```
[ebp+arg 0], esi
```

DEP & ROP

```
Modern Binary Exploitation
 CSCI 4968 - Spring 2015
```

Markus Gaasedelen

Lecture Overview

- 1. Introducing DEP
- 2. The History of DEP
- 3. Bypassing DEP with ROP
- 4. Stack Pivoting

Class up until Now

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

- Reverse Engineering
- Basic memory corruption
- Shellcoding
- Format strings
- Classical exploitation, few protections, pretty eZ
- Time to add some 'modern' to the binary exploitation madness



oc 31307D:

; CODE XREF: sub 312FD

and eax, Offffh or eax, 8007000

3

Modern Exploit Mitigations

 Theres a number of modern exploit mitigations that we've generally been turning off for the labs and exercises

```
DEP
```

- ASLR
- Stack Canaries
- ...?

Modern Exploit Mitigations

 Theres a number of modern exploit mitigations that we've generally been turning off for the labs and exercises

```
DEP
```

- ASLR
- Stack Canaries
- ...?

```
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
```

- Today we turn one back on for the remainder of the course
 - no more silly -z execstack in our gcc commands

```
lecture@warzone:/levels/lecture/rop$
lecture@warzone:/levels/lecture/rop$ checksec --file ./rop_exit

RELRO STACK CANARY NX PIE RPATH RUNPATH FILE

Partial RELRO No canary found NX enabled No PIE No RPATH No RUNPATH ./rop_exit

lecture@warzone:/levels/lecture/rop$
```

and eax, OFFFFh
or eax, 80070000h

5
; CODE XREF: sub

Course Terminology

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

- Data Execution Prevention
 - An exploit mitigation technique used to ensure that only code segments are ever marked as executable
 - Meant to mitigate code injection / shellcode payloads
 - Also known as DEP, NX, XN, XD, W^X

```
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+56
push oDh
call sub_3141B

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
; sub_312FD8
; sub_312FD8
; sub_312FD8
; call sub_3140F3
and eax, 0FFFFh
or eax, 800700000h
```

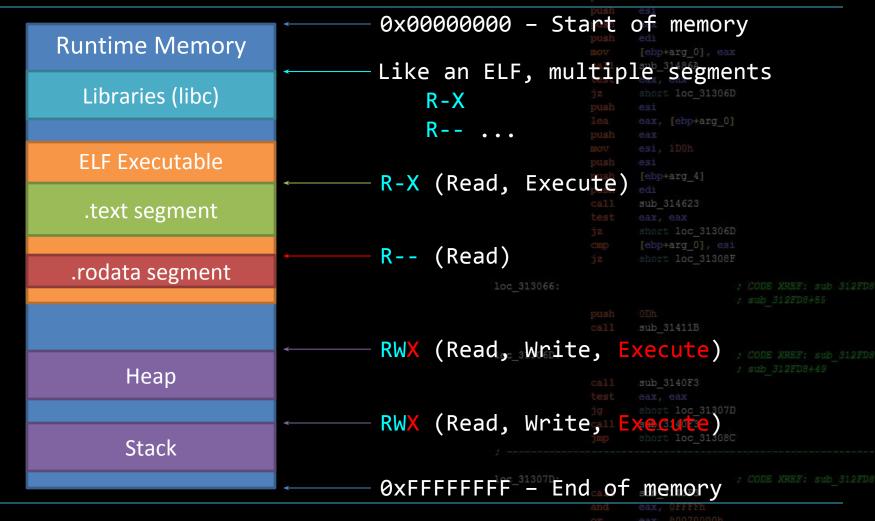
Runtime Process Without DEP



| Runtime Memory | 0x00000000 - Start of memory |
|------------------|------------------------------------------------------------------------------|
| Mantine Wiemory | Like an ELF, multiple segments |
| Libraries (libc) | R-X jz short loc_31306D push esi |
| | R [ebp+arg_0] |
| ELF Executable | mov esi, 1D0h push esi |
| | R-X (Read, Execute) [ebp+arg_4] |
| .text segment | call sub_314623 test eax, eax |
| | R (Read) Short loc_31306D Cmp [ebp+arg_0], esi jz short loc_31306F |
| .rodata segment | |
| | loc_313066: ; CODE XREF: sub 312FD ; sub 312FD8+55 |
| | push ODh call sub_31411B |
| | RWX (Read, Write, Execute) |
| Неар | ; sub_312FD8+49 call sub_3140F3 |
| | DUV (Dood White Tyrocute) |
| Stack | RWX (Read, Write, Execute) |
| Stack | |
| | • Oxfffffff - End of memory |

Runtime Process Without DE





Runtime Process Without



| Runtime Memory | 0x00000000 - Start of memory |
|------------------|-----------------------------------------------------------------------------|
| Libraries (libc) | Like an ELF, multiple segments R-X push lea eax, [ebp+arg_0] |
| ELF Executable | R push eax mov esi, 1D0h push esi R-X (Read, Execute) [abp+arg_4] edi |
| .text segment | call sub_314623 test eax, eax jz short loc_31306D |
| .rodata segment | R (Read) 10c_313066: |
| | Push ODh call sub_31411B RW- (Read, White, Execute); code XREF: sub 312FD8 |
| Неар | ; sub_312FD8+49 call sub_3140F3 test eax, eax ig short log 31307D |
| Stack | RW- (Read, Write; Exactive) short loc 313080 |
| | Oxfffffff - End of memory |

Runtime Process With DEP

est eax, eax

short loc_31306D

[ebp+arg_0], ebx

short loc_313066

eax, [ebp+var_70

eax, [ebp+var_84

short loc_313066

eax, [ebp+var_84

sub

est

est

0x00000000 - Start of memory Runtime Memory Like an ELF, multiple segments Libraries (libc) R-X **ELF** Executable R-X (Read, Execute) .text segment R-- (Read) .rodata segment RW- (Read, Write) Heap RW- (Read, Write) Stack 0xFFFFFFFF = End of memory

DEP Basics

- No segment of memory should ever be Writable and Executable at the same time, 'W^X'
- Common data segments
 - Stack, Heap
 - .bss
 - .ro
 - .data
- Common code segments
 - .text
 - .plt

DEP in Action

0xbffdf000 ----> (lower addrs)

Stack

- Data should never be executable, only code
- What happens if we stack smash, inject shellcode, and try to jump onto the stack?

