr/bin/env python3

import socketserver
import MySQLdb

class MyServer(socketserver.StreamRequestHandler):

        def checkTables(self, db):
                c = db.cursor()
                c.execute("create table IF NOT EXISTS logging (msg
                    text not NULL, ts TIMESTAMP DEFAULT
                    CURRENT_TIMESTAMP)")
                db.commit()
                c.close()

        def showLastMessages(self, db, f):
                c = db.cursor()
                c.execute("select msg from (select * from logging
                    order by ts desc limit 10) as log order by ts")
                r = c.fetchall()
                for row in r:
                        f.write((">" + row[0] + "\n").encode())
                c.close()

        def log(self, db, msg):
                c = db.cursor()
                c.execute("INSERT into logging (msg) values ('" + msg
                    + "')")
                db.commit()
                c.close()

        def handle(self):
                self.wfile.write(("Welcome to the logging service\n").
                    encode())
                db = MySQLdb.connect("db","bham","bham","bham" )
                self.wfile.write(("connected!\n").encode())
                self.checkTables(db)
                self.wfile.write(("tables checked!\n").encode())
                self.showLastMessages(db, self.wfile)
                while True:

                        data = self.rfile.readline().strip()
                        if not data: break
                        self.log(db, data.decode())

if __name__ == "__main__":
        host, port = "", 7777
        socketserver.TCPServer.allow_reuse_address = True
        server = socketserver.TCPServer((host, port), MyServer)
        server.serve_forever()
```

Listing 3: sql-injection/php/app/server.php

```
<?php

function checkTables($link) {
  $create = mysqli_stmt_init($link);
  mysqli_stmt_prepare($create, "create table IF NOT EXISTS logging (
      msg text not NULL, ts TIMESTAMP DEFAULT CURRENT_TIMESTAMP)");
  mysqli_stmt_execute($create);
  mysqli_stmt_close($create);
}

function showLastMessages($link, $msgsock) {
  $messages = mysqli_stmt_init($link);
  mysqli_stmt_prepare($messages, "select msg from (select * from
      logging order by ts desc limit 10) as log order by ts");
  mysqli_stmt_execute($messages);
  mysqli_stmt_bind_result($messages, $msg);
  while (mysqli_stmt_fetch($messages)) {
    $s = "> " . $msg . "\n"; socket_write($msgsock, $s, strlen($s));
  }
  mysqli_stmt_close($messages);
}

function logmsg($link, $msg) {
  $savemsg = mysqli_stmt_init($link);
  $q = "INSERT into logging (msg) values ('" . $msg . "')";
  if(!mysqli_stmt_prepare($savemsg, $q)) {
    echo "mysqli_stmt_prepare failed: " . mysqli_error($link) . "\n";
    echo $q . "\n"; return;
  }
  mysqli_stmt_execute($savemsg);
}

error_reporting(E_ERROR); set_time_limit(0); ob_implicit_flush();

$address = '0.0.0.0'; $port = 7777;


if ((($sock=socket_create(AF_INET, SOCK_STREAM, SOL_TCP)) === false) {
    echo "socket_create() failed: reason: " . socket_strerror(
        socket_last_error()) . "\n"; }

if (socket_bind($sock, $address, $port) === false) {
    echo "socket_bind() failed: reason: " . socket_strerror(
        socket_last_error($sock)) . "\n"; }

if (socket_listen($sock, 5) === false) {
    echo "socket_listen() failed: reason: " . socket_strerror(
        socket_last_error($sock)) . "\n"; }


