			sub_314623	
			short loc 31306D	
			[ebp+arg 0], ebx	
			short loc_313066	
			eax, [ebp+var_70	
			eax, [ebp+var_84	
			short loc_313066	
			eax, [epp+var_64	
			[ebp+arg_0], eax	
			sub_31486A	
		test	eax, eax	
Addroce Space Lav			nndo	
AUDIESS SUALE LAV		NC	e x, [()p+ r 0)	
		push	eax	
	•		Lebotard 41	
Randomiza	ntinr	push	edi	
Νατισυτιτζα		call	sub_314623	
		jz	short loc_31306D	
			short loc 31308F	
Modern Rinary Ev	alaitat	inn		
IVIOUEITI DIITALY LA	Jional			
			ODh	
CCCL 10CQ Corir	-701	call	sub_31411B	
USUI 4900 - SPIII	Ig ZUI)		
Detrial Diem			sub_3140F3	
Patrick Bieri	nat			
			short loc_31307D	
			sub_3140F3	
	loc_31307D:			
		call	sub_3140F3	
		and	eax, OFFFFh	
MBE - 03/31/15			car, courououn	1
	loc 31308C:			; CODE XREF: sub 312FD2

Lecture Overview

1. Introducing ASLR

2. Position Independent Executables

ASLR

- 3. Bypassing ASLR, Examples
- 4. Conclusion

MBE - 03/31/15

		sub_314623	
		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]	
	push	esi	
		Tebp+arg 01, eax	
		sub 31486A	
		short loc 31306D	
22		ear [ebotard 0]	
		eax, [copy,arg_o]	
		Coloniana Al	
		[epp+arg_4]	
		cul 214622	
		3UD_314023	
		eax, eax	
		short Loc_31306D	
		[epp+arg_u], esi	
		short loc_31308F	
6:			
		ODD	
		sub_31411B	
		sub_3140F3	
		short loc_31307D	
		sub_3140F3	
		short loc_31308C	
D:			
	call	sub_3140F3	
			2
			CONTRACTOR OF CONTRACTOR

Modern Exploit Mitigation

	sub_314623
	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]
n	es1

- Theres a number of modern exploit mitigations that we've generally been turning off for the labs and exercises
 - DEP
 - ASLR
 - Stack Canaries
 - ... ?

IIU	exel	CISES_31306		
		eax, [ebp+arg_0		
		[ebp+arg_4]		
		sub_314623		
		short loc_31306		
		[ebp+arg_0], es		
		short loc_31308		
066:				
		sub_31411B		
06D:				
		sub_3140F3		
		short loc_31307		
		sub_3140F3		
		short loc_31308	C	
07D:				
	call	sub_3140F3		
	and	eax, OFFFFh		
				2
				CODE XREF: sub 312FD8

ASLR

Modern Exploit Mitigation

sub_314623
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]
es1

- Theres a number of modern exploit mitigations that we've generally been turning off for the labs and exercises
 - DEP
 - ASLR
 - Stack Canaries
 - ... ?

MBE

eax, [ebp+arg_0]
[ebp+arg_4]
sub_314623
short loc_31306D
[ebp+arg_0], esi
short loc_31308F

We turned on DEP and introduced ROP last lab

03/31/15	ASLK	loc_31308C:		[ebp+var_4], eax	; CODE XREF: sub_312FD8
02/21/15	ACID		and or	eax, OFFFFh eax, 80070000h	4
		loc_31307D:		sub_3140F3	
				sub_3140F3 eax, eax short loc_31307F sub_3140F3 short loc_31308C	
		loc_31306D:		SUD_014IIB	

Modern Exploit Mitigations

Theres a number of modern exploit mitigati	ons	that we	've
generally been turning off for the labs and e	exer	cises_31306D	

- DEP
- ASLR
- Stack Canaries
- ... ?

