BLG 460E SECURE PROGRAMMING

HW2

Ahmet Göktuğ SEVİNÇ

150140120

13.03.2019

Q1)

In the first question we were asked to analyze the given code and determine how a attacker can skip validation performed in code. The code expects *"/home*" path from the user. To be able to compare if the valid path is sent or not, it just checks first 5 characters of the entered path and compares it with the "*/home*". If they match, code considers path as a valid path. However, this validation process enables attacker to use paths that are not canonical. For example, attacker can use *"/home/.."* path. Since first 5 characters are *"/home"*, this path will pass the validation test but instead of accessing *home* directory, we will access to the *root* directory. Below image shows list of directories in *root* path.

```
sevinca@sevinca-VirtualBox:~/Desktop$ ./dene /home/../../..
Process file executes
total 100
drwxr-xr-x  22 root root  4096 Eki 15 01:39 .
drwxr-xr-x  22 root root  4096 Eki 15 01:39 ..
drwxr-xr-x   2 root root  4096 Eki 15 01:40 bin
drwxr-xr-x   3 root root  4096 Kas 18 13:48 boot
drwxrwxr-x   2 root root  4096 Eki 15 01:37 cdrom
drwxr-xr-x  17 root root  4180 Mar 13 12:45 dev
drwxr-xr-x 120 root root 12288 Mar 13 12:48 etc
drwxr-xr-x   3 root root  4096 Eki 15 01:39 home
lrwxrwxrwx   1 root root    32 Eki 15 01:39 initrd.img -> boot/initrd.img-4.4.0-31-generic
drwxr-xr-x  22 root root  4096 Eki 15 01:55 lib
drwx------   2 root root 16384 Eki 15 01:36 lost+found
drwxr-xr-x   4 root root  4096 Eki 24 21:10 media
drwxr-xr-x   2 root root  4096 Nis 11  2014 mnt
drwxr-xr-x   3 root root  4096 Eki 24 21:03 opt
dr-xr-xr-x 174 root root     0 Mar 13 12:45 proc
drwx------   4 root root  4096 Kas 18 14:07 root
drwxr-xr-x  23 root root   740 Mar 13 12:48 run
drwxr-xr-x   2 root root 12288 Eki 24 21:04 sbin
drwxr-xr-x   2 root root  4096 Ağu  3  2016 srv
dr-xr-xr-x  13 root root     0 Mar 13 12:50 sys
drwxrwxrwt   5 root root  4096 Mar 13 12:50 tmp
drwxr-xr-x  10 root root  4096 Ağu  3  2016 usr
drwxr-xr-x  13 root root  4096 Ağu  3  2016 var
lrwxrwxrwx   1 root root    29 Eki 15 01:39 vmlinuz -> boot/vmlinuz-4.4.0-31-generic
sevinca@sevinca-VirtualBox:~/Desktop$
```

Also, since we have access to the *root* directory, we can access any path in the file system. For example, we can use *"/home/../dev"* path or other paths inside *dev* directory.

To prevent directory traversal attacks, we can request user to enter canonical paths and prevent usage of other paths. To do this in given code, we can check if the path user entered is containing any escape characters such as "..", . ". If we add below code;

```
bool checkEscape(char *path){
    if(strstr(path, "..") != NULL || strstr(path, ".") != NULL){
        return false;
    }
    return true;
}
```

We obtain following result;

```
sevinca@sevinca-VirtualBox:~/Desktop$ ./dene /home/..
Path specified is not valid !
sevinca@sevinca-VirtualBox:~/Desktop$
```

Q2)