NOP Sled \x90 \x90 \x90 \x90

Shellcode

...\x90\x90\x90\x90

RET Overwrite

Previous Stack Frame

0xc0000000

(higher addrs) **DEP & ROP**

DEP in Action

0xbffdf000 ----> (lower addrs)

Stack

Data should never be executable, only code

MBE - 03/10/15

What happens if we stack smash, inject shellcode, and try to jump onto the stack?

NOP Sled \x90 \x90 \x90 \x90

Shellcode

... \x90 \x90 \x90 \x90

RET Overwrite

Previous Stack Frame

0xc0000000 (higher addrs)

DEP & ROP

DEP in Action

0xbffdf000 ---->
 (lower addrs)

Stack

NOP Sled

- Data should never be executable, only code
- What happens if we stack smash, inject shellcode, and try to jump onto the stack?

yay mitigation technologies!

\x90 \x90 \x90 \.

SEGFAULT at 0xbffffc04

Previous Stack Frame

0xc0000000 ---->
(higher addrs)

DEP & ROP

eax, 80070000

14

Lecture Overview

- 1. Introducing DEP
- 2. The History of DEP
- 3. Bypassing DEP with ROP
- 4. Stack Pivoting

When was DEP implemented?

```
16
```

- When was DEP implemented?
 - August 14th, 2004 Linux Kernel 2.6.8

- When was DEP implemented?
 - August 14th, 2004 Linux Kernel 2.6.8
 - August 25th, 2004 Windows XP SP2

```
18
```

- When was DEP implemented?
 - August 14th, 2004 Linux Kernel 2.6.8
 - August 25th, 2004 Windows XP SP2
 - June 26th, 2006 Mac OSX 10.5

- When was DEP implemented?
 - August 14th, 2004 Linux Kernel 2.6.8
 - August 25th, 2004 Windows XP SP2
 - June 26th, 2006 Mac OSX 10.5

```
about 10 years ago
```

MBE - 03/10/15 DEP & ROP

20 ; CODE XREF: sub_312FD:

2004 in Perspective

- Facebook is created
- G-Mail launches as beta
- Ken Jennings begins his 74 win streak on Jeopardy
- Halo 2 is released, as is Half Life 2
- LOST airs its first episode





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

FALO 2

3140F3

or eax, 0FFFFh

Security is Young

```
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 eax
```

 Technologies in modern exploit mitigations are incredibly young, and the field of computer security is rapidly evolving

 DEP is one of the of the main mitigation technologies you must bypass in modern

exploitation

MBE - 03/10/15 DEP & ROP

; CODE XREF: &

Lecture Overview

- 1. Introducing DEP
- 2. The History of DEP
- 3. Bypassing DEP with ROP
- 4. Stack Pivoting

```
23
```

Bypassing DEP

```
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
```

- DEP stops an attacker from easily executing injected shellcode assuming they gain control of EIP
 - shellcode almost always ends up in a RW- region

```
If you can't inject (shell)code to do your bidding, you must re-use the existing code!
```

This is technique is usually some form of ROP

MBE - 03/10/15 DEP & ROP

; CODE XREF: sub_312

Course Terminology

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

Return Oriented Programming

- A technique in exploitation to reuse existing code gadgets in a target binary as a method to bypass DEP
- Also known as ROP

push [ebp+arg_4] push edi call sub_314623 test eax, eax jz short loc_31306E cmp [ebp+arg_0], esi jz short loc_31308E

Gadget

- A sequence of meaningful instructions typically followed by a return instruction
- Usually multiple gadgets are chained together to compute malicious actions like shellcode does

DEP & ROP

These chains are called ROP Chains

or eax, 0ffffh
or eax, 80070000h
25

Relevant Quotes

```
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
```

"Preventing the introduction of malicious code is not enough to prevent the execution of malicious computations"

-Dino Dai Zovi

26

oc_31307D:

and eax, Offfth or eax, 80070000h

or eax, 80070000h

Gadgets

MBE - 03/10/15

- **ROP Chains** are made up of gadgets
- Example gadgets -

```
xor
       eax, eax
ret
```

```
ebx
pop
pop
       eax
ret
```

```
add
      eax, ebx
ret
```

