Advanced topics in computer security

Challenges based on course provided in 2019-2020 year at Politecnico di Milano

Student: Kasenov Erbol

ID: 10674231

The first challenge: Revers engineering

Here I want to represent basic of reverse engineering.

Firstly, we start from code part, I create code script in C programming language:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void password ( char input[16]{
        if strlen(input)>16){
                printf("Don't try overflow,put exact 16 byte length\n");
                exit(0);
        }
         char pass[16];
        strcopy(pass,"ilovepolitecnico");
        if strcmp(input,pass)==0){
                printf("congratulation you arr hacker");
        }else{
                Printf("wrong password ,do u love polimi?");
        }
}

void main (int argc, char *argv[]){
        printf("Hi, I am packed ,try to recognize password");
        if(argv[1]!=NULL){
                password(argv[1]);
        }else{
                Printf("u didn't enter the password ");
        }
}
```

In code required the password which is "ilovepolitecnico",if user print nothing code says :"u did not enter password" else if user tries overflow the input code says: «don't try to overflow ,put exact 16 byte long password.

The compilation of code. Firstly compile the code: `$gcc rever.c -o rever`

After compilation need to strip the program, because without stripping the reversing would be so easy. Use command `strip rever.c`

Overall, we have executable file which requires the password. Let us start to observe the password (of course I assume that we don't know the password and source code)

What we need:

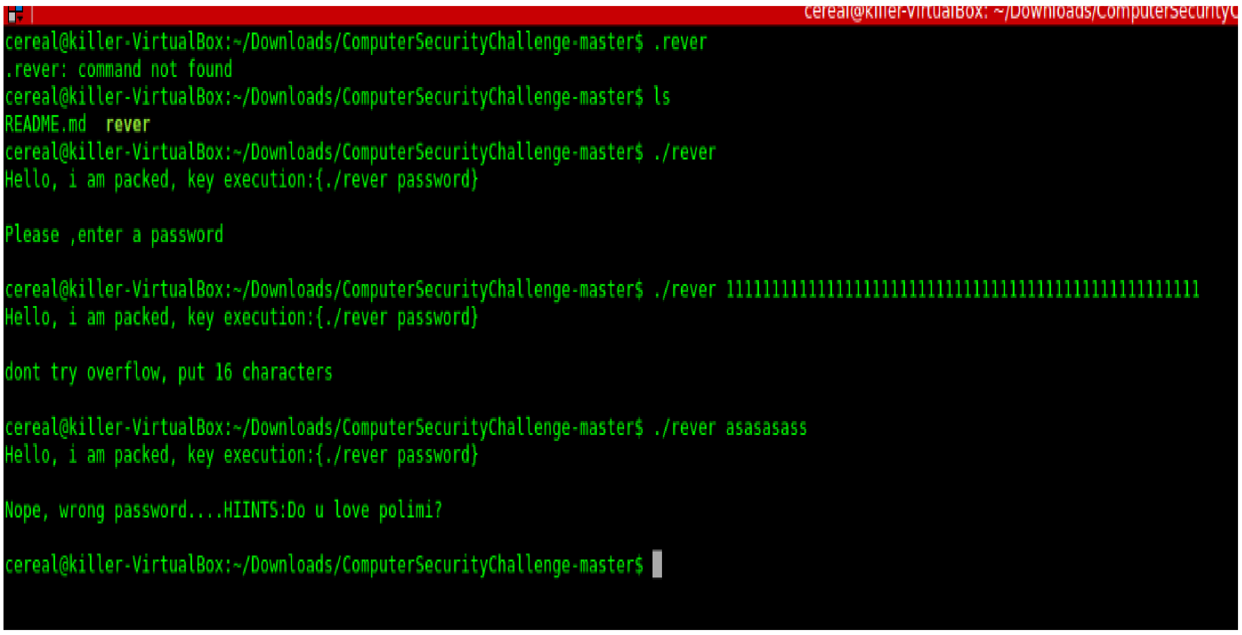
-Linux

-Ghidra program

-Hex-text convertor

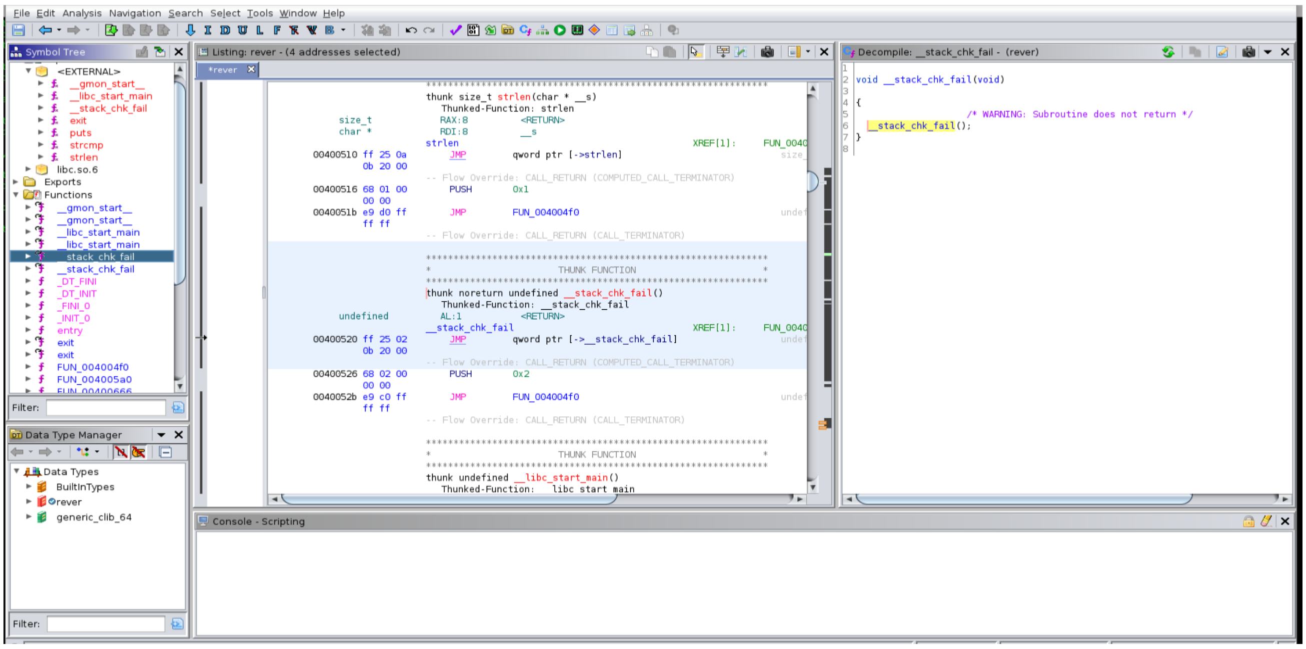
Just want to mention that using GDB will not exactly help to solve challenge because all functions are stripped and changed.

Try to run **rever** executable file:



