# Examination Systems Security

| | |
|---|---|
| Module/course code: | Systems Security |
| Date: | 25 June 2018 |
| Time: | 9:00-12:00 |
| Instructors: | Erik Tews and Stjepan Picek |

Type of test:

- Open book

Allowed material and aids during the test:

- Pretty much everything on paper
- That includes your own notes
- Also slides from the lecture
- And everything else you could find and print somewhere
- Or any kind of book you like

Forbidden material and aids during the test:

- Everything electronically such as:
  - E-Book readers
  - Smartphones
  - Laptop computers
  - Tablets
  - Smartwatches

Additional remarks:

- Read these instructions and the questions carefully! If the questions are unclear, you can ask for clarification.
- Make sure you answered all parts of a question and not just some of them.
- Please make sure that your name and student number appear on all answer sheet
- Try to give precise answers using appropriate terminology and always give a reason for your answer.
- Unreadable or extremely long answers will not be marked.
- Give your answers in English.
- The exam consists of 7 pages for questions and an 8 page long appendix with source code.

| Nr | Question | Points |
|---|---|---|
| 1 | ESP8266 | 6 |

Assume you are building an IoT device based on the ESP8266 chip, using for example the Wemos D1 development board (the same one that was shown in the lecture). A temperature sensor as well as a relay is connected to the device. The temperature can be read from a GPIO port and the relay is connected to a different GPIO port. The main purpose of the device is to monitor the temperature and switch the relay to "on" when the temperature falls below 21°C and turn it off gain once the temperature has reached 21°C again. In addition, it should provide a webserver that allows a user to check the current temperature.

You want to make sure that the relay cannot be controlled from the webserver, so that in case there should be a vulnerability such as a buffer overflow in the webserver code, an adversary will not be able to switch the relay. Can you use Seccomp to sandbox the webserver and prevent it from accessing the relay?

| Nr | Question | Points |
|---|---|---|
| 2 | A new IoT device | 7 |

Assume that you recently bought a new IoT device, a smart plug, a similar but not the same one as the one shown in the lecture. There is a mobile app and both (the device and the mobile app) connect to a cloud service. From observing the network traffic, you see that there is basically just an on and an off command that is sent from the cloud to the device when you press on or off in the app.

You would like to know whether there is more functionality implemented in the device than just on an off, but you don't see any more commands in the network traffic. Suggest an approach that will help you to discover hidden functionality in the device such as for example a command that would instruct the device to send your WiFi password to the cloud.

| Nr | Question | Points |
|---|---|---|
| 3 | Covert channels on rooted Android phones | 6 |

In the assignment, you had to add a covert channel in an Android app. Now assume that the same app would be running on a rooted phone, on which you would have full privileges, which means you are able to do everything that you can do on a normal Linux system as root. Sketch a design for a covert channel that you could implement in this environment, which was not possible for the assignment due to not having sufficient privileges.

| Nr | Question | Points |
|---|---|---|
| 4 | Creating sound on a PC | 6 |

Assume that you would like to exfiltrate data from a normal desktop computer via sound. However there are not speaker connected. Suggest an approach how you can create a sound pattern (from a normal program running without root/Administrator privileges) on this machine, that you can use to exfiltrate a few bits of data such as a pin number.

| Nr | Question | Points |
|---|---|---|
| 5 | A new web application | 7 |

Many students like to speculate about the final grade and try to figure out what final grade they would get in a course when they get X points in assignment A and Y points in assignment B and Z points for the final exam. Of course that is hard work since they need to apply the grading rules to all their speculative points and then determine the final result by hand. Assume that a fellow student will implement a web application that when supplied with the grading rules allow students to enter speculative points and then determine their final grade or the expected range of their final grade. During the course, you all used Canvas at https://canvas.utwente.nl/ and we would like to add a "Speculate with your grades" button to each course which will take you to this web application.