- XCICISES_31306D
 ush esi
 ea eax, [ebp+arg_0]
 ush eax
 ov esi, 1D0h
 ush esi
 ush [ebp+arg_4]
 ush edi
 all sub_314623
 est eax, eax
 z short loc_31306D
- We turned on DEP and introduced ROP last lab

- ; CODE XREF: sub 312FD8 ; sub_312FD8+55
- Today we turn ASLR back on for the remainder of the course

]g call jmp ; loc 31307D:		jg short loc_31307D call sub_3140F3 jmp short loc_31308C		
				sub_3140F3		
			and	eax, OFFFFh		
MBE - 03/31/15	ASLR				5	
		loc_31308C:				
				Tehn+var 41 eav		

What is ASLR?

A: Address S: Space L: Layout R: Randomization

sub 3140F 6

MBE - 03/31/15

ASLR

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 eai
- 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

ASLR

• Known as ASLR for short











	sub_314623
	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]
	[ebp+arg_0], eax
	sub_31486A
	short loc_31306D
	eax, [ebp+arg_0]
	[ebp+arg_4]
	sub_314623
	short loc_31306D
	[ebp+arg_0], esi
	short loc_31308F
	sub 31411B

ya so, nothing changes...

MBE - 03/31/15

			sub_3140F3 short loc_31308	c
	, loc_31307D:			
			sub_3140F3	
		and	eax, OFFFFh	
ASLR				12
	loc 31308C:			
			Tebp+var 41, ea	









> Open up a terminal.

		sub 314623	
		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]	
	push	esi	
		[ebp+arg 0], eax	
		sub 31486A	
		eax, eax	
		short loc 31306D	
		eax, [ebp+arg 0]	
		eax	
		[ebp+arg 4]	
		sub 314623	
		eax eax	
		short loc 31306D	
		[ebp+arg 0] esi	
		short loc 31308F	
		sub 31411B	
D:			
		sub 3140F3	
		eax, eax	
		short loc 31307D	
		sub 3140F3	
		short loc 31308C	
D:			
		sub 3140F3	
	and	eax, OFFFFh	
			17
C:			CODE XREF: sub 312FD8

> Open up a terminal.

MBE - 03/31/15

> Type "cat /proc/self/maps"

ASLR

		sub_314623	
		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]	
	push	esi	
		Tebp+arg 01, eax	
		sub 31486A	
		short loc 31306D	
		eax. [ebp+arg 0]	
		ear [oup and_o]	
		Tebriary Al	
		[coprary_1]	
		gub 21/622	
		500_J1402J	
		shart log 31306D	
		Short 100_31300D	
		[coptarg_0], csr	
		SHOTE TOC_313001	
13066:			
		sub 31411B	
1306D:			
		sub 3140F3	
		eax. eax	
		short loc 31307D	
		sub 3140F3	
		short loc 31308C	
1307D:			
		sub_3140F3	
	and	eax, OFFFFh	
			18
L308C:			

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

		sub_314623	
		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]	
	push	esi	
		[ebp+arg_0], eax	
		sub_31486A	
		short loc_31306D	
		eax, [ebp+arg_0]	
		[ebp+arg_4]	
		sub_314623	
		short loc_31306D	
		[ebp+arg_0], esi	
		short loc_31308F	
c 313066:			
		sub_31411B	
c_31306D:			
		sub_3140F3	
		short loc_31307D	
		sub_3140F3	
		short loc_31308C	
04.0077			
c_31307D:			
	Call	sub_3140F3	
			10
			19

> Open up a terminal.

> Type "cat /proc/self/maps"

> Repeat a few times :)

. . .

MBE - 03/31/15

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

ASLR

	cui 214622
	3UD_314623
	edx, edx
	Short loc_sisuon
	[ebp+arg_0], ebx
	Short loc_slau66
	eax, [ebp+var_/0]
	eax, [epp+var_64]
	SHOLT IOC 313066
	eax, [ebp+var_64]
pusn	esi
	[ebp+arg_0], eax
	3UD_31486A
	eax, eax
	short loc_31306D
	eax, [ebp+arg_0]
	eax
	[ebp+arg_4]
	sub_314623
	short loc_31306D
	[ebp+arg_0], esi
	short loc_31308F
	Istacki
	; sub_312FD8+55
	sub_31411B
	; CODE XREF: sub_31.
	a gash
	short loc_31307D
	sub 3140F3

