System and Software Security 2025

Fall Assignment 01

Made by aimeric, student id : 414410902

Homework one

first question :

explain the buffer overflow here

```
void queryLogin()
{
   char sql[256];
   char *p = sql;
   char userid[64];
   char passwd[64];
   strcpy(p, "UPDATE user_data SET passwd = '"); p += strlen(p);
   gets(userid);
   gets(passwd);
   strcpy(p, passwd); p += strlen(p);
   strcpy(p, "' WHERE userid = '"); p += strlen(p);
   strcpy(p, userid); p += strlen(p);
   strcpy(p, "'");
   db_query(sql);
}
```

- by entering a passwork or a username larger than 64 bits it will make the value overflow
- The function uses unsafe, unbounded input (gets, strcpy) and put the inputs into a stack buffer sql[256]. Any input larger than the destination size can overflow in the stack memory.
- also The code also builds SQL by using user input, which creates SQL injection risk in addition to buffer overflows.

Second question :

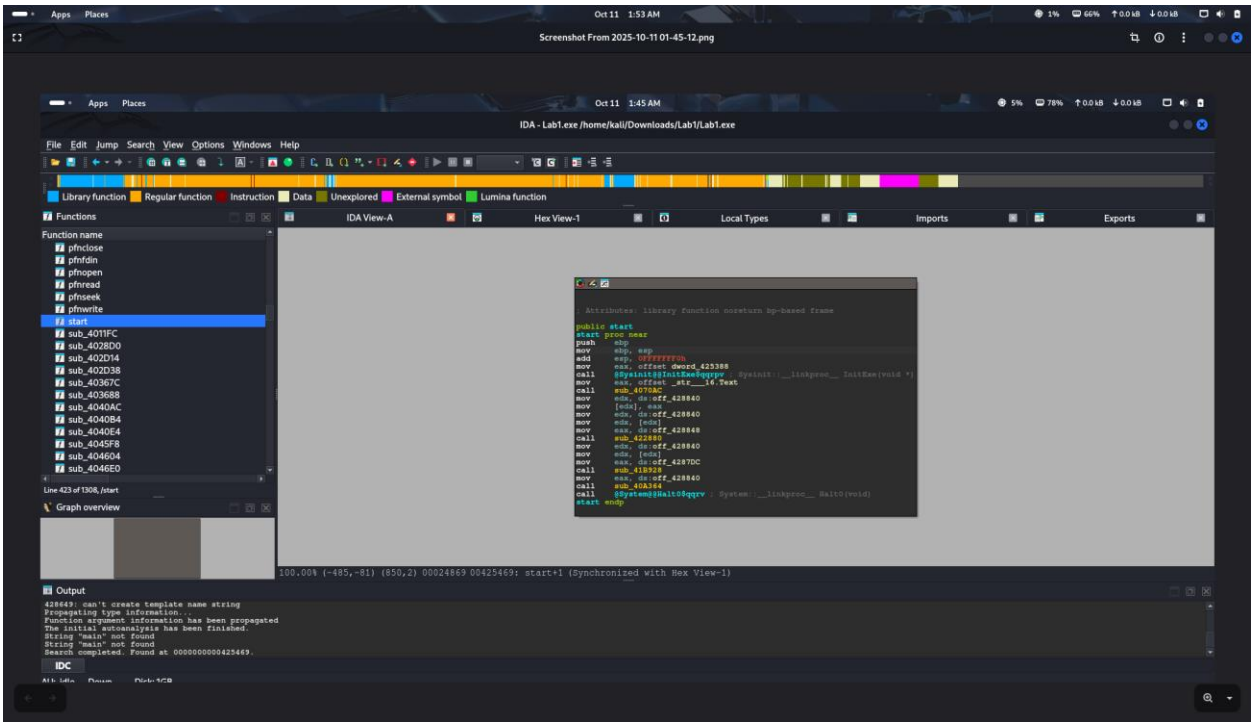
```
#include <stdio.h>
#include <string.h>
void safe_call_function() {
   char buffer[10];
   char input[100];
   printf("Enter some texts: ");
   gets(input);
   if (strlen(input) < 10) {
      strncpy(buffer, input, sizeof(input));
   } else {
      printf("Input too long!\n");
   }
   printf("Buffer contains: %s\n", buffer);
}

int main() {
   printf("== Fix bug ==\n");
   safe_call_function();
   return 0;
}
```

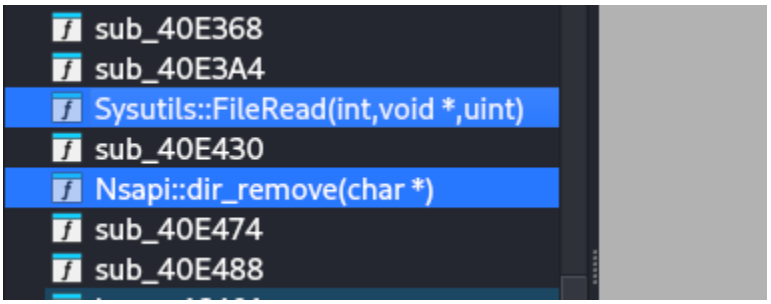
- this code looks safe but it use get, gets is unsafe, it reads unlimited input and put them in input[100] it will overflow it
- Wrong size argument to strncpy(buffer, input, sizeof(input)) uses sizeof(input) (max size 100) when copying to buffer (max size 10). That writes up to 100 bytes into a 10-byte destination -> guaranteed stack overflow. This single mistake completely make the code vulnerable.
- also print the buffer even when not initialised, if we go to the else but still print the buffer you will print an uninitialised buffer

