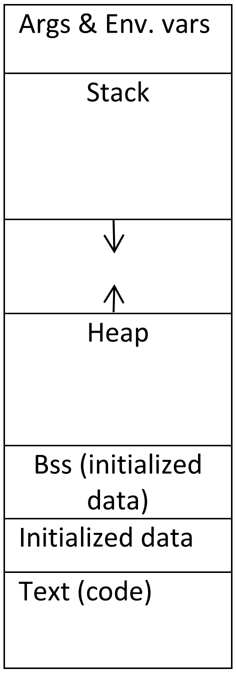
Nikhil Khatu
2/22/2013
CSC574- Assignment#3

Buffer Overflow Vulnerability Lab

Task# 1- Exploiting the Vulnerability

Here we begin the lab by understanding the dynamics of how data is handled by the system during run time. I spent about 8 to 10 hours going through a comprehensive buffer overflow tutorial that started off with the fundamentals of the compilation, assembly, debugging, and runtime environments. In short the fundamental structure is elaborated in the following depiction.

Linux Virtual Memory:



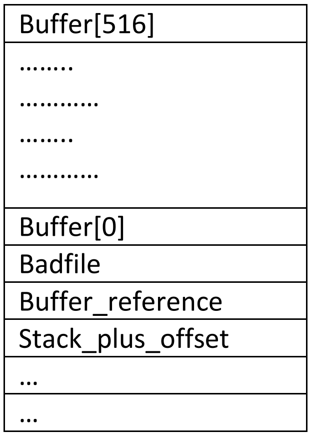
After being able to bring up a shell with call_shellcode.c we can begin to exploit the buffer overflow vulnerability. Since the environment used is the Virtual Machine provided by SEED lab, we execute with the following:

```
su root
sysctl -w kernel.randomize_va_space=0

su root
gcc -o stack -z execstack -fno-stack-protector stack.c
chmod 4755 stack
exit

gcc -o exploit exploit.c
./exploit
./stack
```

Stack from exploit.c

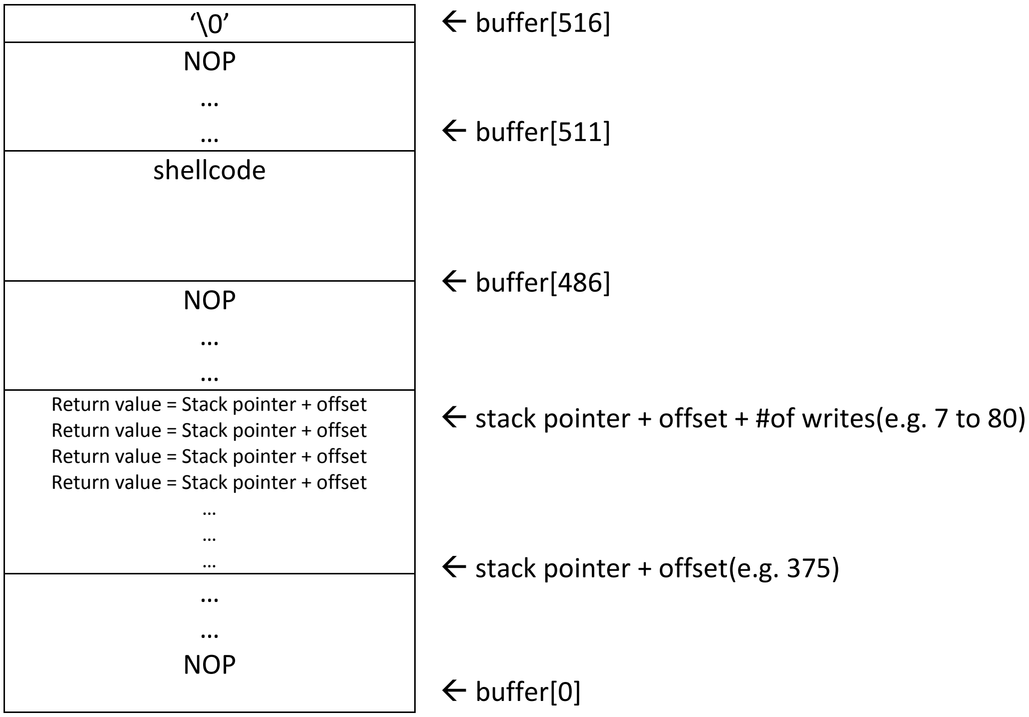


Buffer[516]
……..
…………
……..
…………
Buffer[0]
Badfile
Buffer_reference
Stack_plus_offset
…
…

Allocated to the stack → 0x080484d1 <+6>: sub $0x240,%esp
= 576 bytes

The key to success in exploiting this vulnerability is correctly identifying the return address and writing the return addresses into 'badfile'.

