What is ASLR?

A: Address

S: Space

L: Layout

R: Randomization



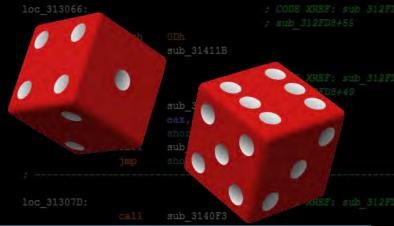
Course Terminology

Address Space Layout Randomization

An exploit mitigation technology used to ensure that address ranges for important memory segments are random for every execution

Meant to mitigate exploits leveraging hardcoded stack, heap, code, libc addresses

Known as ASLR for short



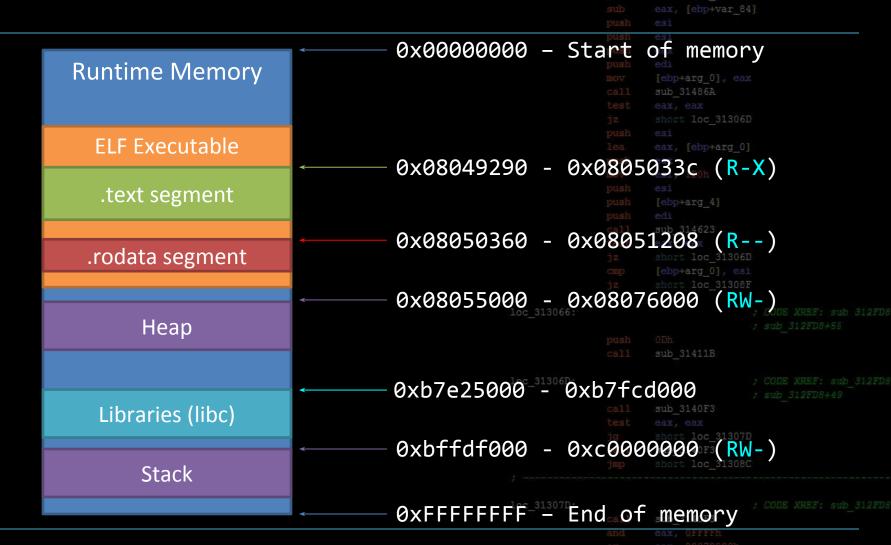
MBE - 03/31/15 **ASLR**

Runtime Process Without ASI



Runtime Memory	0x00000000 - Start of memory push mov [ebp+arg_0], eax call sub_31486A test eax, eax jz short loc_31306D
ELF Executable	push esi lea eax, [ebp+arg_0]
.text segment	0x08049290 - 0x0805033c ₀ (R-X) push esi push [ebp+arg_4] push edi
.rodata segment	0x08050360 - 0x08051208 (R)
	<pre>0x08055000 - 0x08076000 (RW-)</pre>
Heap	loc_313066:
	• 0xb7e25000 - 0xb7fcd000
Libraries (libc)	call sub_3140F3 test eax, eax
Stack	0xbffdf000 - 0xc0000000 (RW-)
	• Oxffffffffffffend.of memory * CODE XREF: Sub_312