[stack]

20

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

ASLR

- > Repeat a few times :)
- Stack Address Changes

		sub_314623	
		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]	
	push	esi	
		Tehntard 01 eav	
		sub 314863	
		short loc 31305D	
		aci	
		ear [ebriand 0]	
		car, [enb+ard_n]	
		cal 1Dab	
		[epp+arg_4]	
		sub_314623	
		short loc_31306D	
		[ebp+arg_0], esi	
		short loc_31308F	
loc_313066:			
		sub_31411B	
loc_31306D:			
		sub_3140F3	
		short loc_31307D	
		sub_3140F3	
		short loc_31308C	
loc_31307D:			
	call	sub_3140F3	
	and	eax, OFFFFh	
			21
loc 31308C:			CODE XREF: sub 312FD8

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

		sub 314623	
		short loc 31306	
		[ebp+arg 0], ebp	
		short loc 313060	6
		eax, [ebp+var 70	21
		eax, [ebp+var 84	11
		short loc 313060	6
		eax. [ebp+var 84	41
	push	esi	
		Tebp+arg 01, eau	
		sub 31486A	
		eax, eax	
		short loc 31306	D
		eax, [ebp+arg 0]	
		eax.	
		Tebp+arg 41	
		edi	
		sub 314623	
		eax, eax	
		short loc 31306	0
		Tebp+arg 01, est	
		short loc 313081	
loc 313066:			
		sub 31411B	
loc 31306D:			
		sub 3140F3	
		short loc 31307	D
		sub 3140F3	
		short loc 313080	
loc 31307D:			
		sub 3140F3	
	and	eax, OFFFFh	
			22
loc 31308C:			; CODE XREF: sub 312FD8

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

		sub_314623	
		short loc 31306	
		[ebp+arg 0], eb	
		short loc 31306	
		eax, [ebp+var 7	01
		eax. [ebp+var 8	41
		short loc 31306	
		eax. [ebp+var 8	41
	push	esi	
		Tebp+arg 01, ea	
		sub 31486A	
		ear ear	
		short loc 31306	
		ear [ebotard 0	
		[enp+arg_4]	
		SUD_314623	
		eax, eax	
		SHOTE LOC_31306	
		[epp+arg_u], es	
		short loc_31308	
0C_313000:			
		ULIN OA 4447D	
		SUD_31411B	
0C_3130001:			
		SUD_314013	
		eax, eax	
		anort 10c_31307	
		sub_3140F3	
		short loc_31308	C.
oc 31307D:			
		sub 3140F3	
	and	eax. OFFFFh	
			23
00 313080			· CODE XREE and 212EDS

ASLR

ASLR Basics	push call test jz cmp jnz mov cmp jb sub push push	edi sub_314623 eax, eax short loc_31306E [ebp+arg_0], ebx short loc_313066 eax, [ebp+var_70 eax, [ebp+var_84 short loc_313066 eax, [ebp+var_84 esi esi	
 Memory segments are no longer in static a 			
• Memory segments are no longer in static a	uui		zs,
rather they are unique for every execution		eax, eax short loc_31306E esi eax, [ebp+arg_0] eax esi, 1D0h esi [ebp+arg_4] edi sub_314623 eax, eax short loc_31306E [ebp+arg_0], esi short loc_31308E) ; CODE XREF: sub 312FD8 ; sub_312FD8+55
loc_31306D: ; loc_31307D:		sub_3140F3 eax, eax short loc_31307F sub_3140F3 short loc_313080	
MBE - 03/31/15 ASLR	call and or	sub_3140F3 eax, 0FFFFh eax, 80070000h	24 ; CODE XREF: sub_312FD8