However, we are a bit concerned about the secure coding skills of that particular student and we assume that there might be several cross side scripting or SQL-injection vulnerabilities in this web application. Suggest a good architecture of this web application so that a cross side scripting or SQL injection vulnerability in this web application should have no effect on canvas itself, so that for example a cross side scripting vulnerability in this application will not enable an attacker to read the assignment submissions of a particular student from canvas or see his real grades.

| Nr | Question | Points |
|---|---|---|
| 6 | Rancomcat redesign | 12 |

In one of the assignments, you modified "randomcatserver.c" so that it would compile and load a Seccomp filter that prevents the execution of any none-required syscalls before untrusted data from a client is processed. Assume that rancomcatserver.c would be modified so that it does not fork anymore after it has accepted a new client, and handles the request directly in the main thread. You find such a modified implementation in "randomcatserver-nofork.c". Lines 201-212 have been altered, all remaining lines are still the same.

a) What would be the implications for adding Seccomp rules? Can you still use the same filter you implemented in your assignment or do you have to allow more syscalls?

b) What are the security implications? Assume that there is a buffer overflow in process (line 112 and following) so that an adversary is able to execute his own code within the server process. Assuming you added a Seccomp rules that prevent the process from executing any syscall that is not required. What is an adversary able to do that was not possible with the previous version that used fork?

| Nr | Question | Points |
|---|---|---|
| 7 | A webserver for static files | 6 |

You would like to write a webserver that serves static files from a specific directory. The files might change during runtime and are also too large to read them all into memory during startup. You are afraid that your server is exploitable and you would like to make sure that your server is only able to access files from that specific directory, but not from the entire filesystem. Can you use Seccomp to enforce that? When yes, describe how, when not, then suggest another feature of Linux that allows you to restrict your server to a specific subdirectory (after initialization).

| Nr | Question | Points |
|---|---|---|
| 8 | RNG | 8 |

You are security expert in a company. You want to use RNG to produce random numbers. Your idea is to use TERO. Explain to your (hypothetical) boss how TERO works. To make it easier, we depict it in Figure 1.

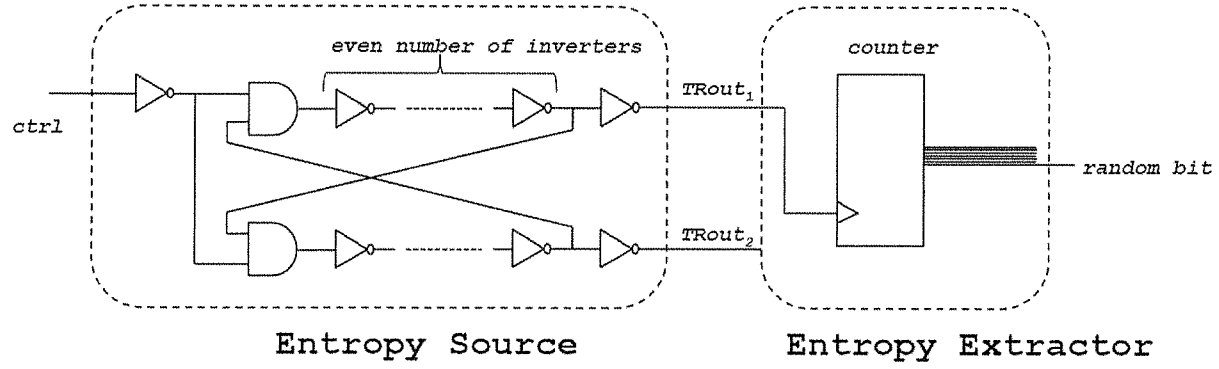


Figure 1.

