Assignment 1 - Buffer Overflow Vulnerability Assignment

Maverick Lee - 20312110
Wien Leung - 20038651

2.4 - Task 1: Exploiting the Vulnerability

In order to exploit the vulnerability, we first disable address-space-layout randomisation. This allows us to be able to set the return address to a location where we know our shellcode is. In addition, we set the default shell to one that wouldn't drop privileges when invoked, so that the shell we launch with the exploit will retain root privileges.

To exploit the vulnerable program, we first design a bad file to be read by it. We initialised a large byte array, and initialised it to NOP instructions. The first 32 bytes is used to store the desired return address, repeated eight times, which gives us a margin of error for overwriting the return address.

We figured out the desired return address of the frame pointer using gdb, we filled out the beginning portion of the buffer using this address plus an offset of 300 bytes. The shellcode was inserted into buffer[400], which also allows for more margin for error for the return address, to ensure the shellcode will be run.

2.5 - Task 2: Protection in /bin/bash

When running this after linking sh to bash, the stack program still runs successfully and the shell is started, but there are no sudo privileges, since bash is written to drop privileges. To circumvent this, we could use the -p option with bash, or set the name parameter of *execve* to "sh".

2.6 - Task 3: Address Randomization

Running ./stack once results in a segmentation fault. However, running
`sh -c "while [ 1 ]; do ./stack; done;"`
eventually leads to shell access, since the address will eventually be randomised to one that will allow our shellcode to be run.

2.7 - Task 4: Stack Guard

After compiling without the -fno-stack-protector, this error was observed after running ./stack:

```
seed@seed-desktop:~/ece458-master/a1$ ./stack
*** stack smashing detected ***: ./stack terminated
======= Backtrace: =========
/lib/tls/i686/cmov/libc.so.6(__fortify_fail+0x48)[0xb7f6cda8]
/lib/tls/i686/cmov/libc.so.6(__fortify_fail+0x0)[0xb7f6cd60]
```

```
./stack[0x8048513]
[0xbffff3f8]
[0x90909090]
======= Memory map: ========
08048000-08049000 r-xp 00000000 08:01 8367
/home/seed/ece458-master/a1/stack
08049000-0804a000 r--p 00000000 08:01 8367
/home/seed/ece458-master/a1/stack
0804a000-0804b000 rw-p 00001000 08:01 8367
/home/seed/ece458-master/a1/stack
0804b000-0806c000 rw-p 0804b000 00:00 0          [heap]
b7e52000-b7e5f000 r-xp 00000000 08:01 278049     /lib/libgcc_s.so.1
b7e5f000-b7e60000 r--p 0000c000 08:01 278049     /lib/libgcc_s.so.1
b7e60000-b7e61000 rw-p 0000d000 08:01 278049     /lib/libgcc_s.so.1
b7e6e000-b7e6f000 rw-p b7e6e000 00:00 0
b7e6f000-b7fcb000 r-xp 00000000 08:01 295506
/lib/tls/i686/cmov/libc-2.9.so
b7fcb000-b7fcc000 ---p 0015c000 08:01 295506
/lib/tls/i686/cmov/libc-2.9.so
b7fcc000-b7fce000 r--p 0015c000 08:01 295506
/lib/tls/i686/cmov/libc-2.9.so
b7fce000-b7fcf000 rw-p 0015e000 08:01 295506
/lib/tls/i686/cmov/libc-2.9.so
b7fcf000-b7fd2000 rw-p b7fcf000 00:00 0
b7fde000-b7fe1000 rw-p b7fde000 00:00 0
b7fe1000-b7fe2000 r-xp b7fe1000 00:00 0          [vdso]
b7fe2000-b7ffe000 r-xp 00000000 08:01 278007     /lib/ld-2.9.so
b7ffe000-b7fff000 r--p 0001b000 08:01 278007     /lib/ld-2.9.so
b7fff000-b8000000 rw-p 0001c000 08:01 278007     /lib/ld-2.9.so
bffeb000-c0000000 rw-p bffeb000 00:00 0          [stack]
Aborted
```