**LAB 1 REPORT**

**Course:** 633

**Section:** 6

**Group Members:**

Alishba Aamir

Ivan Golovine

Fangbo Ren

**BUF\_SIZE = 200**

**TASK 1: Running Shellcode Explanation**

After compiling and running the call\_shellcode file, we noticed that we were able to launch a normal shell. Within this shell, we can perform functions like listing all the items in the directory or making new files/folders as shown below.

Text

Description automatically generated

**TASK 2: Exploiting the Vulnerability Explanation**

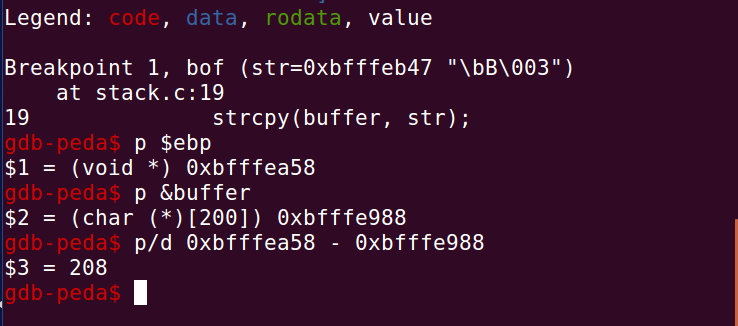
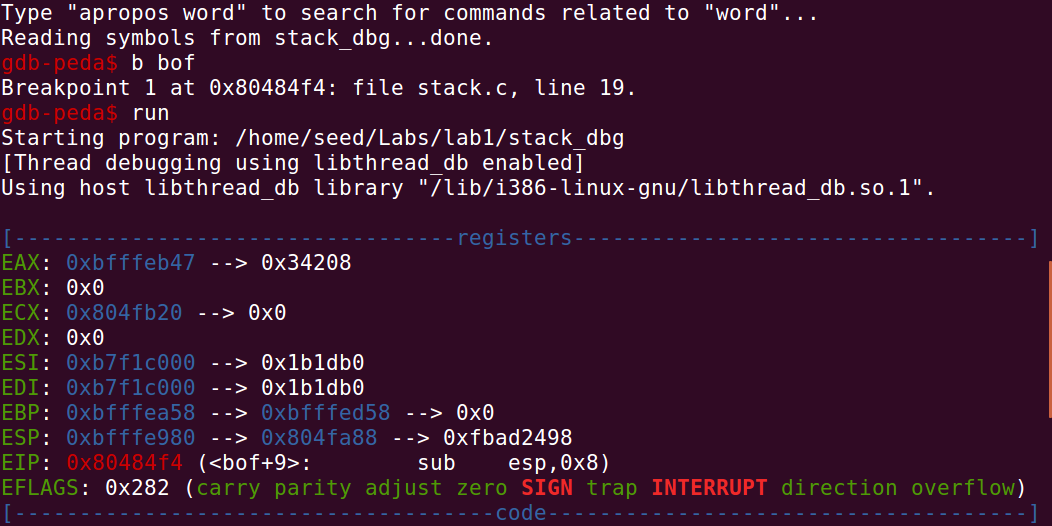
To exploit the vulnerability, the first step is to find the return address. We did this by using a debugging tool (gdb) as shown below. We ran gdb using the statement:

gcc -z execstack -fno-stack-protector -g -o stack\_gdb stack.c

Text

Description automatically generated

To find the return address, we put a breakpoint on the bof function and then ran the program. This allowed us to stop inside the bof function where we were able to get the value of the frame pointer address and the address of the buffer. We calculated the distance between the ebp and the buffer’s starting address. We then added 4 bytes to the result since the return address is stored above the ebp.



**ebp address:** 0xbfffea58

**Buffer address:** 0xbfffe988

**Offset start :** 208 + 4 = 212

**Offset end :** 212 + 4 = 216

**Return Address:** 0xbfffea58 +250

According to the gdb result, the return

address field starts from offset 212 and ends at offset 216. We calculated the return address as

0xbfffea58 +250 and added a larger value because when using the gdb tool, some additional data may have been pushed on the stack. First we tried using 0xbfffea58 + 120. When that didn’t work, we chose a larger number and tried 0xbfffea58 +250. This value worked!

Break point on bof function

Running the program

Find the ebp’s address

Find the buffers’s address

Find the offset

**Compilation**

We compiled the program and turned off the StackGuard and the non-executable stack protections. We then made the program a root-owned Set-UID program as shown below. Since we are working with Ubuntul6.04 VM, we only get the normal shell and not the root shell because of the countermeasures implemented in this version of Ubuntu. To fix this issue we used: **sudo ln - sf /bin/zsh /bin/sh** command to link /bin/sh to another shell that does not have that countermeasure. After filling out and compiling the exploit.py file, we executed stack.c (the vulnerable program). We were successfully able to exploit the vulnerable program and obtain root privilege.

Text

Description automatically generated