Unfortunately, after your explanation your boss is still not happy. He thinks that is too much money and proposes to use a simple rand() and modulo operations to produce random numbers. The rand() function results in numbers between 0 and RAND_MAX (including RAND_MAX).
Then, the result of a rand() function is calculated modulo n in order to obtain values between 0 and n-1. Let us assume that RAND_MAX = 10 and n = 6. Does the output have good properties to be used as PRNG? Why? If not, what would you change with the value (RAND_MAX and/or n) to make it better?

| Nr | Question | Points |
|---|---|---|
| 9 | PUF | 10 |

You decided to use arbiter PUF in your RFID tags. Still, before running the actual production you want to conduct some analysis. When conducting the numerical analysis/simulation what is the role of feature vector when simulating a PUF? Do we need it?
In our simulation, arbiter PUF consists of two 2-bit multiplexers. Usually, we used random delay to model behaviour of each stage but this is somewhat simplified view. What are the actual values that are encompassed in our random delay? Why is it more difficult to attack XOR PUF than arbiter PUF with machine learning (numerical modelling)?
We said that strong PUFs (those that give many different responses) are possible to be attacked by machine learning. On the other hand, weak PUFs (those that give only one or a few responses) are in essence not possible to be attacked with machine learning. Describe a situation involving any combination of PUFs where weak PUF can be also attacked by machine learning.

| Nr | Question | Points |
|---|---|---|
| 10 | SCA | 16 |

You work as a side-channel expert in a security evaluation company. You need to consult your clients who are major smartcard manufacturers but do not know anything about SCA.

What is distinguisher and what is leakage model in SCA? What is the distinguisher used in template attack? What are the conditions to mount template attack? What would happen with template attack if not all covariance matrices are defined?

Assume we conduct CPA on AES-256 in HW model. Why is it a good choice to attack after the S-box operation and not for instance before S-box operation? What would happen for DPA if you try to attack an "improved" version of AES that consists of only AddRoundKey, ShiftRows, and MixColumns operations (since you need to implement a cipher that is very cheap in hardware and you know that SubBytes is by far the most expensive operation in AES).

What are the conditions to mount a nonprofiling attack on AES in ECB mode (depicted in Figure 2)? When would you use HW model and when HD model in nonprofiling attacks?

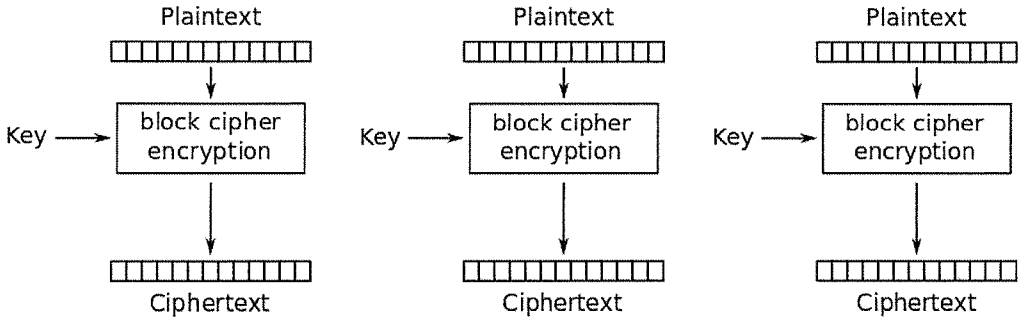


Figure 2.

| Nr | Question | Points |
|---|---|---|
| 11 | Countermeasures | 10 |

You explained well to your clients why SCA is dangerous and now they want to defend against it. Still, their implementation needs to be area-friendly so it is not possible to use threshold implementations. Consequently, you decide to use RSM. We implement AES-128 with RSM (Rotating S-box Masking countermeasure). How to conduct CPA attack on such a scheme (assume it is properly implemented)? Why do we need to mask the input value for AES in RSM if we assume that we cannot attack before the S-box part?