Contents of Badfile:



'\0' ← buffer[516]
NOP
…
… ← buffer[511]
shellcode
← buffer[486]
NOP
…
…
Return value = Stack pointer + offset ← stack pointer + offset + #of writes(e.g. 7 to 80)
Return value = Stack pointer + offset
Return value = Stack pointer + offset
Return value = Stack pointer + offset
…
…
… ← stack pointer + offset(e.g. 375)
…
…
NOP
← buffer[0]

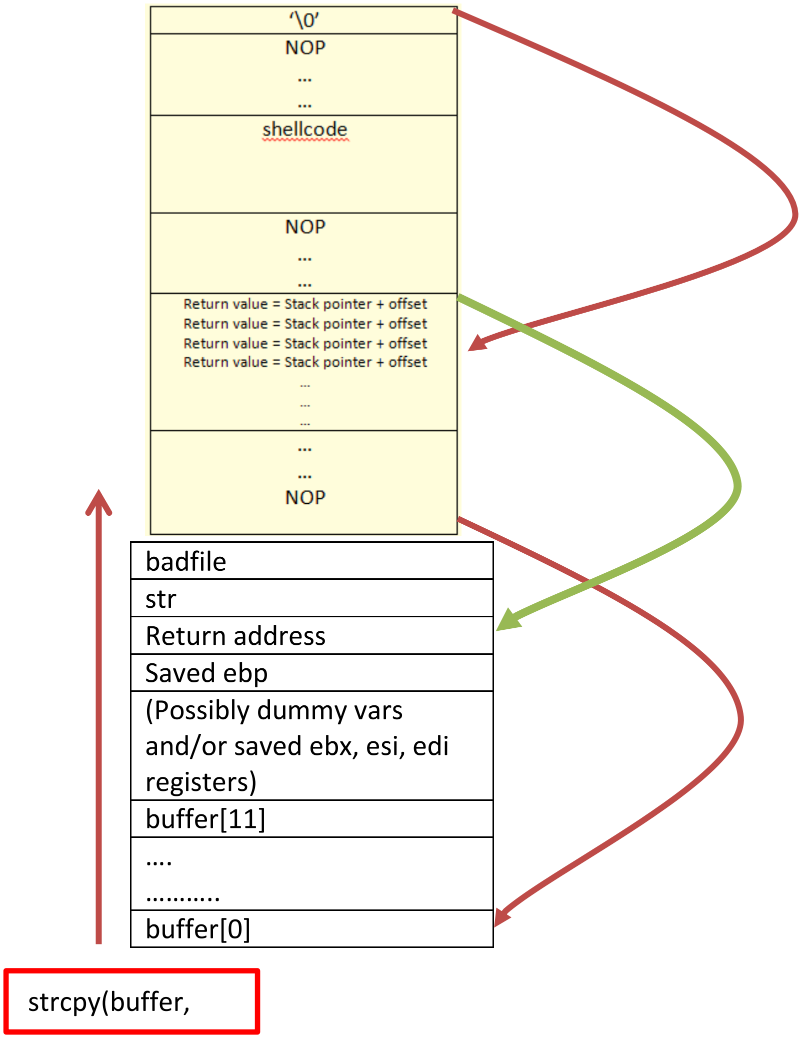
After editing exploit.c to create the badfile we are ready to hijack the vulnerable stack.c program:

Hijacking stack.c:

The following depicts the allocated stack during compilation.

str[516]
… …… …. ……… ….
str[0]
badfile
str
Return address
Saved ebp
(Possibly dummy vars and/or saved ebx, esi, edi registers)
buffer[11]
…. ………..
buffer[0]

During runtime the str array is loaded with the badfile. In turn this array is “copied” using ‘strcpy’ into the much smaller buffer array. Since the buffer is much smaller everything above it including the return address is overwritten.