\$ ropgadget --binary /bin/bas

```
| Short loc_31306D | Short loc_31306D | Short loc_31306C | Short loc_31306C | Sub eax, [ebp+var_84] | push esi | push eax | push edi
```

```
0x080d2262 : xor ebx, ebx ; mov esi, edi ; jmp 0x80d227d
0x080ac337 : xor ecx, dword ptr [ecx + 0x448b2404] ; and al, 0xc ; call eax
0x080d02b8 : xor ecx, ecx ; cmp dword ptr [edx], 0x2e ; je 0x80d02f1 ; mov eax, ecx ; ret
0x080cc175 : xor ecx, ecx ; mov eax, edx ; pop ebx ; mov edx, ecx ; pop esi ; pop edi ; ret
0x0808b728 : xor ecx, ecx ; xor edx, edx ; mov eax, esi ; call 0x8087958
0x080bc610 : xor edi, edi ; pop ebx ; mov eax, edi ; pop esi ; pop edi ; pop ebp ; ret
0x0812b059 : xor edi, edx ; jmp dword ptr [ebx]
0x0811a06d : xor edx, edi ; jmp dword ptr [eax]
0x080fcc4d : xor edx, edx ; add esp, 0x14 ; pop esi ; pop edi ; pop ebp ; ret
0x080fcb6c : xor edx, edx ; add esp, 0xc ; pop esi ; pop edi ; pop ebp ; ret
0x080a395b : xor edx, edx ; call 0x80a2879
0x080d6e71 : xor edx, edx ; cmp eax, 0x16 ; setne dl ; jmp 0x80d6e53
0x08072090 : xor edx, edx ; mov dword ptr [eax + 8], edx ; add esp, 0x18 ; pop ebx ; ret
0x0808b72a : xor edx, edx ; mov eax, esi ; call 0x8087956
0x080861bd : xor edx, edx ; pop ebx ; pop esi ; ret
0x08070246 : xor edx, edx ; pop esi ; pop edi ; pop ebp ; ret
0x08075a58 : xor edx, edx ; pop esi ; pop edi ; ret
0x080f8877 : xor esi, 0x89c085ff ; ret
0x080f3a88 : xrelease ; mov dword ptr [esp], esi ; call 0x80efd46
Unique gadgets found: 15840
lecture@warzone:/levels$
```

call sub_3140F3
jmp short loc_31308C
;
loc_31307D: ; CODE XREF: sub_312FD&
call sub_3140F3

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

28

```
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
```

It is almost always possible to create a logically equivalent
 ROP chain for a given piece of shellcode

exit(0) - shellcode

xor eax, eax

xor ebx, ebx

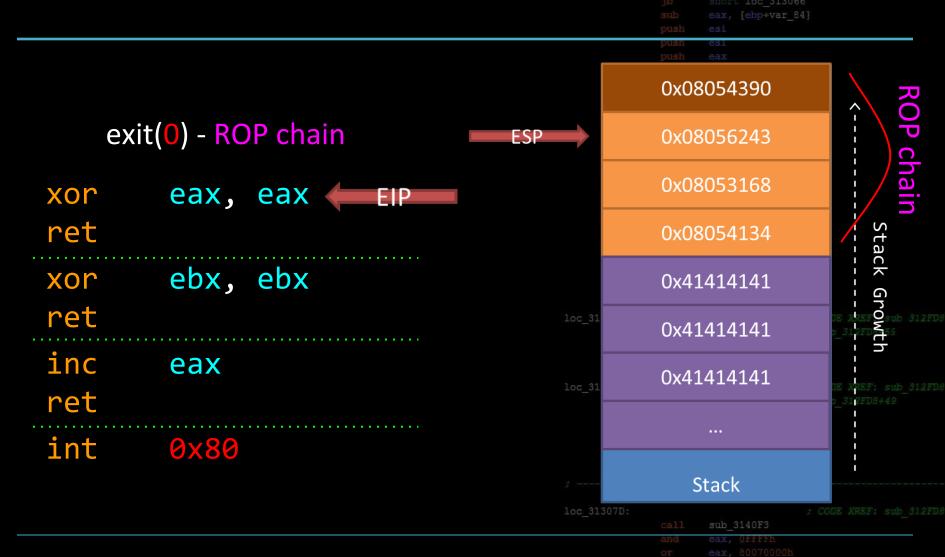
inc eax

int 0x80

```
exit(0) - ROP
xor
           eax,
ret
           ebx, ebx
xor
ret
inc
                     sub 3140F3
           eax
ret
                     sub 3140F3
int
        loc 30 X80
```

