CIS 4930: Secure Coding [Fall 2018]
Test III

NAME: _____

Instructions:

1) This test is 7 pages in length.

2) You have 40 minutes to complete and turn in this test.

3) Short answer questions include a guideline for how many sentences to write. Respond in complete English sentences. Responses will be graded as described on the syllabus.

4) This test is closed books, notes, papers, friends, neighbors, etc.

5) Use the backs of pages in this test packet for scratch work. If you write more than a final answer in the area next to a question, circle your final answer.

1. [6 points]
What are three techniques we discussed in class for mitigating insider threats/attacks? [1 sentence]

2. [10 points]
Compare and contrast access control lists with capability lists. [2-3 sentences]

3. [20 points]
Describe five of the secure-software-design principles discussed in class. [5 sentences]

4. [30 points]
Consider the following code from the SimpleWebServer example.

```
1    package com.learnsecurity;
2    import java.io.*;
3    import java.net.*;
4    import java.util.*;
5
6    public class SimpleWebServer {
7
8        /* Run the HTTP server on this TCP port. */
9        private static final int PORT = 8080;
10
11       /* The socket used to process incoming connections from web clients */
12       private static ServerSocket dServerSocket;
13
14       public SimpleWebServer () throws Exception {
15           dServerSocket = new ServerSocket (PORT);
16       }
17
18       public void run() throws Exception {
19           while (true) {
20               /* wait for a connection from a client */
21               Socket s = dServerSocket.accept();
22
23               /* then process the client's request */
24               processRequest(s);
25           }
26       }
27
28       /* Reads the HTTP request from the client, and
29          responds with the file the user requested or
30          a HTTP error code. */
31       public void processRequest(Socket s) throws Exception {
32           /* used to read data from the client */
33           BufferedReader br = new BufferedReader
34                       (new InputStreamReader (s.getInputStream()));
35
36           /* used to write data to the client */
37           OutputStreamWriter osw = new OutputStreamWriter (s.getOutputStream());
38
39           /* read the HTTP request from the client */
40           String request = br.readLine();
41
42           String command = null;
43           String pathname = null;
44
45           /* parse the HTTP request */
46           StringTokenizer st = new StringTokenizer (request, " ");
47
48           command = st.nextToken();
49           pathname = st.nextToken();
50
51           if (command.equals("GET")) {
52               /* if the request is a GET
53                  try to respond with the file
54                  the user is requesting */
55               serveFile (osw,pathname);
56           } else {
```

```
57              /* if the request is a NOT a GET,
58                 return an error saying this server
59                 does not implement the requested command */
60              osw.write ("HTTP/1.0 501 Not Implemented\n\n");
61          }
62
63          /* close the connection to the client */
64          osw.close();
65      }
66
67      public void serveFile (OutputStreamWriter osw,
68                     String pathname) throws Exception {
69          FileReader fr=null;
70          int c=-1;
71          StringBuffer sb = new StringBuffer();
72
73          /* remove the initial slash at the beginning
74             of the pathname in the request */
75          if (pathname.charAt(0)=='/')
76              pathname=pathname.substring(1);
77
78          /* if there was no filename specified by the
79             client, serve the "index.html" file */
80          if (pathname.equals(""))
81              pathname="index.html";
82
83          /* try to open file specified by pathname */
84          try {
85              fr = new FileReader (pathname);
86              c = fr.read();
87          } catch (Exception e) {
88              /* if the file is not found,return the
89                 appropriate HTTP response code  */
90              osw.write ("HTTP/1.0 404 Not Found\n\n");
91              return;
92          }
93
94          /* if the requested file can be successfully opened
95             and read, then return an OK response code and
96             send the contents of the file */
97          osw.write ("HTTP/1.0 200 OK\n\n");
98          while (c != -1) {
99              sb.append((char)c);
100             c = fr.read();
101         }
102         osw.write (sb.toString());
103      }
104
105      /* This method is called when the program is run from
106         the command line. */
107     public static void main (String argv[]) throws Exception {
108
109         /* Create a SimpleWebServer object, and run it */
110         SimpleWebServer sws = new SimpleWebServer();
111         sws.run();
112     }
113 }
```

Identify three vulnerabilities in this SimpleWebServer code (including line numbers when possible). For each vulnerability, explain (a) the damage an attacker could do, by exploiting that vulnerability, and (b) how the code could be modified, to remove the vulnerability (please explain in English; you need not write any code).

5. [34 points]
Consider the following code. (Assume all needed *#include* directives are also present.)

```
1   char *c;
2
3   void h(char *a) {
4      /* Allocates two characters and returns a pointer to the first, i.e., c[0]. */
5       c = malloc(2);
6       c[0] = 'X';
7   }
8
9   void g(char *a) {
10      h(a);
11  }
12
13  int f() {
14     char s[512];
15     gets(s);
16     return 4;
17  }
18
19  void i(char *a) {
20     f();
21  }
22
23  int main (int argc, char *argv[]) {
24     char *a;
25     g(a);
26     i(a);
27
28     return 0;
29  }
```

(a) Draw a representation of the program memory segments (including their contents when known) right before Line 25 is executed, at the level of detail shown in class.

(b) Draw a representation of the program memory segments (including their contents when known) right before Line 15 is executed, again at the level of detail shown in class.

(c) Explain how a user can attack this program. [2-3 sentences]