After you did good job implementing RSM, your clients are asking you to suggest how to make other algorithms they use more secure. They explain to you they use RSA and square-and-multiply algorithm. You tell them this is not a good choice, why? Instead, you propose to use Montgomery Ladder (Figure 3) or square-and-multiply-always algorithms (Figure 4).
Unfortunately, your clients do not understand those algorithms. You decide to explain them with a small example. Calculate 2^5 mod 7 with Montgomery Ladder technique and square-and-multiply-always technique. Show steps for the calculation.

```
Input: g, k = (k_{t-1}, ..., k_0)_2
Output: y = g^k
R_0 ← 1; R_1 ← g
for j = t − 1 downto 0 do
    R_{¬k_j} ← R_0 R_1; R_{k_j} ← (R_{k_j})^2
return R_0
```

Figure 3.

```
Input: g, k = (k_{t-1}, ..., k_0)_2
Output: y = g^k
R_0 ← 1; R_2 ← g
for j = t − 1 downto 0 do
    b ← ¬k_j
    R_0 ← (R_0)^2; R_b ← R_b R_2
return R_0
```

Figure 4.

Now your clients understand these techniques but they need to decide which one to use. Which of these two displayed algorithms is more secure? Consider a powerful attacker able to do side-channel attacks (SPA) and/or fault injection.

| Nr | Question | Points |
|---|---|---|
| 12 | Fault injection | 6 |

You work as a fault injection expert. You just received the latest batch of credit cards to test them. While inspecting the code you noticed the following block and you immediately see there is a place to attack it. What is that place?

```
for (i = 0; i < 6; i ++) {
        if (pin[i] == input[i])
                ok++;
}
if (ok == 6)
        respond_code (0x00, SW_NO_ERROR_msb, SW_NO_ERROR_lsb);
else
        respond_code (0x00, 0x69, 0x85);
```

Propose a countermeasure against it. Why would that work? Propose attack against that countermeasures (assume arbitrary powerful attacker). Note: depending on the countermeasure you propose, there could be no possible attack.

Listing 1: randomcatserver.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>          /* See NOTES */
#include <sys/socket.h>
#include <arpa/inet.h>
#include <strings.h>
#include <string.h>
#include <unistd.h>
#include <signal.h>
#include <errno.h>
#include <dirent.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <time.h>

#define BUFSIZE 4096
#define MAXFILES 40

/*
 * error - wrapper for perror used for bad syscalls
 */
void error(char *msg) {
  perror(msg);
  exit(1);
}

int randomFile(char * path) {
  DIR * dirp;
  struct dirent * entry;
  char files[MAXFILES][256];
  int file;
  int result;
  int file_count = 0;
  int dirfd;
  dirfd = open(path, O_RDONLY);
  dirp = fdopendir(dirfd);
  if (dirp == NULL) {
    error("Cannot open directory");
  }
  while ((entry = readdir(dirp)) != NULL) {
    if (entry->d_type == DT_REG) { /* If the entry is a regular file */
      file_count++;
      strncpy(files[file_count-1], entry->d_name, 256);
    }
  }
  srand(time(NULL));
  file = rand()%file_count;
  result = openat(dirfd, files[file], O_RDONLY);
  closedir(dirp);
  return result;
}
```

```
56  /* http://man7.org/tlpi/code/online/dist/sockets/read_line.c.html
57     Read characters from 'fd' until a newline is encountered. If a newline
58     character is not encountered in the first (n - 1) bytes, then the excess
59     characters are discarded. The returned string placed in 'buf' is
60     null-terminated and includes the newline character if it was read in the
61     first (n - 1) bytes. The function return value is the number of bytes
62     placed in buffer (which includes the newline character if encountered,
63     but excludes the terminating null byte). */
64
65  ssize_t
66  readLine(int fd, void *buffer, size_t n)
67  {
68      ssize_t numRead;                    /* # of bytes fetched by last read() */
69      size_t totRead;                     /* Total bytes read so far */
70      char *buf;
71      char ch;
72
73      if (n <= 0 || buffer == NULL) {
74          errno = EINVAL;
75          return -1;
76      }
77
78      buf = buffer;                       /* No pointer arithmetic on "void *" */
79
80      totRead = 0;
81      for (;;) {
82          numRead = read(fd, &ch, 1);
83
84          if (numRead == -1) {
85              if (errno == EINTR)         /* Interrupted --> restart read() */
86                  continue;
87              else
88                  return -1;              /* Some other error */
89
90          } else if (numRead == 0) {      /* EOF */
91              if (totRead == 0)           /* No bytes read; return 0 */
92                  return 0;
93              else                        /* Some bytes read; add '\0' */
94                  break;
95
96          } else {                        /* 'numRead' must be 1 if we get here */
97              if (totRead < n - 1) {      /* Discard > (n - 1) bytes */
98                  totRead++;
99                  *buf++ = ch;
100             }
101
102             if (ch == '\n')
103                 break;
104         }
105     }
106
107     *buf = '\0';
108     return totRead;
109 }
110
111
112 void process(int fd, struct sockaddr_in *clientaddr){
```