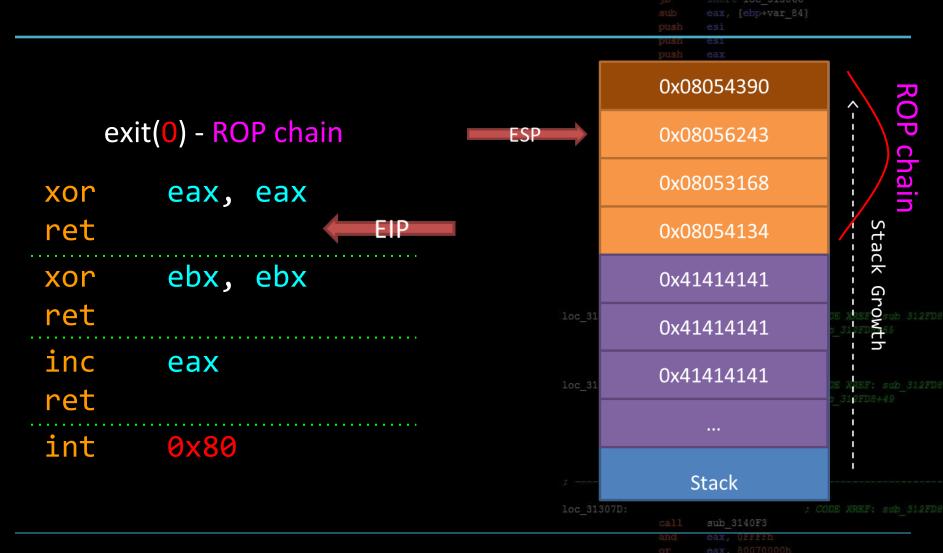


MBE - 03/10/15

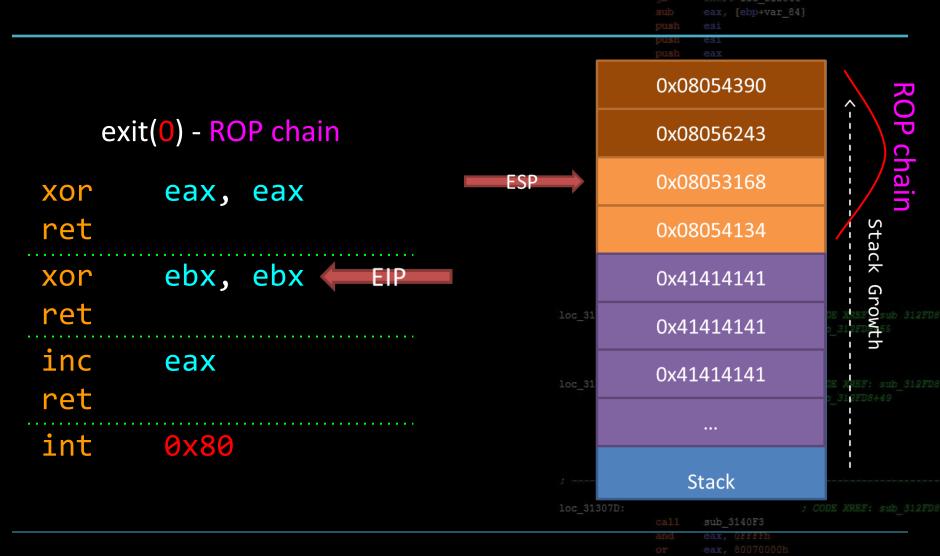


DEP & ROP

MBE - 03/10/15



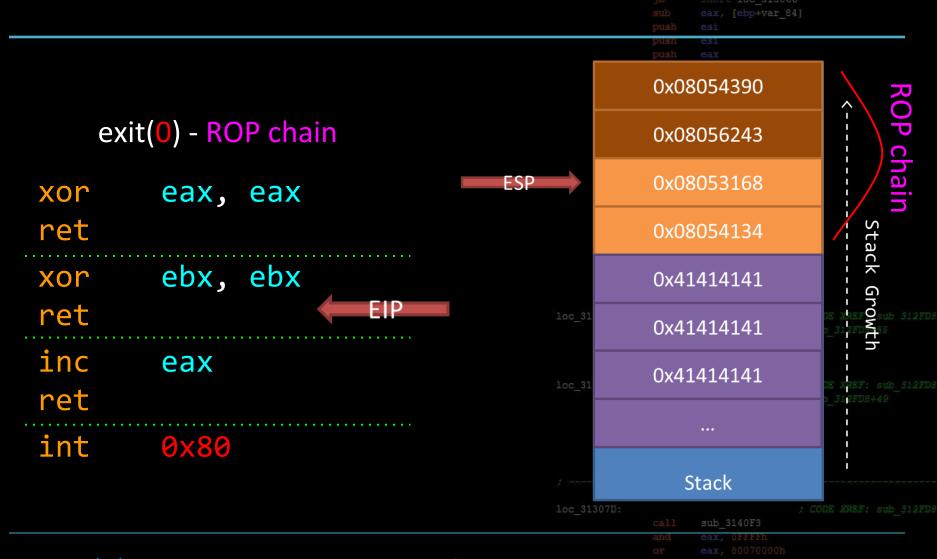
DEP & ROP



MBE - 03/10/15

DEP & ROP

10c_31308C: ; CODE XREF: sub_3:

