Lybryant Wright

CUNY John Jay College

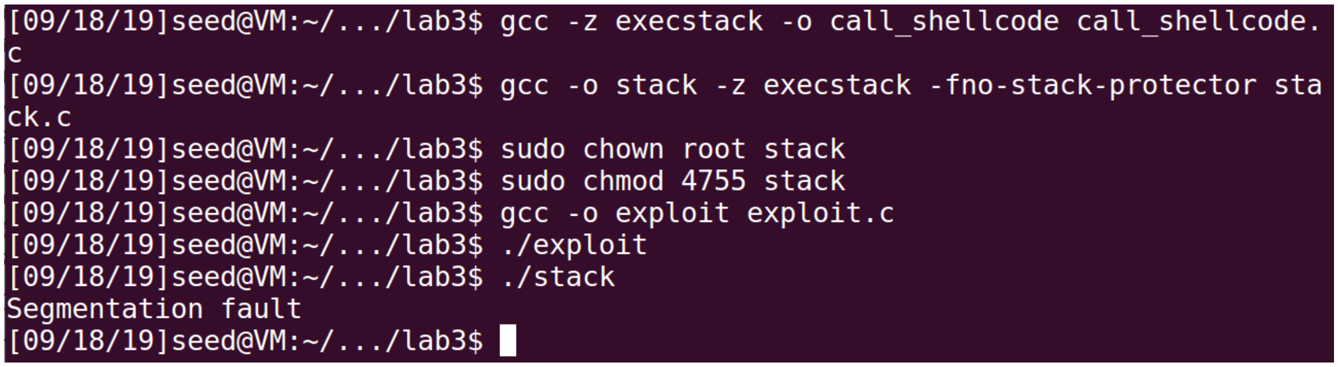
LAB 3

Buffer Overflow Vulnerability Lab

This lab is designed to teach us about the structure of the stack, shellcode and how we can exploit programs allowing us to exceed the boundaries of buffers giving us root privileges on a system. This type of vulnerability is called a buffer overflow. Operating systems such as Windows, MacOS, and various Linux operating systems have built in countermeasures and protocols to make buffer overflows difficult to achieve. Unfortunately in my attempts to perform a buffer overflow I could not achieve root privileges successfully.

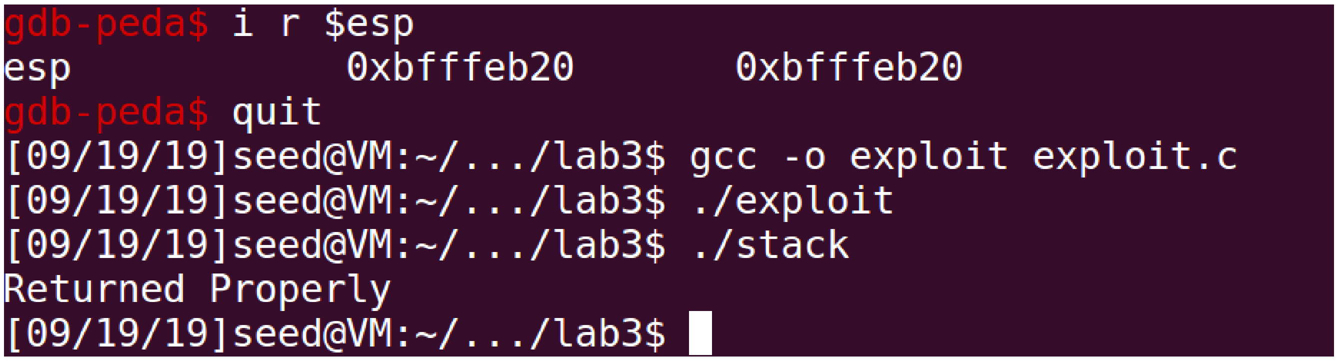
First step in attempting this lab is to disable certain countermeasures that would make it difficult to perform this attack. Disabling address space randomization and linking /bin/zsh to /bin/sh are essential measures that need to be addressed. Disabling address space randomization allows the user to guess the exact location of the heap and stack while you attach /bin/zsh to /bin/sh because it doesn't have any countermeasures that prevent buffer overflows. Another element to performing this attack is disable stackeguard protection and allow your stack to be executable. Using -fno-stack-protector will disable stackguard and adding execstack to the CLI command will allow the stack to be executable.

My first attempt at this attack continuously resulted in segmentation faults.



```
[09/18/19]seed@VM:~/.../lab3$ gcc -z execstack -o call_shellcode call_shellcode.
c
[09/18/19]seed@VM:~/.../lab3$ gcc -o stack -z execstack -fno-stack-protector sta
ck.c
[09/18/19]seed@VM:~/.../lab3$ sudo chown root stack
[09/18/19]seed@VM:~/.../lab3$ sudo chmod 4755 stack
[09/18/19]seed@VM:~/.../lab3$ gcc -o exploit exploit.c
[09/18/19]seed@VM:~/.../lab3$ ./exploit
[09/18/19]seed@VM:~/.../lab3$ ./stack
Segmentation fault
[09/18/19]seed@VM:~/.../lab3$ 
```

After further research and examining the instructions closer, I realized I had to alter the exploit.c code to fill the buffer for the badfile. I have to perform a strcpy function to add a bunch of 90's to the beginning of the stack to achieve root privileges.



```
gdb-peda$ i r $esp
esp            0xbfffeb20       0xbfffeb20
gdb-peda$ quit
[09/19/19]seed@VM:~/.../lab3$ gcc -o exploit exploit.c
[09/19/19]seed@VM:~/.../lab3$ ./exploit
[09/19/19]seed@VM:~/.../lab3$ ./stack
Returned Properly
[09/19/19]seed@VM:~/.../lab3$ 
```

The value in need for this is the value sitting in the $esp register. From my research this is where the str begins so I had to add this address into the function:

strcpy(buffer,"\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x20\xeb\xff\xbf");

However with this address, instead of segmentation faults I started getting “returned properly” and not achieving root privileges. Even with countermeasures disabled I couldn't find a solution to execute this task successfully.