ASLR Basics

SLR Basics		push call test jz cmp jnz mov cmp jb sub push	edi sub_314623 eax, eax short loc_313060 [ebp+arg_0], ebp short loc_313060 eax, [ebp+var_70 eax, [ebp+var_84 short loc_313060 eax, [ebp+var_84 eai	
		push push push	esi eax edi	
Nemory segments are r	no longer in static a	aar	ess range	es,
rather they are unique f	or every execution		eax, eax short loc_31306 esi eax, [ebp+arg_0] eax	
A simple stack smash m does it matter if vou hav	ay get you control o ve no idea where vo	of El ou c	IP, but w an go wi	hat th it?
	,	jz cmp jz	short loc_31306 [ebp+arg_0], es: short loc_31308]	
	loc_313066:			
			ODh sub_31411B	
	loc_31306D:			
			sub_3140F3 eax, eax short loc_31307	
			sub_3140F3 short loc_31308(
	loc_31307D:	call	sub_3140F3	; CODE XREF: sub_312FD8
		and	eax, OFFFFh eax, 80070000h	
- 03/31/15	ASLR 10c_31308C:		Topputter 41	25 ; CODE XREF: sub_312FD8
			CODTVAL 11, Ed.	

MBE - 03/31/15

ightarrow

ullet

ASLR Basics

- 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?
 - 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
 loc_31306D:
 code XREF: sub_3
 sub_312ED8+55

				eax, eax short loc_313071 sub_3140F3 short loc_313080	
		;; loc_31307D;		sub_3140F3	
MBE - 03/31/15	ASLR	loc_31308C:	and or	eax, OFFFFh eax, 80070000h	26 ; CODE XREF: sub_312FD8

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)
 - 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)

		sub_314623	
		Short loc_31306D	
		short loc 313066	
		eav [ebp+var 70]	
		eax, [ebp+var_/6]	
		short loc 313066	
		eax, [ebp+var 84]	
p	ush	es1	
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~		[ebp+arg_0], eax	
eu! 🗠		sub_31486A	
j, j		short loc_31306D	
nman 👘			
		eax, [ebp+arg_0]	
) ら 1 ) / <mark>c t</mark> つ		mman	
	UN,	IIIIIap	
		[epp+arg_4]	
Vieta (full)		en1	
VISLA (IUII)		SUD_314023	
Ť		short loc 31306D	
	IIID	Tepp+arg 011. dsi	•
10.5 Leop	ard	SVS ID	raries
loc_313066:			
Phone 7 (	ΠП		
	GI	IDh	
		sub_31411B	
/ 2 /full)			
4.9_\IUII/			
		sub_3140F3	
LION (TUII)			
		short loc_31307D	
		Sub_3140F3	
		anore 100_313080	
loc 31307D:			
	a11	sub 3140F3	
a	nd	eax, OFFFFh	

۸	C	D	
н	Э	Г	

### History of ASLR

- When was ASLR implemented?
  - May 1st, 2004 OpenBSD 3.5 (mmap)
  - June 17th, 2005 Linux Kernel 2.6.12 (stack, mmap)

ASLR

- January 30th, 2007 Windows Vista (full)
- 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



								0+arg_0]	
Reminder:							esi [ebp+arg_ edi sub_31462	_4] 23	
Security	is	ra	pic	lly	e١	test jz	eax, eax short loc		

		TOC_27200D:		<pre>sub_3140F3 eax, eax short loc_31307 sub_3140F3 short loc_31308</pre>	) CODE AREF: SUB 312FD8 ; sub 312FD8+49 D C
		loc_31307D:		sub 3140F3	
			and	eax, OFFFFh	
MBE - 03/31/15	ASLR	loc_31308C:		car, sourououn	29 ; CODE XREF: sub_312FD8

### Checking for ASLR

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

#### \$ cat /proc/sys/kernel/randomize_va_space

MBE - 03/31/15 ASLR (CODE XREF: sub_312FD8) MBE - 03/31/15 ASLR (CODE XREF: sub_312FD8) and eax, 0FFFh or eax, 80070000h 10c_31308C: ; CODE XREF: sub_312FD8

### Checking for ASLR

#### \$ cat /proc/sys/kernel/randomize_va_space

31

**ASLR** 

MBE - 03/31/15

2

			<b>f</b>		
nec	KI	ng	TOr	AS	LK
		0			

**1: Conservative Randomization** 

ecking for ASLR	<pre>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</pre>
at /proc/sys/kernel/random	mize_var_spush esi test eax, eax jz short loc_31306D push esi lea eax, [ebp+arg_0] push eax
lo ASLR onservative Randomization	<pre>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</pre>
(Stack, Heap, Shared Libs, PIE, mm	ap ⁽¹⁾ , VDRO) ; CODE XREF: sub 312