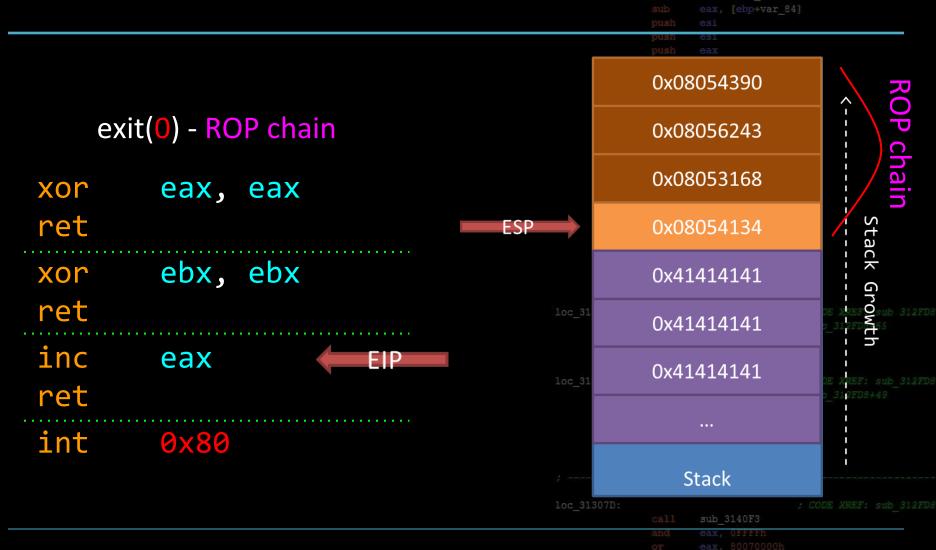


MBE - 03/10/15

DEP & ROP

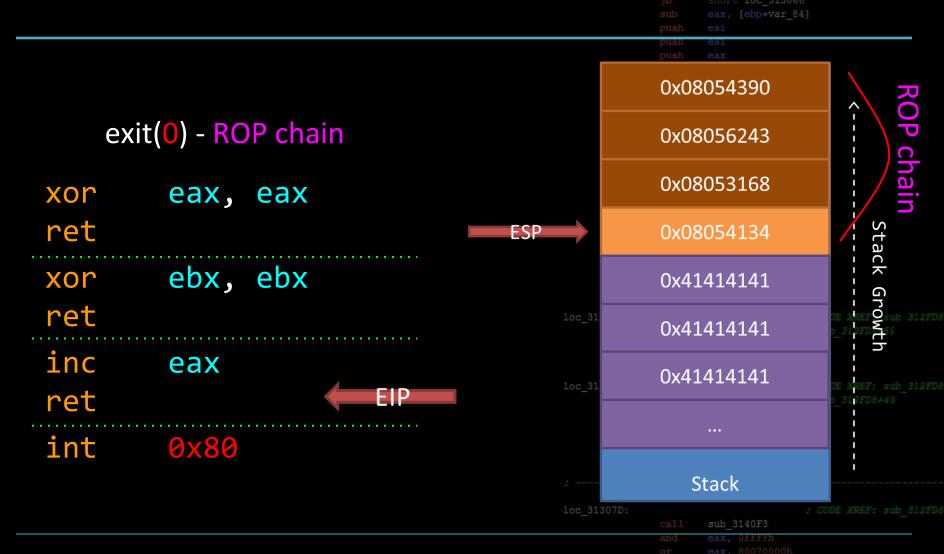
.oc_31308C:

hn-war 41 eav



MBE - 03/10/15 DEP & ROP 35
loc 31308C: ; CODE XREF: sub 31

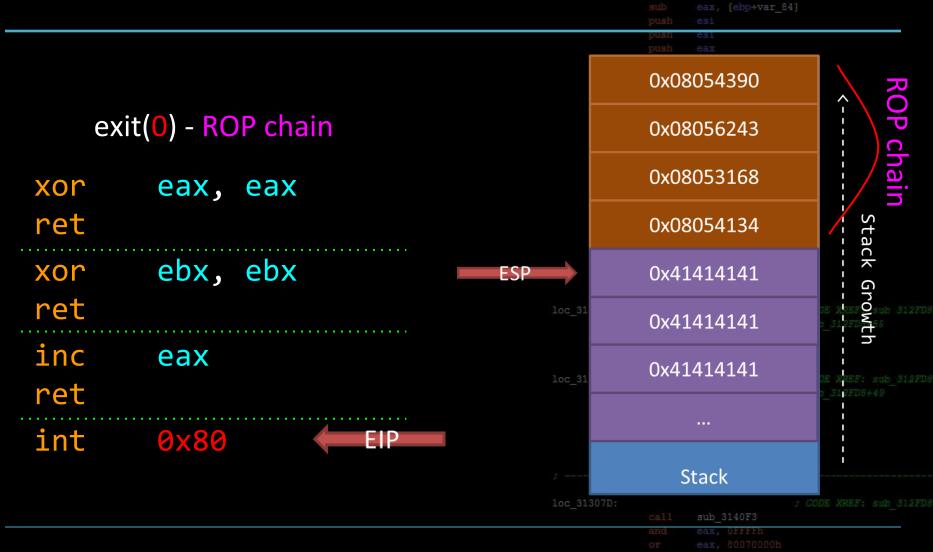
MBE - 03/10/15



DEP & ROP

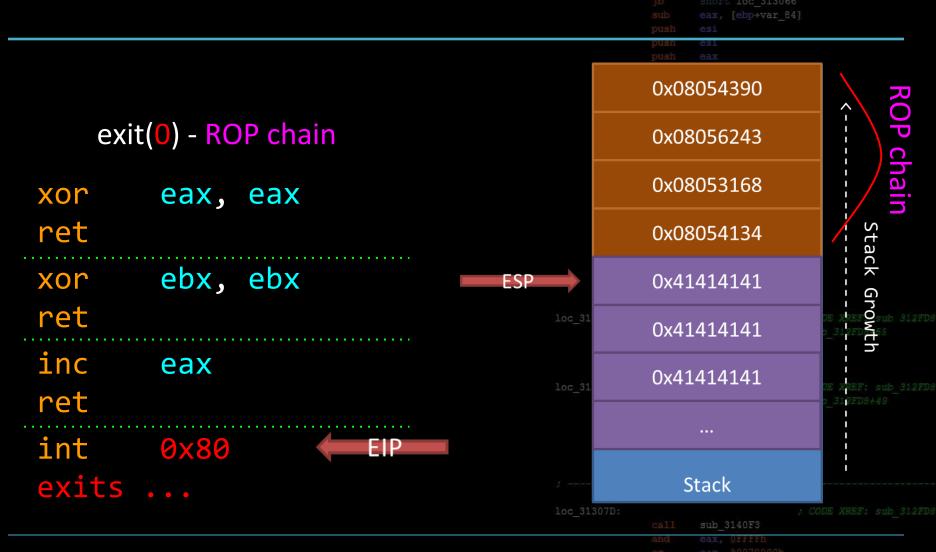
36

Understanding ROP



MBE - 03/10/15 DEP & ROP

Understanding ROP



MBE - 03/10/15

DEP & ROP

oc_31308C:

; CODE XREF: sub

Bypassing DEP with ROP

```
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 eax
```

- We called exit(0) without using any sort of shellcode!
- With that said, writing ROP can be difficult and you will usually have to get creative with what gadgets you find

MBE - 03/10/15 DEP & ROP

; CODE XREF; sub 312FD8

/levels/lecture/rop/rop_ex

- Play around with ROP on the warzone
- Can you make a ROP chain to set arbitrary exit values? 0? 200? 64?