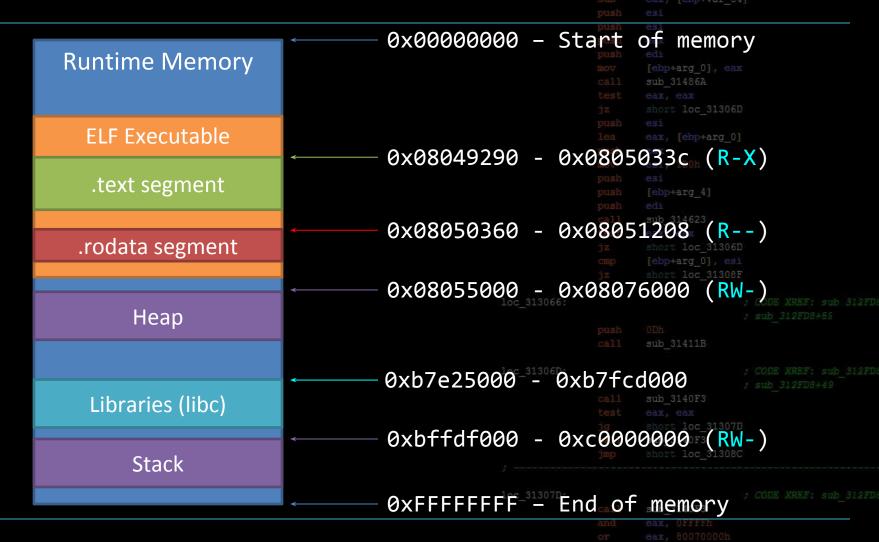
Run #1 Without ASLR



Run #2 Without ASLR



Run #3 Without ASLR



```
12
```

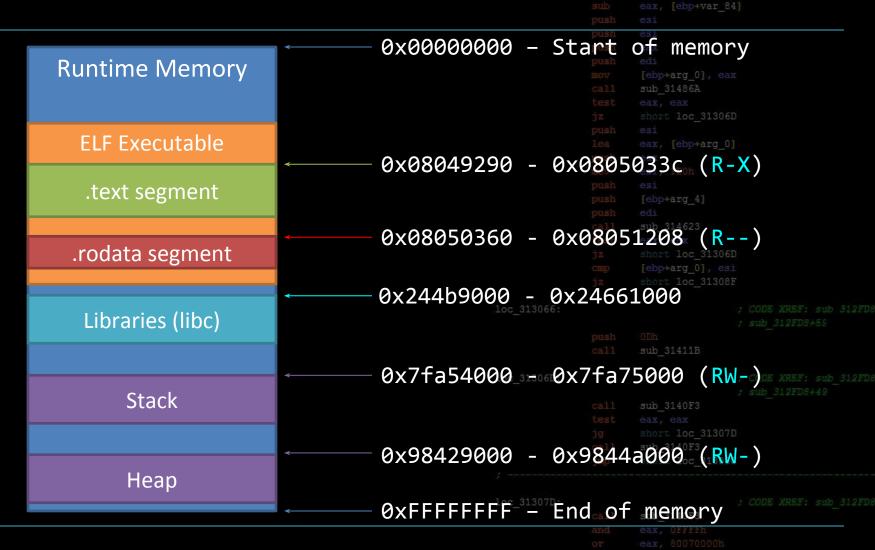
ya so, nothing changes...

Runtime Process Without A



	<pre>0x00000000 - Start of memory</pre>
Runtime Memory	push edi mov [ebp+arg_0], eax call sub_31486A test eax, eax jz short loc_31306D push esi
ELF Executable	lea eax, [ebp+arg_0]
.text segment	0x08049290 - 0x0805033c _n (R-X) push esi push [ebp+arg_4] push edi
us data as sus sust	0x08050360 - 0x08051208 (R)
.rodata segment	jz short loc_31306D cmp [ebp+arg_0], esi
	0x08055000 - 0x08076000 (RW-)
Неар	; sub_312FD8+55 push
	• 0xb7e25000 - 0xb7fcd000
Libraries (libc)	call sub_3140F3 test eax, eax
	<pre>0xbffdf000 - 0xc0000000 (RW-)</pre>
Stack	jmp short loc_31308C *
	• OxfFFFFFF - End of memory

Run #1 With ASLR



Run #2 With ASLR

```
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
```

```
0x00540000 - 0x006e8000
Libraries (libc)
ELF Executable
                           0 \times 08049290 - 0 \times 0805033c (R-X)
 .text segment
                           0x08050360 - 0x08051208 (R--)
.rodata segment
                           0x10962000 = 0x10983000 (RW-)
    Stack
                           0xa07ee000 - 0xa080f000 (RW-)
    Heap
                           0xFFFFFFFFF - End of memory
```

un #3 With	ASLR inz short loc_313066 mov eax, [ebp+var_70] cmp eax, [ebp+var_84] jb short loc_313066 sub eax, [ebp+var_84] push esi
Runtime Memory	0x00000000 - Start of memory mov
ELF Executable	push est lea eax, [ebp+arg_0] ← 0x08049290 - 0x0805033c _{on} (R-X)
.text segment	push [ebp+arg_4] push edi
.rodata segment	0x08050360 - 0x08051208 (R)
Stack	→ Ox094fb000 - Ox0951c000 (RW-) Sub_312FD8+55
Неар	0x43db2000 - 0x43dd3000 (RW-)

0xbf8c3000 - 0xbf8e4000 Libraries (libc) 0xFFFFFFFFFF = End of memory

> Open up a terminal.