```
cereal@killer-VirtualBox: ~/Downloads/ComputerSecurityC
cereal@killer-VirtualBox:~/Downloads/ComputerSecurityChallenge-master$ .rever
.rever: command not found
cereal@killer-VirtualBox:~/Downloads/ComputerSecurityChallenge-master$ ls
README.md  rever
cereal@killer-VirtualBox:~/Downloads/ComputerSecurityChallenge-master$ ./rever
Hello, i am packed, key execution:{./rever password}

Please ,enter a password

cereal@killer-VirtualBox:~/Downloads/ComputerSecurityChallenge-master$ ./rever 11111111111111111111111111111111111111111111111111111
Hello, i am packed, key execution:{./rever password}

dont try overflow, put 16 characters

cereal@killer-VirtualBox:~/Downloads/ComputerSecurityChallenge-master$ ./rever asasasass
Hello, i am packed, key execution:{./rever password}

Nope, wrong password....HIINTS:Do u love polimi?

cereal@killer-VirtualBox:~/Downloads/ComputerSecurityChallenge-master$
```

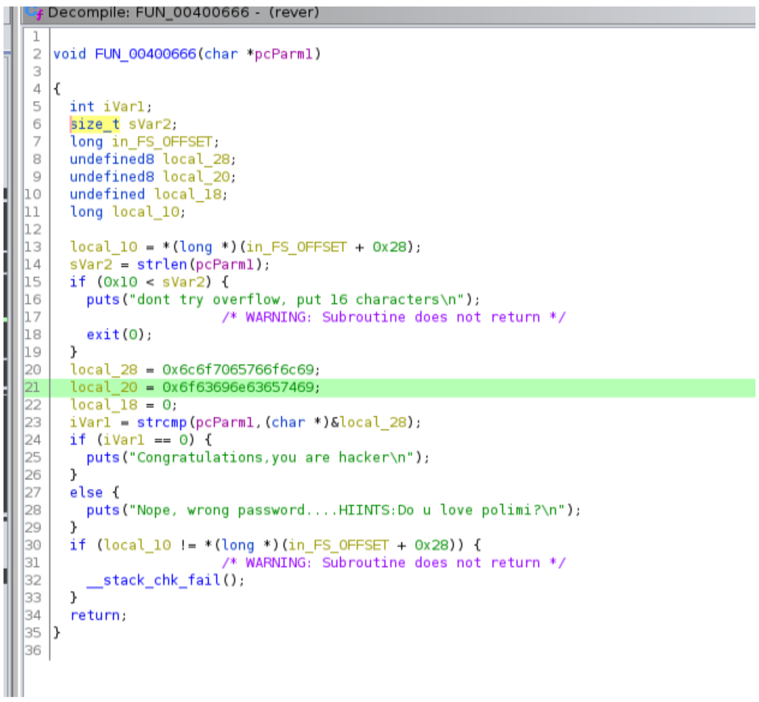
Lets open Ghidra and try to analyze the file



```
                   ****************************************************************
                   thunk size_t strlen(char * __s)
                     Thunked-Function: strlen
     size_t          RAX:8          <RETURN>
     char *          RDI:8          __s
                   strlen                                    XREF[1]:     FUN_0040
00400510 ff 25 0a     JMP          qword ptr [->strlen]
         0b 20 00
                   -- Flow Override: CALL_RETURN (COMPUTED_CALL_TERMINATOR)
00400516 68 01 00     PUSH         0x1
         00 00
0040051b e9 d0 ff     JMP          FUN_004004f0
         ff ff
                   -- Flow Override: CALL_RETURN (CALL_TERMINATOR)

                   ****************************************************************
                   *                       THUNK FUNCTION                         *
                   ****************************************************************
                   thunk noreturn undefined __stack_chk_fail()
                     Thunked-Function: __stack_chk_fail
     undefined       AL:1           <RETURN>
                   __stack_chk_fail                          XREF[1]:     FUN_0040
00400520 ff 25 02     JMP          qword ptr [->__stack_chk_fail]
         0b 20 00
                   -- Flow Override: CALL_RETURN (COMPUTED_CALL_TERMINATOR)
00400526 68 02 00     PUSH         0x2
         00 00
0040052b e9 c0 ff     JMP          FUN_004004f0
         ff ff
                   -- Flow Override: CALL_RETURN (CALL_TERMINATOR)

                   ****************************************************************
                   *                       THUNK FUNCTION                         *
                   ****************************************************************
                   thunk undefined __libc_start_main()
                     Thunked-Function:   libc start main
```

```
void __stack_chk_fail(void)