2: Full Randomization

\$

2

**0: No ASLR** 

			η.	£.	n

(Conservative Randomization + memory managed via brk())



#### Lecture Overview

#### 1. Introducing ASLR

- 2. Position Independent Executables
- 3. Bypassing ASLR, Examples
- 4. Conclusion

	sub_314623	
	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]	
push	esi	
	[ebp+arg_0], eax	
	sub_31486A	
	short loc_31306D	
	eax, [ebp+arg_0]	
	[ebp+arg_4]	
	sub_314623	
	short loc 31306D	
	[ebp+arg_0], esi	
	short loc 31308F	
	sub 31411B	
	sub 3140F3	
	short loc 31307D	
	sub 3140F3	
	short loc 31308C	
	sub_3140F3	
and	eax, OFFFFh	
		33

#### ELF's and ASLR

#### On Linux, not everything is randomized...

	sub_314623	
	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]	
push	esi	
	[ebp+arg_0], eax	
	sub 31486A	
	short loc 31306D	
	eax, [ebp+arg_0]	
	[ebp+arg 4]	
	sub 314623	
	short loc 31306D	
	[ebp+arg 0], esi	
	short loc 31308F	
	sub_31411B	
	sub_3140F3	
	short loc_31307D	
	sub_3140F3	
	short loc_31308C	
	aub 31/0F2	

34









#### Not Randomized

• Main ELF Binary

ME

- .text / .plt / .init / .fini Code Segments (R-X)
- .got / .got.plt / .data / .bss Misc Data Segments (RW-)

sub 31486A

- .rodata Read Only Data Segment (R--)
- At minimum, we can probably find some ROP gadgets!
  - Warning: They won't be pretty gadgets loc_313066:

		loc_31308C:		[ebp+var 4], ea	
E - 03/31/15	ASLR				39
			and	eax, OFFFFh	
				sub_3140F3	
		loc_31307D:			
				short loc_31308	
				sub_3140F3	
				short loc_31307	
				sub_3140F3	
		loc 31306D:			
				sub_31411B	
				0Dh	

### **Course Terminology**

- Position Independent Executable
  - Executables compiled such that their base address does not matter, 'position independent code'
  - Shared Libs /must/ be compiled like this on modern Linux

ASLR

- eg: libc
- Known as **PIE** for short



313066

## Applying ASLR to ELF's

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

 To make an executable position independent, you must compile it with the flags -pie -fPIE

\$ gcc -pie -fPIE -o tester tester.c

**ASLR** 

cmp [ebp+arg_0], es; jz short loc 31308



## Applying ASLR to ELF's

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 typush edi

 To make an executable position independent, you must compile it with the flags -pie -fPIE

\$ gcc -pie -fPIE -o tester tester.c

ASLR

mp [ebp+arg_0], esi
jz short loc 31308F



loc_31307D: ; CODE XREF: sub_312FD8 call sub_3140F3 and eax, 0FFFFh or eax, 80070000h loc_31308C: ; CODE XREF: sub_312FD8

### Checking for PIE

Most binaries aren't actually compiled as PIE

	doom@upuncu:~>			
	doom@ubuntu:~\$	checksecfile	/bin/bash	
	RELRO	STACK CANARY	NX	
	Partial RELRO	Canary found	NX enabled	
	doom@ubuntu:~\$	checksecfile	/bin/ping	
	RELRO	STACK CANARY	NX	
	Partial RELRO	Canary found	NX enabled	
	doom@ubuntu:~\$	checksecfile	/usr/sbin/sshd	
	RELRO	STACK CANARY	NX	
	Full RELRO	Canary found	NX enabled	
	doom@ubuntu:~\$			
I	doom@ubuntutes\$			

		sub_31	4623	
			loc_31306D	
			rg_0], ebx	
			loc_313066	
			ebp+var_70]	
			ebp+var_84]	
			loc_313066	
			ebp+var_84]	
	push	esi		
_	push			
as	Vo <del>n</del>		rg_0], eax	
	call	sub_31	486A	
			loc_31306D	
TH	RUNPATH		FILE	
DATH	No PIINE	лтн	/hin/hagh	
FAIII	NO KONEI	ATH	/ DIII/ Dasii	
iH.	RUNPATH		FILE	
PATH	No RUNP	ATH	/bin/ping	
TH	RUNPATH		FILE	
РАТН	No RUNP	АТН	/usr/shin/sahd	
	no nomi		/ 402/ 00111/ 00114	