```
113     char buf[BUFSIZE];
114     const char * randomcatrequest = "GET /randomcat ";
115     const char * hellorequest = "GET / ";
116     const char * helloresponse = "HTTP/1.0 200 OK\r\nContent-Type: text/html\r\
            nConnection: close\r\n\r\nRandom cat image server!";
117     const char * randomcatresponse = "HTTP/1.0 200 OK\r\nContent-Type: image/jpeg\r
            \nConnection: close\r\n\r\n";
118     const char * notfound = "HTTP/1.0 404 Not Found\r\nConnection: close\r\n\r\n";
119     int filefd;
120     ssize_t length;
121
122     // TODO: Add Seccomp rules here!
123
124     length = readLine(fd, buf, BUFSIZE);
125     if (length <= 0) {
126       error("Failed to read request");
127     }
128     if (strncmp(buf, hellorequest, strlen(hellorequest)) == 0) {
129       write(fd, helloresponse, strlen(helloresponse));
130     } else if (strncmp(buf, randomcatrequest, strlen(randomcatrequest)) == 0) {
131       printf("Random cat\n");
132       write(fd, randomcatresponse, strlen(randomcatresponse));
133       filefd = randomFile("images");
134       if (filefd < 0) {
135         error("Cannot open file");
136       }
137       int bytesRead = read(filefd, buf, BUFSIZE);
138       while (bytesRead > 0) {
139         write(fd, buf, bytesRead);
140         bytesRead = read(filefd, buf, BUFSIZE);
141       }
142     } else {
143       write(fd, notfound, strlen(notfound));
144     }
145     shutdown(fd, SHUT_WR);
146     close(fd);
147     return;
148   }
149
150   int main(int argc, char** argv) {
151     int parentfd;          /* parent socket */
152     int childfd;           /* child socket */
153     struct sockaddr_in serveraddr; /* server's addr */
154     struct sockaddr_in clientaddr; /* client addr */
155     socklen_t clientlen;         /* byte size of client's address */
156     int optval;            /* flag value for setsockopt */
157     pid_t pid;
158
159     if (signal(SIGCHLD, SIG_IGN) == SIG_ERR) {
160       perror(0);
161       exit(1);
162     }
163
164     //srand(time(NULL));
165
166
167     /* open socket descriptor */
```