{
                    /* WARNING: Subroutine does not return */
  __stack_chk_fail();
}
```

As u can see the common functions are changed, for example I couldn't find -main function just typing it in search field. So since open the file need to analyze the code and find interesting parts of file in ghidra



```
1
2  void FUN_00400666(char *pcParm1)
3
4  {
5    int iVar1;
6    size_t sVar2;
7    long in_FS_OFFSET;
8    undefined8 local_28;
9    undefined8 local_20;
10   undefined local_18;
11   long local_10;
12
13   local_10 = *(long *)(in_FS_OFFSET + 0x28);
14   sVar2 = strlen(pcParm1);
15   if (0x10 < sVar2) {
16     puts("dont try overflow, put 16 characters\n");
17                     /* WARNING: Subroutine does not return */
18     exit(0);
19   }
20   local_28 = 0x6c6f7065766f6c69;
21   local_20 = 0x6f63696e63657469;
22   local_18 = 0;
23   iVar1 = strcmp(pcParm1,(char *)&local_28);
24   if (iVar1 == 0) {
25     puts("Congratulations,you are hacker\n");
26   }
27   else {
28     puts("Nope, wrong password....HIINTS:Do u love polimi?\n");
29   }
30   if (local_10 != *(long *)(in_FS_OFFSET + 0x28)) {
31                     /* WARNING: Subroutine does not return */
32     __stack_chk_fail();
33   }
34   return;
35 }
36
```

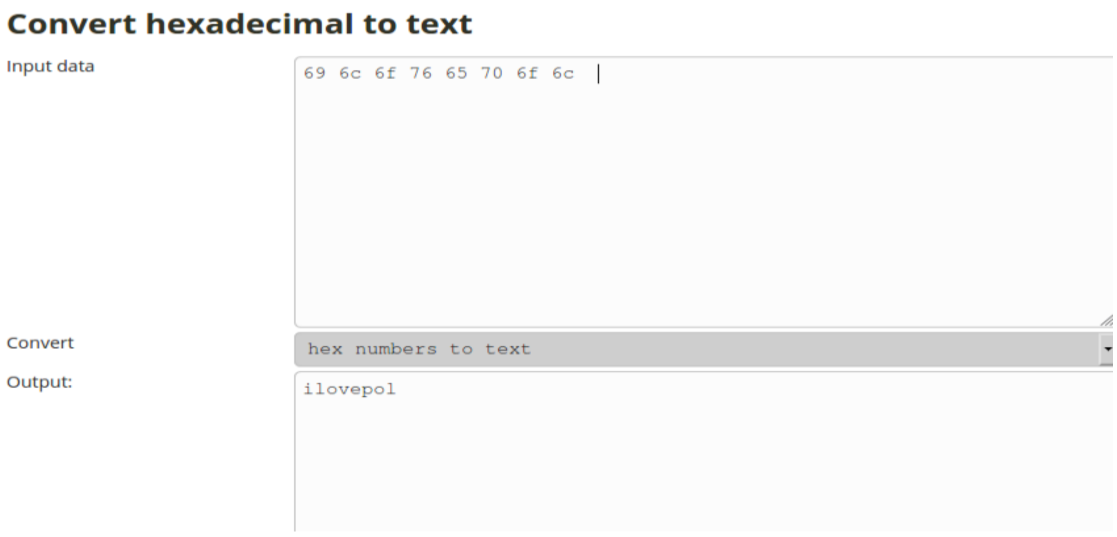
Here found interesting row in code (highlighted with green)

Also, we see on 23-th row there is compare function which give us 2 options: if data are matched then prints Congratulations, you are hacker, else Nope, wrong password.

Lets open hex convertor and convert highlighted row:

0x 6c 6f 70 65 76 6f 6c 69 we need change the order=>

69 6c 6f 76 65 70 6f 6c


Convert hexadecimal to text

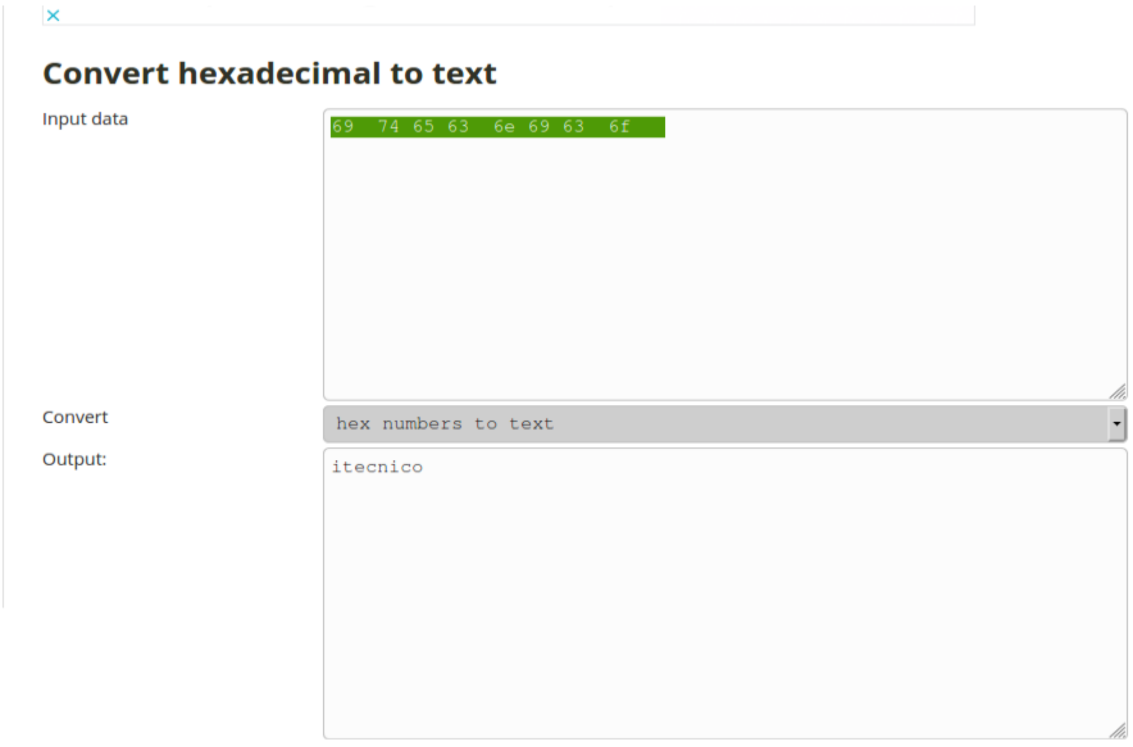
Input data