RPZ

RP# No

RPA

enabled

ASLR

 Generally only on remote services, as you don't want your server to get owned

	loc_31307D:			
			sub_3140F3	
		and	eax, OFFFFh	
				43
	loc_31308C:			
			Tebp+var 41, e	

#### Lecture Overview

- 1. Introducing ASLR
- 2. Position Independent Executables
- 3. Bypassing ASLR, Examples
- 4. Conclusion

44

MBE - 03/31/15

- Assume you can get control of EIP
- What information does ASLR deprive us of?

ASLR

		sub_314623
		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]
	push	esi
		[ebp+arg_0], eax
		sub_31486A
		short loc_31306D
		eax, [ebp+arg_0]
•		eax
D donri	170	
n uepii	VC	
		[ebp+arg_4]
		sub_314623
		short loc 31306D
		[ebp-
		sh
loc_313066:		XREF: sub 312FD8
		12ED8+55
		sub_31411B
a Salara		
loc_31306D:		ODE XREF: SUD 312FD8
		SUD_312008+49
		3ub_3140
		Gan, Gan
		sub 2140F2
		Sub_3140F3
		51010 100 000
log 31307D.		· CODE VDEE: auto 212EDS
100_010000		sub 3140F3
	and	eax. OFFFF
		45
loc 31308C:		: CODE XREF: sub 312EDA
		[ebp+var 4], eax

- Assume you can get control of EIP
- sub eax, [ebp+var_84]
  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]
- What information does ASLR deprive us of?
  - You don't know the address of ANYTHING



- Assume you can get control of EIP
- What information does ASLR deprive us of?
  - You don't know the address of ANYTHING

ASLR

How can we get that information?
 Or work around it?



- There's a few common ways to bypass ASLF
  - Information disclosure (aka info leak)
  - Partial address overwrite + Crash State
  - Partial address overwrite + Bruteforce

	sub_314623
	short loc_31306D
	[ebp+arg_0], ebx
	short loc_313066
	eax, [ebp+var_70]
	eax, [ebp+var_84]
	short loc_313066
	eax, [ebp+var_64]
push	031
170	
Jas	
	short loc_31306D
	eax, [ebp+arg_0]

WBL - 05/51/15	AJLN	loc_31308C:			; CODE XREF: sub_312FD&
MRE 02/21/15					19
			and	eax, OFFFFh	
			call	sub_3140F3	
		loc_31307D:			
				short loc 313080	
				snort 10C_313U/I	
				sub_3140F3	
		loc_31306D:			
				sub_31411B	
		loc 313066:			
				short loc_31308H	

### What are Info Leaks?

	sub_314623
	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]
push	esi

- An info leak is when you can extract meaningful information (such as a memory address) from the ASLR protected service or binary
- If you can leak any sort of pointer to code during your exploit, you have likely defeated ASLR
  - Why is a single pointer leak so damning?

0.Dh	
sub_31411B	
sub_3140F3	
short loc 31307D	
sub 3140F3	
short loc 31308C	
sub 3140F3	
eax, OFFFFh	
eax, 80070000h	
	49
	DE XREF: sub 312FD8

push	est
	eax, [ebp+var_84]
	short loc_313066
	eax, [ebp+var_84]
	eax, [ebp+var_70]
	short loc_313066
	[ebp+arg_0], ebx
	short loc_31306D
	sub_314623



	sub_314623
	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]
push	AS1



	sub_314623
	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]
push	AS1



	sub_314623
	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]
push	esi



push	est
	eax, [ebp+var_84]
	short loc_313066
	eax, [ebp+var_84]
	eax, [ebp+var_70]
	short loc_313066
	[ebp+arg_0], ebx
	short loc_31306D
	sub_314623