DEP & ROP

MBE - 03/10/15

Relevant Tips/Tools/Commands

- \$ ropgadget --binary ./rop_exit > /tmp/gadgetzXYZ.txt
- \$ asm
 - easy way to get the bytes for gadgets you're looking for
- \$ gdbpeda
 - searchmem, find raw bytes in an executing program
 - ropsearch, a crappy rop gadget finder
- python def q(addr):

return struct.pack("I", addr)

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

and eax, 0ffffh
or eax, 80070000h

Lecture Overview

- 1. Introducing DEP
- 2. The History of DEP
- 3. Bypassing DEP with ROP
- 4. Stack Pivoting

Typical Constraints in ROP

cmp ean
jb sho
sub ean
push esi
push esi
push esi

- Typically in modern exploitation you might only get one targeted overwrite rather than a straight stack smash
- What can you do when you only have one gadget worth of execution?
 - Answer: Stack Pivoting

MBE - 03/10/15 DEP & ROP

[ebp+var 4], 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

You control the orange

You have one gadget before you drop into arbitrary data on the stack

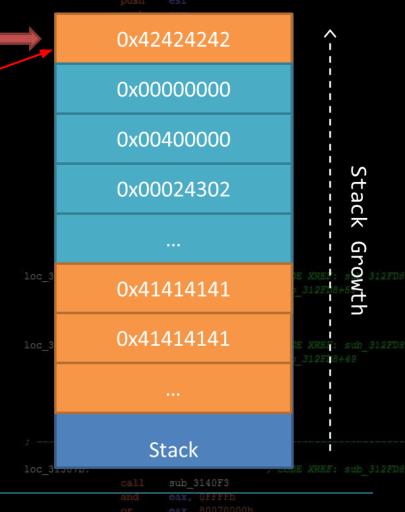


MBE - 03/10/15 DEP & ROP

ush edi
all sub_314623
est eax, eax
z short loc_31306D
mp [ebp+arg_0], ebx
nz short loc_313066
ov eax, [ebp+var_70]
mp eax, [ebp+var_84]
b short loc_313066
ub eax, [ebp+var_84]
ush esi

You control the orange

You have one gadget before you drop into arbitrary data on the stack



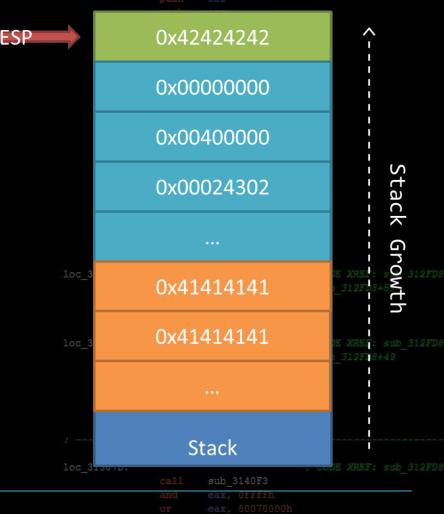
 ESP

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]

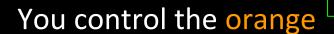
You control the orange

You have one gadget before you drop into arbitrary data on the stack

Use your one gadget to move ESP into a more favorable location (Stack Pivot)



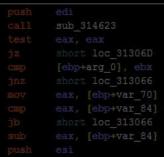


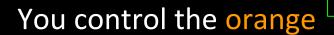


You have one gadget before you drop into arbitrary data on the stack

Use your one gadget to move ESP into a more favorable location (Stack Pivot)

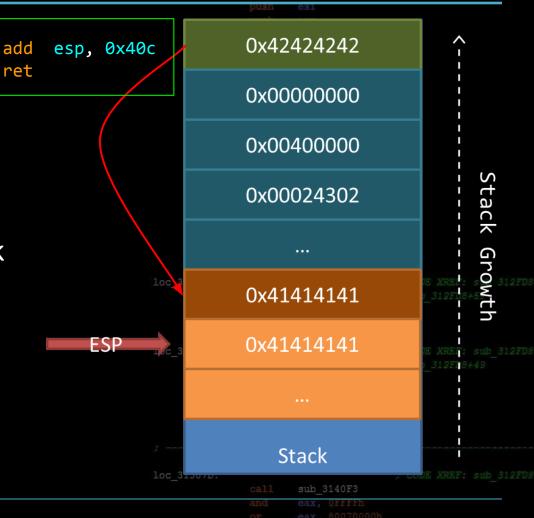






You have one gadget before you drop into arbitrary data on the stack

Use your one gadget to move ESP into a more favorable location (Stack Pivot)



```
Stack Pivoting Tips
      esp, 0xXXXX
add
ret
      esp, 0xXXXX
sub
                              any gadgets that touch esp
ret
                             will probably be of interest
                                 for a pivot scenario
ret 0xXXXX
leave ; (mov esp, ebp)
ret
xchg eXX, esp
ret
```

MBE - 03/10/15

DEP & ROP

c_31308C: ; CODE XREF: sub_312FD

Stack Pivoting Tips

```
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
```

 You may not find an exact pivot, or you may need to pivot multiple times!

```
test eax, eax
jz short loc_313061
push esi
lea eax, [ebp+arg_0]
push eax
mov esi, 1D0h
push esi
push [ebp+arg_4]
```

 You can always pad your ROP Chains with ROP NOPs which are simply gadgets that point to ret's

/levels/lecture/rop/rop_pivotion=133

Play around with Stack Pivoting on the warzone

MBE - 03/10/15 DEP & ROP

loc_31308C:

; CODE XREF: sub_312FD

ret2libc

```
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
```

• 'ret2libc' is a technique of ROP where you return to functions in standard libraries (libc), rather than using gadgets

If you know the addresses of the functions you want to ROP through in libc (assuming libc exists), ret2libc is easier than making a ROP

chain with gadgets

```
jmp short loc_31308C

; CODE XREF: sub_312FD

call sub_3140F3

and eax, 0FFFFh

or eax, 80070000h
```

