Aufgabenblatt 1
Sicherheit digitaler Systeme

Institut: Berliner Hochschule für Technik
Dozent: Prof. Dr. Christian Forler
Url: `https://lms.bht-berlin.de/`
Email: `cforler(at)bht-berlin.de`
Wintersemester 23/24

Aufgabe 1 Sicherheitslücken
Laden sie die folgenden Programme herunter und bringen Sie diese zum Absturz und Fixen Sie im Anschluss die Sicherheitslücke.

a) `https://raw.githubusercontent.com/Moktur/ProgrammierenC/pepe/%C3%9Cbungsaufgaben/A0407.c`

b) `https://github.com/alandipert/minirpn`
Hier müssen sie ggf. noch das Makefile anpassen.

Aufgabe 2 You win **(4 Punkte)**

a) Finden Sie eine Nutzereingabe, welche bei dem folgenden Programm `You Win` ausgibt. Falls Sie unter Ihrem Betriebssystem Probleme mit der Aufgabe haben, dann verwenden sie bitte das Compilerflag `-fno-stack-protector`.

b) Fixen Sie den Bug. Schreiben Sie das untere Progamm so um, dass es für keine Benutzereingabe den String `You Win` ausgibt.

```c
#include <stdio.h>
#include <stdlib.h>

enum { BUFFER_SIZE = 10 };

int main() {
    int check1 = 0;
    char buffer[BUFFER_SIZE];
    int check2 = 0;

    scanf("%s",buffer);

    if( check1 || check2 ) puts("You Win");

    return EXIT_SUCCESS;
}
```

Aufgabe 3 Taschenrechner **(4 Punkte)**

Finden Sie für das folgende Programm eine Eingabe, so dass der Taschenrechner `xcalc` aufgerufen wird. Falls Sie unter Ihrem Betriebssystem Probleme mit der Aufgabe haben, dann verwenden sie bitte das Compilerflag `-fno-stack-protector`.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main() {
  char cmd[6];
  strcpy(cmd,"clear");

  char name[10];
  scanf("%s",name);

  system(cmd);
  printf("Hallo %s\n",name);
}
```

Aufgabe 4 Sendfiles

Machen Sie sich mit der folgenden naive quick-and-dirty Client-Server Lösung zum Übertragen von Dateien vertraut. Diese funktioniert wie folgt:

- Der Client bekommt einen Dateinamen übergeben.
- Der Client sendet den Dateinamen an den Server
- Der Server startet einen neuen Thread.
- Der Server-Thread empfängt den Dateinamen und legt die entsprechende Datei an.
- Der Client sendet den Inhalt der Datei.
- Der Server-Thread empängt den Inhalt der Datei und speichert diese in der angelegten Datei.

Sie finden die Quellen im Moodelkurs. In den kommenden Übungsblätter sollen Sie sukessive Sicherheitsfeatures implementieren um die Vertraulichkeit und Integrität der übertragenen Dateien zu schützen.
Testen Sie die Implementation. Sollte die Datei-Übertragung auf Ihrem System nicht erfolgreich sein, dann erhöhen Sie den TIMEOUT in dem Headerfile `utils.h`. Sollte dies das Problem nicht lösen, dann ist debuggen angesagt. Dazu können Sie sich an mich wenden.

Client