#### By Example:

#### -You have a copy of the libc binary, ASLR is or

		sub 314623			
		short loc 31306D			
		Tebp+arg 01, ebx			
		short loc 313066			
		eax. [ebp+var 70]			
		eav [ebp+var_70]			
		short log 313066			
		eav [ebn_war 84]			
		esi			
	push	031			
		Lebo+arg 01 eav			
		aub 31/967			
		ADV ADV			
		short loc 31306D			
-					
S	ON	ear [ebo+arg_0]			
		ear [asp.arg_o]			
		Tebo+arg 41			
		edi			
		sub 314623			
		eax, eax			
		short loc 31306D			
		Tebp+arg 01, esi			
		short loc 31308F			
					12FD8
		sub 31411B			
		sub_3140F3			
		short loc_31307D			
		sub 3140F3			
		short loc_31308C			
		sub_3140F3			
	and	eax, OFFFFh			
			5	5	

#### By Example:

-You have a copy of the libc binary, ASLR is or

-You've leaked a pointer off the stack to printf printf() is @ 0xb7e72280

**ASLR** 

		sub 314623	
		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	
	push	esi	
		<pre>[ebp+arg_0], eax</pre>	
		sub_31486A	
		short loc_31306D	
SIR IS	on		
	lca	eax, [ebp+arg_0]	
to pr	vint		
	call	sub 314623	
		eav eav	
		short loc 31306	
		[ebp+arg 0], esi	
		short loc 31308E	
c 313066:			
		sub_31411B	
c_31306D:			
		sub_3140F3	
		short loc_31307D	
		sub_3140F3	
		short loc_313080	
C_31307D:			
	Call	aup_st40fs	
		eax, urrra	
		dan, odurououn	56
c 31309C+			· CODE VEEF, auth 212502
- 313000.			

MBE - 03/31/15

#### By Example:

-You have a copy of the libc binary, ASLR is on

-You've leaked a pointer off the stack to printf()
printf() is @ 0xb7e72280

-Look at the libc binary, how far away is system() from printf()?
 system() is -0xD0F0 bytes away from printf()

: sub 312FD8+49

WIBE - 03/31/15	ASLR	loc 31308C:			: CODE XREF: sub 312FD8
			and	eax, OFFFFh	
				sub_3140F3	
		loc_31307D:			
				sub_3140F3	
				short loc_3130	)7D
				sub 3140F3	

#### By Example:

-You have a copy of the libc binary, ASLR is on

-You've leaked a pointer off the stack to printf()
printf() is @ 0xb7e72280

58

-Look at the libc binary, how far away is system() from printf()?
 system() is -0xD0F0 bytes away from printf()

therefore system() is at @ 0xb7e65190.100313070 (0xb7e65190-0xD0F0)

/levels/lecture/a	slr/as	slr_le	call test jz cmp ov pp jb sub push	sub_314623 eax, eax short loc_31306D [ebp+arg_0], ebx hor oc_313066 ex, [bp+var_70] abort loc_313066 eax, [ebp+var_84] esi	
ssh <u>lecture@warzo</u>	<u>ne.rpi</u>	<u>ls.ec</u>	push push mov 2 call2	esi edi 2bp+arg_0], eax	
Fully Position Independent F	vecutable			<pre>short loc_31306D esi eax, [ebp+arg_0] eax esi, 1D0h esi</pre>	
gcc -pie -fPIE -fno-stack-p	rotector	./aslr le	push push ak1	esi [ebp+arg_4] edi are_314623	
8 p				<pre>short loc_31306D [ebp+arg_0], esi short loc_31308F</pre>	
Force it to execute the "i_am	_rly_leet	"•functio	<mark>DN</mark> push		
		loc_31306D:		sub_31411B sub_3140F3 eax, eax short loc_31307D sub_3140F3 short loc_31308C	
		;	call and	sub_3140F3	CODE XREF: sub_312FD8
MBE - 03/31/15	ASLR	loc_31308C:		eax, 80070000h	59 CODE XREF: sub_312FD8