Third question :

We had a file called Lab1.zip and we had to show the list of function invoqued by the virus so after setting up kali linux then ida pro i reverse engeniered the .exe file and here you can see the main function and on the left the list of functions called by the programm

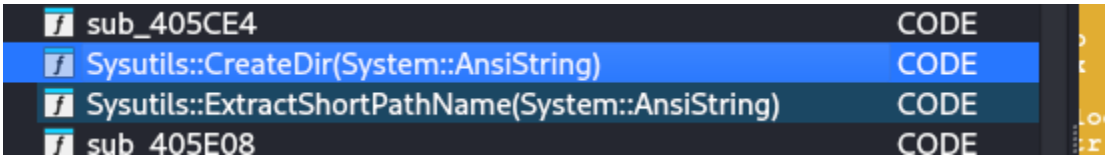


I could for example show :



We see one function made to read file and one function made to delete folder (probably using the name)

Or :



Made to create directory or exact path name

But the file use way too many function to be able to list al of them

Fourth question :

When we execute the command sha256 on the file test.txt given in the document we get :

```
$ sha256sum test.txt
12b1c3270175d0d61ee58866fe193e58932cf19319fa4448d4685dfcbf24ccac  test.txt
```

So the hash in form of sha256 of the test.txt file is
12b1c3270175d0d61ee58866fe193e58932cf19319fa4448d4685dfcbf24ccac

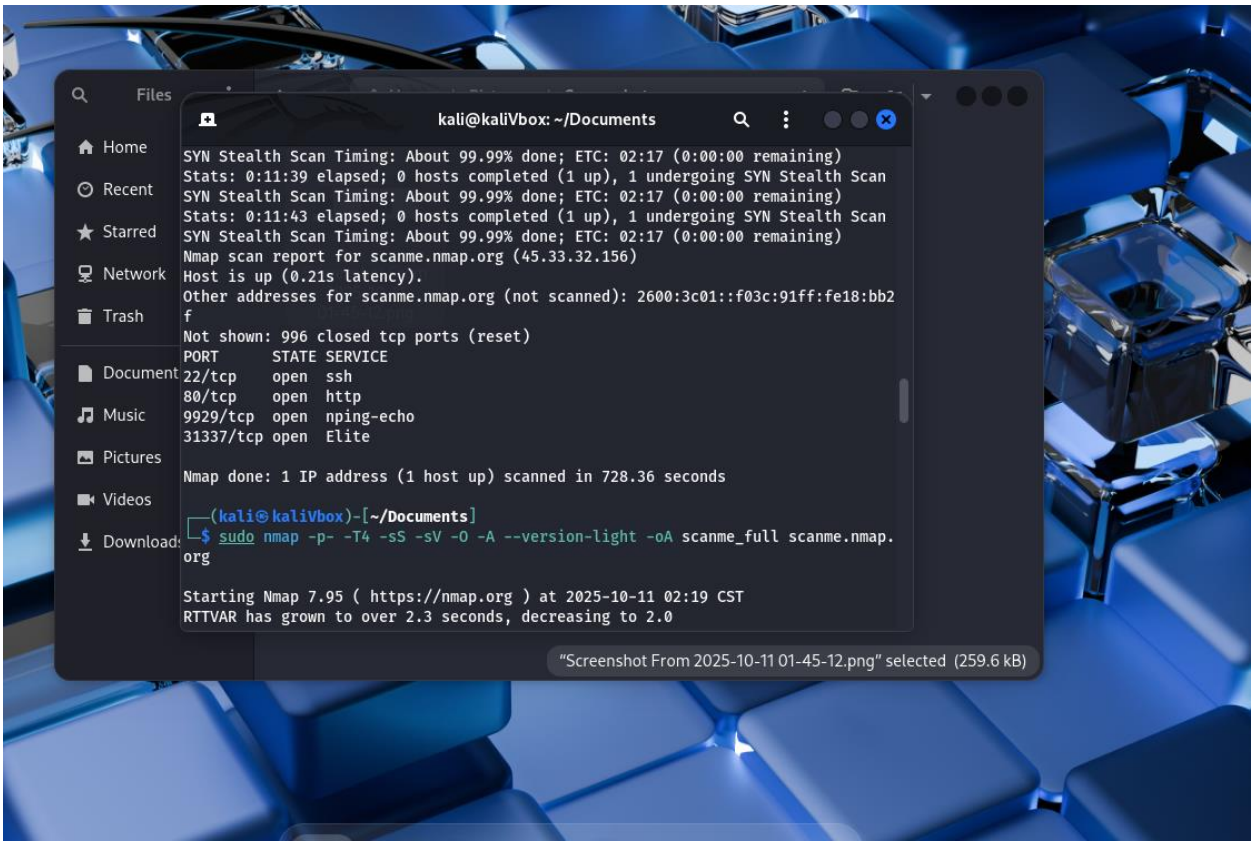
```
$ sha1sum test.txt
369f2f81e72addb9fc473a5e04b39f31aacf31d6  test.txt
```

And the hash in form of sha1 is 369f2f81e72addb9fc473a5e04b39f31aacf31d6

Last question:

Scan the adress : http://scanme.nmap.org/

Using nmap and show the result :



From this result what kind of vulnerability can we find :

Port 22/ssh : Weak or credentials used at multiple places –> easy to take control of the server

Port 80/https : Web vulnerabilities: XSS, SQL injection

Port 31337 :
Could be a malicious/backdoor service (remote shell or custom protocol).

Could contain buffer overflow vulnerabilities in authentification code code.