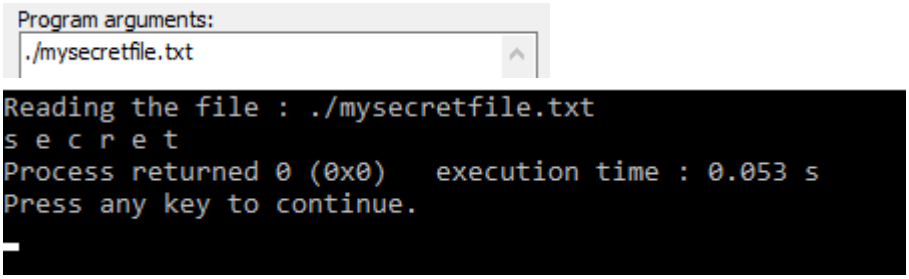
In the second question we were asked to analyze the code and explain and try to find a solution to the possible attacks. In the code, we have a file called “mysecretfile.txt” that we don’t want to be accessible. There are several ways to attack the written code and some of them are listed below:

- We can access the file by using “., ..” operators.

  Linux:

```
sevinca@sevinca-VirtualBox:~/Desktop$ ./dene  ./../Desktop/mysecretfile
Reading the file : ./../Desktop/mysecretfile
gizli
```

  Windows:

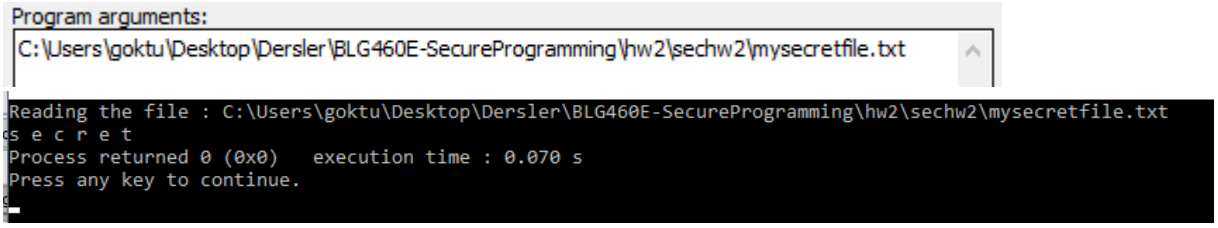


- We can access the file by giving its full path.

  Linux:

```
sevinca@sevinca-VirtualBox:~/Desktop$ ./dene /home/sevinca/Desktop/mysecretfile
Reading the file : /home/sevinca/Desktop/mysecretfile
gizli
```

  Windows:

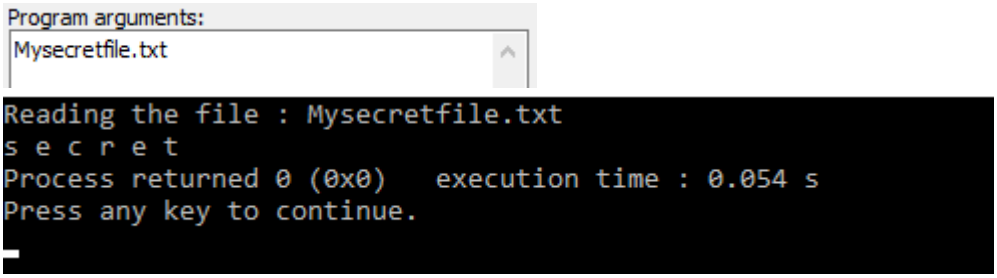


- On Windows operating system, reading file operation is not case sensitive. So, we can use that property to access the file but in Linux we cannot apply this operation.

  Linux:

```
sevinca@sevinca-VirtualBox:~/Desktop$ ./dene Mysecretfile
Reading the file : Mysecretfile
Error: No such file or directory
```

Windows:



To overcome those attacks, firstly I converted all letters of the given path to lowercase characters and then checked if the given path contains *mysecretfile.txt*. If it contains, I prevented to access to the path.

```
char *fn = argv [ 1 ] ;
char *temp = fn;

while(*temp){ //convert to lower characters
   *temp = tolower((unsigned char)*temp);
   temp++;
}

if(strstr(fn, protectedfile)==NULL){ //if private file does not exist in given path
   if (strcmp (fn , protectedfile) != 0) {
      readFile ( fn ) ;
   }
}
```