```c
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <arpa/inet.h>
```

```
7  #include <stdbool.h>
8  #include <fcntl.h>
9  #include <dirent.h>
10
11 #include "helper.h"
12
13 static void usage() {
14   fputs("Usage: ./client file\n",stderr);
15   exit(EXIT_FAILURE);
16 }
17
18
19 void send_file(int fd, int sockfd, const char *filename){
20   if (write(sockfd, filename, strlen(filename)) < 1) ERROR(
       filename);
21   fsync(sockfd);
22   usleep(TIMEOUT);
23   while(true) {
24     char buf[BUF_LEN];
25     int n = read(fd, buf, BUF_LEN);
26     if(n == 0) break;
27
28     if (write(sockfd, buf, n) <0) error(true, errno,"%s",
       filename);
29   }
30 }
31
32
33 int open_server_connection( const char *ip, const int port)
       {
34   int sockfd = socket(AF_INET, SOCK_STREAM, 0);
35   if(sockfd < 0) ERROR("socket");
36
37
38   struct sockaddr_in server_addr;
39   server_addr.sin_family = AF_INET;
40   server_addr.sin_port =  htons(port);
41   server_addr.sin_addr.s_addr = inet_addr(ip);
42
43   if ( connect(sockfd, (struct sockaddr*)&server_addr,
44                sizeof(server_addr)) == -1)  ERROR("connect")
       ;
45
46   return sockfd;
47 }
48
49
50 int main(int args, char *argv[]){
51   if(args != 2) usage();
52
53   const char *ip = "127.0.0.1";
54   //const char *ip = "192.168.2.108";
55
56
```

```
57    int sockfd = open_server_connection(ip,SERVER_PORT);
58
59    int fd = open(argv[1], O_RDONLY);
60    if (fd <0) ERROR(argv[1]);
61
62    send_file(fd, sockfd, argv[1]);
63
64    close(fd);
65    close(sockfd);
66  }
```

Server

```
1   #include <stdio.h>
2   #include <stdint.h>
3   #include <stdlib.h>
4   #include <string.h>
5   #include <arpa/inet.h>
6   #include <stdbool.h>
7   #include <unistd.h>
8   #include <fcntl.h>
9   #include <sys/stat.h>
10  #include <pthread.h>
11  #include <libgen.h>
12
13  #include "helper.h"
14
15  void write_file(int sockfd){
16    char buf[BUF_LEN];
17
18    int n = read(sockfd, buf, BUF_LEN-1);
19    if(n<0) ERROR("read");
20    buf[n] = '\0';
21    char *bn = basename(buf);
22    int fd = open(bn, O_CREAT | O_WRONLY | O_EXCL, 0664);
23
24    if(fd<0) WARN(buf);
25    while (true) {
26      n = read(sockfd, buf, BUF_LEN);
27      if (n <= 0) break;
28      if (write(fd,buf,n) != n) ERROR("write");
29    }
30    close(fd);
31  }
32
33
34  void* client_handler(void *sockfd) {
35    int *socket = sockfd;
36    write_file(*socket);
37
38    close(*socket);
39    free(sockfd);
40    return NULL;
41  }
```

```
42
43 int open_server_socket(const char *ip, int port) {
44   int sockfd = socket(AF_INET, SOCK_STREAM, 0);
45   if(sockfd < 0) ERROR("socket");
46
47   struct sockaddr_in server_addr;
48   bzero(&server_addr, sizeof(server_addr));
49   server_addr.sin_family = AF_INET;
50   server_addr.sin_port = htons(port);
51   server_addr.sin_addr.s_addr = inet_addr(ip);
52
53   if ( bind(sockfd, (struct sockaddr*) &server_addr, sizeof(
       struct sockaddr_in)) != 0)
54     ERROR("bind");
55
56   if(listen(sockfd, 10) != 0) ERROR("listen");
57   return sockfd;
58 }
59
60
61
62 int main(){
63   const char *ip = "127.0.0.1";
64
65   int sockfd = open_server_socket(ip, SERVER_PORT);
66
67 #ifndef DEBUG
68   if( daemon(true, true) == -1) ERROR("daemon");
69 #endif
70
71   pthread_t t;
72   struct sockaddr_in new_addr;
73   while(true) {
74     socklen_t  addr_size = sizeof(new_addr);
75     int *sockp = malloc(sizeof(int));
76     *sockp = accept(sockfd, (struct sockaddr*) &new_addr, &
      addr_size);
77
78     if (pthread_create(&t, NULL, client_handler, sockp))
      WARN("pthread_create");
79     pthread_detach(t);
80   }
81
82 }
```