Common ret2libc Targets

```
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
```

- system()
 - Executes something on the command line
 - system("cat flag.txt");
- (f) open() / read() / write()
 - Open/Read/Write a file contents

```
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
```

```
call sub_31411B

call sub_31411B

; CODE XREF: sub_312FD
; sub_312FD8+49

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

; CODE XREF: sub_312FD
```

```
push edi
call sub_314623
test eax, eax
jz short loc_313061
cmp [ebp+arg_0], ebp
jnz short loc_313066
eax, [ebp+var_84
jb short loc_313066
sub eax, [ebp+var_84
```

```
system() --->
```

0x08045430: ret

.....

system()

0xb7e65190: push ebx

0xb7e65191: sub esp, 8

0xb7e65194: mov eax, DWORD PTR

[esp+0x10]

• • •



Returning to System

```
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
```

- We want to call system("cat flag.txt"); [cbp+arg_0], eax test); sub_31486A eax, eax short loc 31306D
- Because we are ROPing into system rather than calling it, you have to think about setting up the stack (to pass arguments) a little bit differently

```
system()
```

0x08045430: ret

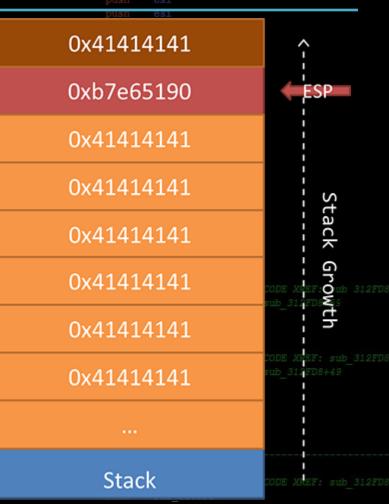
system()

0xb7e65190: push ebx

0xb7e65191: sub esp, 8

0xb7e65194: mov eax, DWORD PTR

[esp+0x10]



```
coush edi
call sub_314623
test eax, eax
jz short loc_31306D
[ebp+arg_0], ebx
jnz short loc_313066
eax, [ebp+var_70
eax, [ebp+var_84
jb short loc_313066
eax, [ebp+var_84
```

```
system()
                                 33333
                                 33353
0x08045430:
              ret
system()
0xb7e65190:
              push
                      ebx
                                  \mathsf{EIP}
0xb7e65191:
              sub
                      esp, 8
0xb7e65194:
              mov
                      eax, DWORD PTR
  [esp+0x10]
```

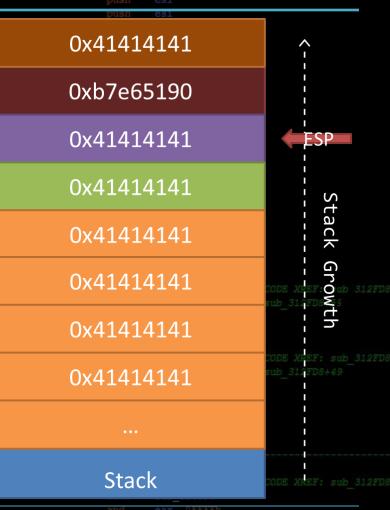


MBE - 03/13/15 DEP & ROP

; CODE XREF:

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

```
system()
                     ret address
                        first arg
0x08045430:
            ret
system()
0xb7e65190:
             push
                    ebx
0xb7e65191:
            sub
                   esp, 8
0xb7e65194:
            mov
                    eax, DWORD PTR
  [esp+0x10]
```



MBE - 03/13/15 DEP & ROP

push

sub

mov

```
DWORD PTR [esp+0x10]
```

system()

0xb7e65190:

0xb7e65191:

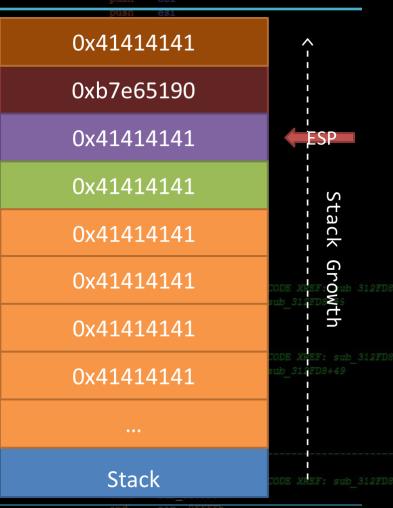
0xb7e65194:

ebx

esp,

eax,

```
system()
                     ret address
                        first arg
0x08045430:
            ret
system()
0xb7e65190:
             push
                    ebx
0xb7e65191:
            sub
                   esp, 8
0xb7e65194:
            mov
                    eax, DWORD PTR
  [esp+0x10]
```

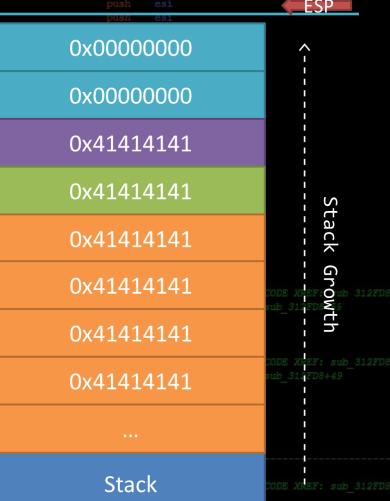


MBE - 03/13/15 **DEP & ROP** Growth

ESP

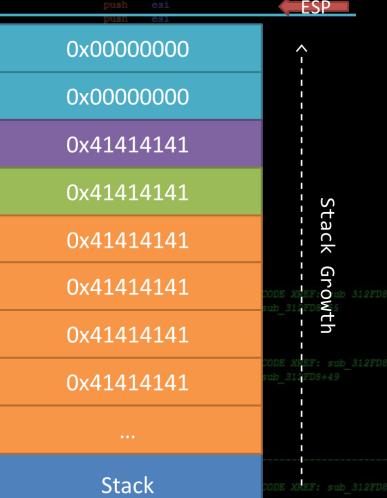
Growth

```
system()'s
                        stack frame
                    ret address
                       first arg
0x08045430:
            ret
system()
0xb7e65190:
            push
                   ebx
0xb7e65191:
            sub
                   esp, 8
0xb7e65194:
                   eax, DWORD PTR
            mov
  [esp+0x10]
               EIP
```