```
69 6c 6f 76 65 70 6f 6c
```

Convert: hex numbers to text

Output:

```
ilovepol
```


Convert hexadecimal to text

Input data

```
69 74 65 63 6e 69 63 6f
```

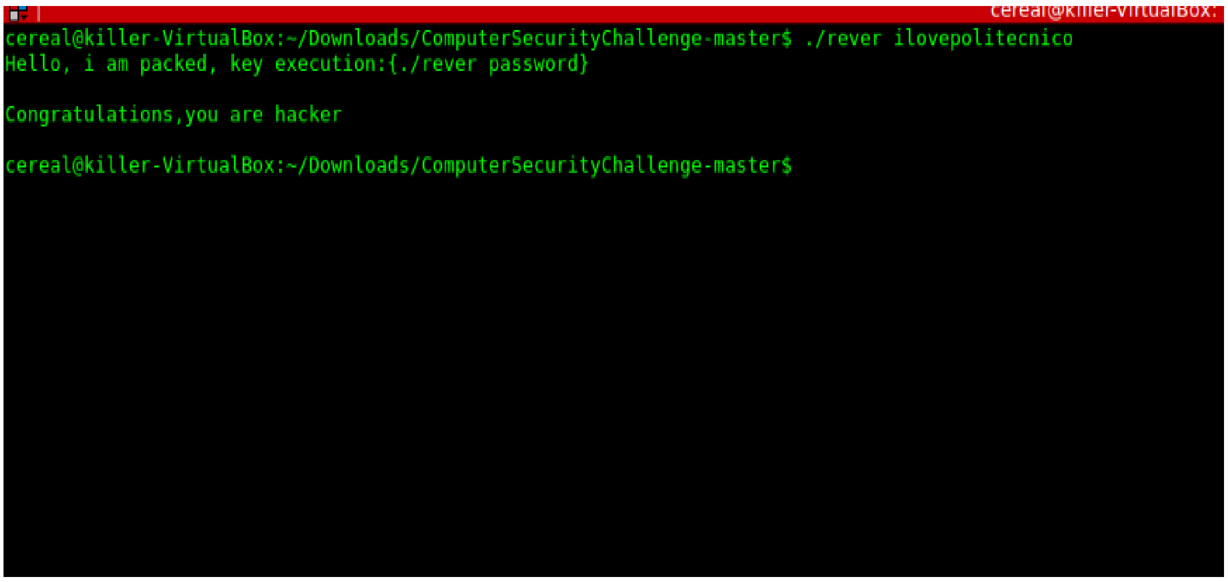
Convert: hex numbers to text

Output:

```
itecnico
```

Concatenating to results we got password: ilovepolitecnico

If we try, we solve challenge:

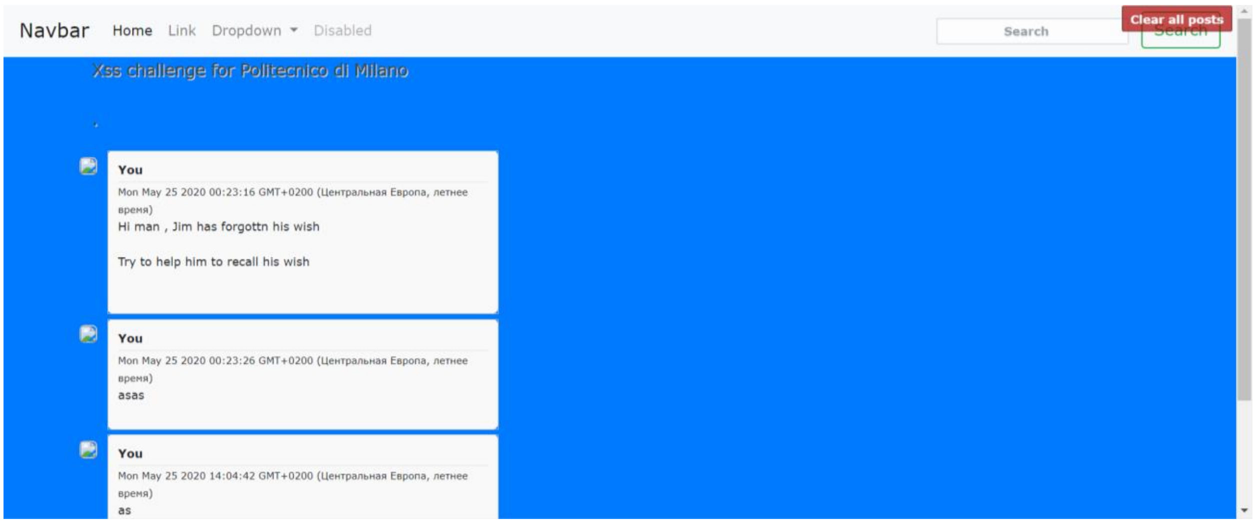


```
cereal@killer-VirtualBox:~/Downloads/ComputerSecurityChallenge-master$ ./rever ilovepolitecnico
Hello, i am packed, key execution:{./rever password}