// Continued over page.
```

```
48 do {
49   if (($msgsock = socket_accept($sock)) === false) {
50     echo "socket_accept() failed: reason: " . socket_strerror(
         socket_last_error($sock)) . "\n";
51     break;
52   }
53   $msgwelcome = "Welcome to the logging service\n";
54   socket_write($msgsock, $msgwelcome, strlen($msgwelcome));
55   $link = mysqli_connect("db","bham","bham","bham");
56   if (!$link) { echo "mysqli_connect() failed\n"; break; }
57   mysqli_autocommit($link, TRUE);
58   checkTables($link);
59   showLastMessages($link, $msgsock);
60   do {
61     $buf = "";
62     while(true) {
63       if (false === ($buf = socket_read($msgsock, 2048,
           PHP_NORMAL_READ))) { break 2; }
64       $c = substr($buf, -1);
65       if (($c == "\r") || ($c == "\n")) { break; }
66     }
67     $buf = str_replace("\n", '', str_replace("\r", '', $buf));
68     logmsg($link, $buf);
69   } while (true);
70   socket_close($msgsock);
71 } while (true);
72
73 socket_close($sock);
74
75 ?>
```

2. **[Total: 30%]** Your task is to fix an application that is written to show all lines in /etc/passwd that contain a specific string. The applications read the string from stdin and then invoke *grep* to accomplish this. All applications contain a Shell-Injection vulnerability, which must be fixed. Since the search pattern might be arbitrary, white listing approaches are not allowed here.

   You can find the code for this app, in Java, Python and PHP in the three figures below. Pick any language and answer the questions below. State clearly which language your answers relate to. You only need to answer the questions for one language, if you provide answers for more than one language you will be awarded the maximum marked achieved on any any of them.

   (a) Name all variables that contain input that is (partially) under external control. For local variables, also name the function they are declared in. **[12%]**

   (b) Which lines call external commands? Is this way of calling vulnerable to Shell-Injection? **[12%]**

   (c) Craft an input for the application that would it make delete the file */tmp/bham* instead of searching for a specific user in */etc/passwd*. **[6%]**

Listing 4: shell-escape/java/App.java

```
1
2  public class App
3  {
4      public static void main( String[] args ) throws IOException
5      {
6          Runtime runtime = Runtime.getRuntime();
7          BufferedReader br = new BufferedReader(new InputStreamReader(
               System.in));
8          String user = br.readLine().replaceAll("\\r|\\n", "");
9          Process p = runtime.exec("grep " + user + " /etc/passwd");
10         InputStream is = p.getInputStream();
11         byte[] buf = new byte[4096];
12         int l = is.read(buf);
13         while (l > 0) {
14                 System.out.write(buf, 0, l);
15                 l = is.read(buf);
16         }
17     }
18 }
```

Listing 5: shell-escape/python/user.py

```
1  #!/usr/bin/env python3
2
3  import sys
4  import os
5
6  user = sys.stdin.readline()
7  os.system("grep " + user.rstrip("\n\r") + " /etc/passwd")
```

Listing 6: shell-escape/php/app/user.php

```php
<?php

    $fp=fopen("php://stdin","r");
    $line=stream_get_line($fp,65535, "\n");
    fclose($fp);
    system("grep␣$line␣/etc/passwd");

?>
```

3. **[Total: 10%]** Consider the following two functions *clock1* and *clock2* in C:

Listing 7: verification/functions.c

```c
#include <stdio.h>
#include <stdint.h>
#include <assert.h>

uint64_t clock1(uint64_t in) {
        return ((((in >> 0)^(in >> 1)^(in >> 3)^(in >> 4))&1)<<63)|(in
            >> 1);
}

uint64_t clock2(uint64_t in) {
        uint64_t t64, t63, t61, t60;
        t64 = in&1;
        t63 = (in >> 1)&1;
        t61 = (in >> 3)&1;
        t60 = (in >> 4)&1;
        in = in >> 1;
        return (((t64+t63+t61+t60)&1)<< 63) | in;
}
```

Here *clock1* is supposed to be an optimized version of the function that is implemented in *clock2*. Of course, both implementations are supposed to return the same output for the same input.

(a) How in general can you check that both implementations are equivalent? Name a tool or a method that would support you with that. **[5%]**

(b) How would you run the tool? What would be a suitable input for the tool and how would you have to rewrite the source code file to make it suitable an input for the tool? **[5%]**

4. **[Total: 15%]** Assume you would like to write a web application. You are concerned about Cross-Side- Scripting attacks.

(a) Explain what a Cross-Side-Scripting attack is. Expected as answer are 1-2 paragraphs. **[5%]**

(b) Name and briefly outline two different ways how you can counter Cross-Side- Scripting attacks in your application. Expected as answer is about one paragraph per countermeasure. **[10%]**

5. **[Total: 15%]** Explain the difference between *setuid* and *seteuid*? For each of them, outline a scenario in which this would be useful but the other one would not be. Expected as answer are 1-2 paragraphs for the general difference and 1-2 paragraphs for each scenario.