```
168     parentfd = socket(AF_INET, SOCK_STREAM, 0);
169     if (parentfd < 0)
170       error("ERROR opening socket");
171
172     /* allows us to restart server immediately */
173     optval = 1;
174     setsockopt(parentfd, SOL_SOCKET, SO_REUSEADDR,
175          (const void *)&optval , sizeof(int));
176
177     /* bind port to socket */
178     bzero((char *) &serveraddr, sizeof(serveraddr));
179     serveraddr.sin_family = AF_INET;
180     serveraddr.sin_addr.s_addr = htonl(INADDR_ANY);
181     serveraddr.sin_port = htons((unsigned short)5000);
182     if (bind(parentfd, (struct sockaddr *) &serveraddr,
183        sizeof(serveraddr)) < 0)
184       error("ERROR on binding");
185
186     /* get us ready to accept connection requests */
187     if (listen(parentfd, 5) < 0) /* allow 5 requests to queue up */
188       error("ERROR on listen");
189
190     /*
191      * main loop: wait for a connection request, parse HTTP,
192      * serve requested content, close connection.
193      */
194     clientlen = sizeof(clientaddr);
195     while (1) {
196       /* wait for a connection request */
197       childfd = accept(parentfd, (struct sockaddr *) &clientaddr, &clientlen);
198       printf("New connection: %d\n", childfd);
199       if (childfd < 0)
200         error("ERROR on accept");
201       pid = fork();
202       if (pid < 0) {
203         error("Cannot fork");
204       }
205       if (pid == 0) {
206         // child
207         close(parentfd);
208         process(childfd, &clientaddr);
209         return 0;
210       } else {
211         close(childfd);
212       }
213     }
214
215 }
```

Listing 2: randomcatserver-nofork.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <sys/types.h>          /* See NOTES */
4  #include <sys/socket.h>
5  #include <arpa/inet.h>
6  #include <strings.h>
7  #include <string.h>
8  #include <unistd.h>
9  #include <signal.h>
10 #include <errno.h>
11 #include <dirent.h>
12 #include <sys/stat.h>
13 #include <fcntl.h>
14 #include <time.h>
15
16 #define BUFSIZE 4096
17 #define MAXFILES 40
18
19 /*
20  * error - wrapper for perror used for bad syscalls
21  */
22 void error(char *msg) {
23   perror(msg);
24   exit(1);
25 }
26
27 int randomFile(char * path) {
28   DIR * dirp;
29   struct dirent * entry;
30   char files[MAXFILES][256];
31   int file;
32   int result;
33   int file_count = 0;
34   int dirfd;
35   dirfd = open(path, O_RDONLY);
36   dirp = fdopendir(dirfd);
37   if (dirp == NULL) {
38     error("Cannot open directory");
39   }
40   while ((entry = readdir(dirp)) != NULL) {
41     if (entry->d_type == DT_REG) { /* If the entry is a regular file */
42       file_count++;
43       strncpy(files[file_count-1], entry->d_name, 256);
44     }
45   }
46   srand(time(NULL));
47   file = rand()%file_count;
48   result = openat(dirfd, files[file], O_RDONLY);
49   closedir(dirp);
50   return result;
51 }
52
53
54
55
```

```
56  /* http://man7.org/tlpi/code/online/dist/sockets/read_line.c.html
57     Read characters from 'fd' until a newline is encountered. If a newline
58     character is not encountered in the first (n - 1) bytes, then the excess
59     characters are discarded. The returned string placed in 'buf' is
60     null-terminated and includes the newline character if it was read in the
61     first (n - 1) bytes. The function return value is the number of bytes
62     placed in buffer (which includes the newline character if encountered,
63     but excludes the terminating null byte). */
64
65  ssize_t
66  readLine(int fd, void *buffer, size_t n)
67  {
68      ssize_t numRead;                    /* # of bytes fetched by last read() */
69      size_t totRead;                     /* Total bytes read so far */
70      char *buf;
71      char ch;
72
73      if (n <= 0 || buffer == NULL) {
74          errno = EINVAL;
75          return -1;
76      }
77
78      buf = buffer;                       /* No pointer arithmetic on "void *" */
79
80      totRead = 0;
81      for (;;) {
82          numRead = read(fd, &ch, 1);
83
84          if (numRead == -1) {
85              if (errno == EINTR)         /* Interrupted --> restart read() */
86                  continue;
87              else
88                  return -1;              /* Some other error */
89
90          } else if (numRead == 0) {      /* EOF */