Congratulations,you are hacker

cereal@killer-VirtualBox:~/Downloads/ComputerSecurityChallenge-master$
```

Challenge 2: XSS vulnerability

Cross-site scripting (XSS) is a type of computer security vulnerability typically found in web applications. XSS attacks enable attackers to inject client-side scripts into web pages viewed by other users.

I have created a vulnerable web page



Xss challenge for Politecnico di Milano

You
Mon May 25 2020 00:23:16 GMT+0200 (Центральная Европа, летнее время)
Hi man , Jim has forgottn his wish

Try to help him to recall his wish

You
Mon May 25 2020 00:23:26 GMT+0200 (Центральная Европа, летнее время)
asas

You
Mon May 25 2020 14:04:42 GMT+0200 (Центральная Европа, летнее время)
as

Here user allowed to post a text which will be stored on website part

User required to help to Jim. There is written that Jim has forgotten his wish.

Using xss vulnerable, user tries execute injected script.

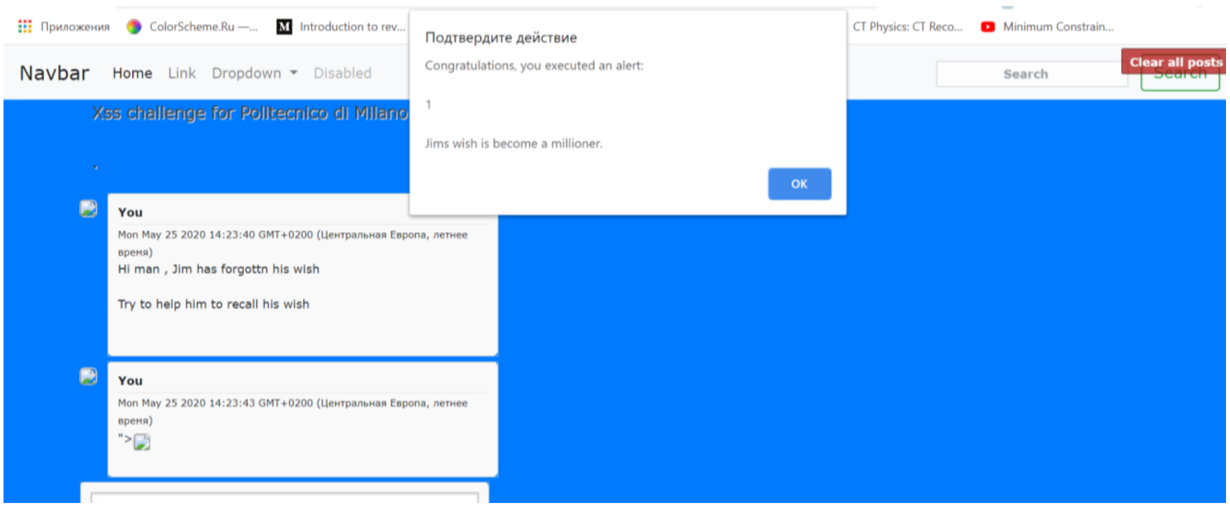
Firstly most common inection will fall :<script>alert(“xss”)</script> ,because there is sanitizer.

With this challenge attached source code: index.html

After observing the code, we will find exact attack:

`"><img src=x onerror=alert(1)>`

After executing we got exact flag:



Подтвердите действие

Congratulations, you executed an alert:

1

Jims wish is become a millioner.

You

Mon May 25 2020 14:23:40 GMT+0200 (Центральная Европа, летнее время)

Hi man , Jim has forgottn his wish

Try to help him to recall his wish

You

Mon May 25 2020 14:23:43 GMT+0200 (Центральная Европа, летнее время)

">

Answer is: Jim's wish is become a millioner

Lets have a look the script:

"><img src=x onerror=alert(1)> : at first quotes and closing tags which close previous tag and going out from sanitizer function after script says: "render image from src=x if there is error then alert 1. Overall, we execute alert script without common <script></script> syntax.