‘\0’
NOP
…
…
shellcode
NOP
…
…
Return value = Stack pointer + offset
Return value = Stack pointer + offset
Return value = Stack pointer + offset
Return value = Stack pointer + offset
…
…
…
…
…
NOP
badfile
str
Return address
Saved ebp
(Possibly dummy vars and/or saved ebx, esi, edi registers)
buffer[11]
….
……….
buffer[0]
strcpy(buffer,

Allocated to stack of stack.c → 0x08048489 <+6>: sub $0x220,%esp
= 544 bytes

Task# 2- Protection in /bin/bash

NOTE: Skipping this section since the Ubuntu 11.04 VM is used. The VM uses ‘dash’)

Task# 3- Address Randomization

```
seed@ubuntu:~$ su root
Password:
root@ubuntu:/home/seed# sysctl -w kernel.randomize_va_space=2
kernel.randomize_va_space = 2
root@ubuntu:/home/seed# exit
exit
seed@ubuntu:~/Desktop$ gcc -o exploit exploit.c
seed@ubuntu:~/Desktop$ ./exploit
seed@ubuntu:~/Desktop$ ./stack
Segmentation fault
seed@ubuntu:~/Desktop$ su root
Password:
root@ubuntu:/home/seed/Desktop# sysctl -w kernel.randomize_va_space=0
kernel.randomize_va_space = 0
root@ubuntu:/home/seed/Desktop# exit
exit
seed@ubuntu:~/Desktop$ gcc -o exploit exploit.c
seed@ubuntu:~/Desktop$ ./exploit
seed@ubuntu:~/Desktop$ ./stack
# exit
seed@ubuntu:~/Desktop$
sh -c "while [ 1 ]; do ./stack; done;"
```

After turning the Address Randomization feature on I was unsuccessful in getting access to bash. This is theoretically possible by optimizing the badfile.

Task# 4- Stack Guard

```
seed@ubuntu:~/Desktop$ su root
Password:
root@ubuntu:/home/seed/Desktop# gcc -o stack -z execstack stack.c
root@ubuntu:/home/seed/Desktop# chmod 4755 stack
root@ubuntu:/home/seed/Desktop# exit
exit
seed@ubuntu:~/Desktop$ gcc -o exploit exploit.c
seed@ubuntu:~/Desktop$ ./stack
*** stack smashing detected ***: ./stack terminated
======= Backtrace: =========
/lib/i386-linux-gnu/libc.so.6(__fortify_fail+0x50)[0x225df0]
/lib/i386-linux-gnu/libc.so.6(+0xe5d9a)[0x225d9a]
./stack[0x80484f3]
[0xbffff2b0]
[0x90909090]
======= Memory map: ========
00110000-0012c000 r-xp 00000000 08:01 263123     /lib/i386-linux-gnu/ld-2.13.so
0012c000-0012d000 r-xp 0001b000 08:01 263123     /lib/i386-linux-gnu/ld-2.13.so
0012d000-0012e000 rwxp 0001c000 08:01 263123     /lib/i386-linux-gnu/ld-2.13.so
0012e000-0012f000 r-xp 00000000 00:00 0          [vdso]
0012f000-00132000 rwxp 00000000 00:00 0
00140000-0029a000 r-xp 00000000 08:01 263136     /lib/i386-linux-gnu/libc-2.13.so
0029a000-0029b000 ---p 0015a000 08:01 263136     /lib/i386-linux-gnu/libc-2.13.so
0029b000-0029d000 r-xp 0015a000 08:01 263136     /lib/i386-linux-gnu/libc-2.13.so
0029d000-0029e000 rwxp 0015c000 08:01 263136     /lib/i386-linux-gnu/libc-2.13.so
0029e000-002a2000 rwxp 00000000 00:00 0
002b1000-002cb000 r-xp 00000000 08:01 263164     /lib/i386-linux-gnu/libgcc_s.so.1
002cb000-002cc000 r-xp 00019000 08:01 263164     /lib/i386-linux-gnu/libgcc_s.so.1
002cc000-002cd000 rwxp 0001a000 08:01 263164     /lib/i386-linux-gnu/libgcc_s.so.1
08048000-08049000 r-xp 00000000 08:01 1180443    /home/seed/Desktop/stack
08049000-0804a000 r-xp 00000000 08:01 1180443    /home/seed/Desktop/stack
0804a000-0804b000 rwxp 00001000 08:01 1180443    /home/seed/Desktop/stack
0804b000-0806c000 rwxp 00000000 00:00 0          [heap]
bffdf000-c0000000 rwxp 00000000 00:00 0          [stack]
Aborted
```

Task# 5- Non-executable Stack

```
root@ubuntu:/home/seed/Desktop# gcc -o stack -z noexecstack -fno-stack-protector stack.c
root@ubuntu:/home/seed/Desktop# chmod 4755 stack
root@ubuntu:/home/seed/Desktop# exit
exit
seed@ubuntu:~/Desktop$ gcc -o exploit exploit.c
seed@ubuntu:~/Desktop$ ./exploit
seed@ubuntu:~/Desktop$ ./stack
Segmentation fault
```