DEP & ROP

```
system()'s
                        stack frame
                    ret address
                       first arg
0x08045430:
            ret
system()
0xb7e65190:
            push
                   ebx
0xb7e65191:
            sub
                   esp, 8
0xb7e65194:
                   eax, DWORD PTR
            mov
  [esp+0x10]
               EIP
```



MBE - 03/13/15 **DEP & ROP** Growth

ESP

REWIND

DEP & ROP

14 ; CODE XREF: sub 312FD8

```
push edi
call sub_314623
test eax, eax
jz short loc_313061
cmp [ebp+arg_0], ebp
jnz short loc_313066
mov eax, [ebp+var_84
jb short loc_313066
sub eax, [ebp+var_84
```

```
system() --->
```

0x08045430: ret

.....

system()

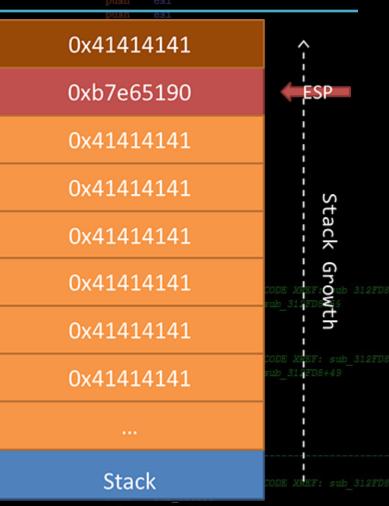
0xb7e65190: push ebx

0xb7e65191: sub esp, 8

0xb7e65194: mov eax, DWORD PTR

[esp+0x10]

• • •



or eax, 8007000

```
push edi
call sub_314623
test eax, eax
jz short loc_313061
cmp [ebp+arg_0], ebp
jnz short loc_313066
mov eax, [ebp+var_84
jb short loc_313066
sub eax, [ebp+var_84
```

```
system()
                      ret address
                         first arg --->
                          "cat flag.txt"
0x08045430:
             ret
                     \mathsf{EIP}
system()
0xb7e65190:
             push
                    ebx
0xb7e65191:
             sub
                    esp, 8
0xb7e65194:
             mov
                    eax, DWORD PTR
  [esp+0x10]
```



MBE - 03/13/15 DEP & ROP

```
call sub_314623
cest eax, eax

jz short loc_31306D

[ebp+arg_0], ebx

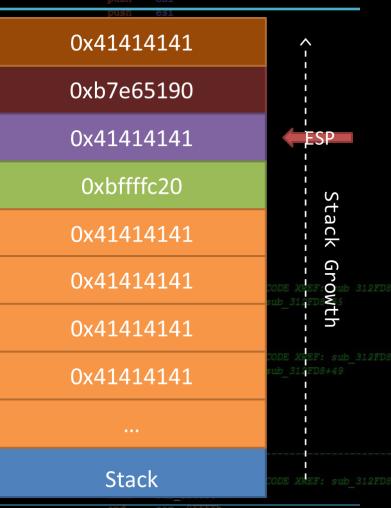
inz short loc_313066

cov eax, [ebp+var_70

cmp eax, [ebp+var_84

sub eax, [ebp+var_84
```

```
system()
                     ret address
                       first arg --->
                        "cat flag.txt"
0x08045430:
            ret
system()
0xb7e65190:
            push
                   ebx
0xb7e65191:
            sub
                   esp, 8
0xb7e65194:
            mov
                   eax, DWORD PTR
  [esp+0x10]
```



MBE - 03/13/15 DEP & ROP

ESP

Growth

```
system()'s
                       stack frame
                    ret address
                       first arg --->
                        "cat flag.txt"
0x08045430:
            ret
system()
0xb7e65190:
            push
                   ebx
0xb7e65191:
            sub
                   esp, 8
0xb7e65194:
                   eax, DWORD PTR
            mov
  [esp+0x10]
               EIP
```



```
        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
```

```
ESP
                            system()'s
                                                         0x0000000
                           stack frame
                                                         0x0000000
                        ret address
                                                         0x41414141
                          first arg
                                                          0xbffffc20
                                                                                Stack
                                                                                Growth
                                                         0x41414141
system()
                                                         0x41414141
0xb7e65190:
              push
                      ebx
0xb7e65191:
              sub
                      esp, 8
                                                         0x41414141
0xb7e65194:
                      eax, DWORD PTR
              mov
  [esp+0x10]
                 \mathsf{EIP}
                                                            Stack
                                                                            DE XREF: sub 312FD8
```

MBE - 03/13/15 DEP & ROP

Chaining Calls

```
0x41414141
     open() --->
                          0xb7eff740
       pivot --->
                          0x08046a4c
  first arg --->
                          0xbffffc20
second arg --->
                          0x0000000
     read() --->
                          0xb7effbd0
ret address --->
                          0x080453ad
  first arg --->
                          0x0000003
                            Stack
```

```
SP 10c_31306D
  10c 31306D
Bub 3140F3
                         20
```