- > Open up a terminal.
- > Type "cat /proc/self/maps"

```
18
```

- > Open up a terminal.
- > Type "cat /proc/self/maps"
- > Repeat a few times :)

- > Open up a terminal.
- > Type "cat /proc/self/maps"
- > Repeat a few times :)

You'll see lots of lines like this:

bfe49000-bfe6a000 rw-p 00000000 00:00 0

• • •

bfa23000-bfa44000 rw-p 00000000 00:00 0

• • •

bfdab000-bfdcc000 rw-p 00000000 00:00 0

```
[stack]
[stack]
[stack]
```

- > Open up a terminal.
- > Type "cat /proc/self/maps"
- > Repeat a few times :)
- Stack Address Changes

```
21
```

- > Open up a terminal.
- > Type "cat /proc/self/maps"
- > Repeat a few times :)
- Stack Address Changes
- Heap Address Changes

```
22
```

- > Open up a terminal.
- > Type "cat /proc/self/maps"
- > Repeat a few times :)
- Stack Address Changes
- Heap Address Changes
- Library Addresses Change

```
23
```

ASLR Basics

```
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
```

 Memory segments are no longer in static address ranges, rather they are unique for every execution

ASLR Basics

```
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
```

- Memory segments are no longer in static address ranges, rather they are unique for every execution
- A simple stack smash may get you control of EIP, but what does it matter if you have no idea where you can go with it?

ASLR Basics

```
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
```

 Memory segments are no longer in static address ranges, rather they are unique for every execution

```
jz short loc_31306D
push esi
lea eax, [ebp+arg_0]
push eax
mov esi, 1D0h
```

- A simple stack smash may get you control of EIP, but what does it matter if you have no idea where you can go with it?
 - The essence of ASLR

```
jz short loc_31306D
cmp [ebp+arg_0], esi
jz short loc_31308F
```

You must work with no expectation of where anything is in memory anymore

History of ASLR

- When was ASLR implemented?
 - May 1st, 2004 OpenBSD 3.5 (mmap)
 - June 17th, 2005 Linux Kernel 2.6.12 (stack, mmap)
 - January 30th, 2007 Windows Vista (full) sub_314623 eax, eax
 - October 26th, 2007 Mac OSX 10.5 Leopard (sys libraries)
 - October 21st, 2010 Windows Phone 7 (full)
 - March 11th, 2011 iPhone iOS 4.3 (full)
 - July 20th, 2011 Mac OSX 10.7 Lion (full)

```
### CODE XREF: sub 312FD:

### CODE XREF: sub 312FD:

### Sub_312FD8+59

### CODE XREF: sub_312FD:

### Sub_31411B

### CODE XREF: sub_312FD:

### Sub_3140F3

### Sub_3140F3

### CODE XREF: sub_312FD:

### CODE XREF: sub_312FD:

#### Sub_3140F3
```

MBE - 03/31/15 ASLR

1308C: ;

History of ASLR

- When was ASLR implemented?
 - May 1st, 2004 OpenBSD 3.5 (mmap)
 - June 17th, 2005 Linux Kernel 2.6.12 (stack, mmap)
 - January 30th, 2007 Windows Vista (full) st sub_31462
 - October 26th, 2007 Mac OSX 10.5 Leopard (sys libraries)
 - October 21st, 2010 Windows Phone 7 (full)
 - March 11th, 2011 iPhone iOS 4.3 (full)
 - July 20th, 2011 Mac OSX 10.7 Lion (full)

perspective: markus is accepted to RPI

Reminder: Security is

Security is rapidly evolving

Checking for ASLR

```
cat /proc/sys/kernel/randomize_va_space
```

Checking for ASLR

```
cat /proc/sys/kernel/randomize_va_space
```

MBE - 03/31/15 **ASLR** 31

```
Checking for ASLR
   cat /proc/sys/kernel/randomize va space
 0: No ASLR
 1: Conservative Randomization
     (Stack, Heap, Shared Libs, PIE, mmap(), VDRO)
 2: Full Randomization
    (Conservative Randomization + memory managed via brk())
                                              sub 3140F3
```