91              if (totRead == 0)           /* No bytes read; return 0 */
92                  return 0;
93              else                        /* Some bytes read; add '\0' */
94                  break;
95
96          } else {                        /* 'numRead' must be 1 if we get here */
97              if (totRead < n - 1) {      /* Discard > (n - 1) bytes */
98                  totRead++;
99                  *buf++ = ch;
100             }
101
102             if (ch == '\n')
103                 break;
104         }
105     }
106
107     *buf = '\0';
108     return totRead;
109 }
110
111
112 void process(int fd, struct sockaddr_in *clientaddr){
```

```
113     char buf[BUFSIZE];
114     const char * randomcatrequest = "GET /randomcat ";
115     const char * hellorequest = "GET / ";
116     const char * helloresponse = "HTTP/1.0 200 OK\r\nContent-Type: text/html\r\
            nConnection: close\r\n\r\nRandom cat image server!";
117     const char * randomcatresponse = "HTTP/1.0 200 OK\r\nContent-Type: image/jpeg\r
            \nConnection: close\r\n\r\n";
118     const char * notfound = "HTTP/1.0 404 Not Found\r\nConnection: close\r\n\r\n";
119     int filefd;
120     ssize_t length;
121
122     // TODO: Add Seccomp rules here!
123
124     length = readLine(fd, buf, BUFSIZE);
125     if (length <= 0) {
126       error("Failed to read request");
127     }
128     if (strncmp(buf, hellorequest, strlen(hellorequest)) == 0) {
129       write(fd, helloresponse, strlen(helloresponse));
130     } else if (strncmp(buf, randomcatrequest, strlen(randomcatrequest)) == 0) {
131       printf("Random cat\n");
132       write(fd, randomcatresponse, strlen(randomcatresponse));
133       filefd = randomFile("images");
134       if (filefd < 0) {
135         error("Cannot open file");
136       }
137       int bytesRead = read(filefd, buf, BUFSIZE);
138       while (bytesRead > 0) {
139         write(fd, buf, bytesRead);
140         bytesRead = read(filefd, buf, BUFSIZE);
141       }
142     } else {
143       write(fd, notfound, strlen(notfound));
144     }
145     shutdown(fd, SHUT_WR);
146     close(fd);
147     return;
148   }
149
150   int main(int argc, char** argv) {
151     int parentfd;          /* parent socket */
152     int childfd;           /* child socket */
153     struct sockaddr_in serveraddr; /* server's addr */
154     struct sockaddr_in clientaddr; /* client addr */
155     socklen_t clientlen;         /* byte size of client's address */
156     int optval;            /* flag value for setsockopt */
157     pid_t pid;
158
159     if (signal(SIGCHLD, SIG_IGN) == SIG_ERR) {
160       perror(0);
161       exit(1);
162     }
163
164     //srand(time(NULL));
165
166
167     /* open socket descriptor */
```

```
168     parentfd = socket(AF_INET, SOCK_STREAM, 0);
169     if (parentfd < 0)
170       error("ERROR opening socket");
171
172     /* allows us to restart server immediately */
173     optval = 1;
174     setsockopt(parentfd, SOL_SOCKET, SO_REUSEADDR,
175          (const void *)&optval , sizeof(int));
176
177     /* bind port to socket */
178     bzero((char *) &serveraddr, sizeof(serveraddr));
179     serveraddr.sin_family = AF_INET;
180     serveraddr.sin_addr.s_addr = htonl(INADDR_ANY);
181     serveraddr.sin_port = htons((unsigned short)5000);
182     if (bind(parentfd, (struct sockaddr *) &serveraddr,
183        sizeof(serveraddr)) < 0)
184       error("ERROR on binding");
185
186     /* get us ready to accept connection requests */
187     if (listen(parentfd, 5) < 0) /* allow 5 requests to queue up */
188       error("ERROR on listen");
189
190     /*
191      * main loop: wait for a connection request, parse HTTP,
192      * serve requested content, close connection.
193      */
194     clientlen = sizeof(clientaddr);
195     while (1) {
196       /* wait for a connection request */
197       childfd = accept(parentfd, (struct sockaddr *) &clientaddr, &clientlen);
198       printf("New connection: %d\n", childfd);
199       if (childfd < 0)
200         error("ERROR on accept");
201       /* pid = fork();
202       if (pid < 0) {
203         error("Cannot fork");
204       }
205       if (pid == 0) {
206         // child
207         close(parentfd); */
208         process(childfd, &clientaddr);
209       /* return 0;
210       } else {
211         close(childfd);
212       } */
213     }
214
215 }
```