/levels/lecture/a	islr/a	slr_l	call test jz cmp enz jb sub push	sub_314623 eax, eax short loc_31306D [ebp+arg_0], ebx bart 2c_313066 ex, 2pp+var_70] short loc_313066 eax, [ebp+var_84] esi	
ssh <u>lecture@warzo</u>	ne.rp	is.ec	push push push mov call	esi eax edi 2bp+arg_0], eax 2bb_31486A	
				eax, eax short loc_31306D esi eax, [ebp+arg_0]	
The exercise is equally as sm this is typically how an infole	all and di ak might	rty as th appear	ne la in t	nst one, k he wild.	out
			test jz cmp jz	<pre>eax, eax short loc_31306D [ebp+arg_0], esi short loc_31308F</pre>	
Can you parse it? Build a ROI	chain b	ased off	It?		
		loc_31306D:		sub_31411B	
				sub_3140F3 eax, eax short loc_31307D sub_3140F3 short loc_31308C	
		;	call	sub_3140F3 eax, OFFFFh	CODE XREF: sub_312FD8
MBE - 03/31/15	ASLR	loc_31308C:		eax, 80070000h [ebp+var 4], eax	60 CODE XREF: sub_312FD8

	sub_314623
	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]
push	es1

- Can be used on hardest scenario of PIE, full ASLR
  - Usually comes with 100% exploit reliability!
  - 'it just works'
- Info leaks are the most used ASLR bypass in real world exploitation as they give assurances
  - Someone's life might depend on your exploit landing

```
ODE XREF: sub_312F
; sub_312FD8+49
```

				short loc_3130 sub_3140F3 short loc_3130		
		loc_31307D:		sub_3140F3		
			and	eax, OFFFFh		
MBE - 03/31/15	ASLR	loc 31308C:			61 ; CODE XREF: sub 312FD8	

#### **Partial Overwrites**

- 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
- Assume you have no way to leak an address, but you can overwrite one

from multiple runs: 0xb756b132 0xb758e132 0xb75e5132 0xb754d132 0xb75cf132

#### Guaranteed 255 byte ROP/ret range around that address

#### Partial Overwrites

	sub_314623
	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]
push	esi

 Assume you have no way to leak an address, but you can overwrite one

from multiple runs: 0xb756b132 0xb758e132 0xb75e5132 0xb754d132 0xb75cf132



ASLR

#### Bruteforcing

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

- Note that these bruteforcing details apply only to Ubuntu 32bit
- Don't bother to try bruteforcing addresses on a 64bit machine of any kind

ASLR

Ubuntu ASLR is rather weak, low entropy

			64 CODE XREF: sub_312FD8
	and	eax, OFFFFh	
		sub_3140F3	
C	call test jg call jmp	sub_3140F3 eax, eax short loc_31307 sub_3140F3 short loc_31308	
	call ntr	ODh sub_31411B	

#### **ASLR** Tips

- What does your crash state look like?
  - What's in the registers?
  - What's on the stack around you?
- Even if you can't easily leak some data address out of a register or off the stack, there's nothing that's stopping you from using it for stuff
  - As always: get creative

				sub_5140F5 short loc_31308	.308C	
		, loc 31307D:				
				sub_3140F3		
			and	eax, OFFFFh		
1BE - 03/31/15	ASLR	loc_31308C:			65 ; CODE XREF: sub_312FD8	

#### Lecture Overview

- 1. Introducing ASLR
- 2. Position Independent Executables
- 3. Bypassing ASLR, Examples
- 4. Conclusion

		eax, 80070000h	
	and	eax, OFFFFh	
		sub_3140F3	
		short loc_31308C	
		sub_3140F3	
		short loc_31307D	
		sub_3140F3	
		sub 31411B	
		short loc_31308F	
		[ebp+arg_0], esi	
		short loc_31306D	
		sub_314623	
		[ebp+arg_4]	
J		eax, [ebp+arg 0]	
		short loc 31306D	
		eax, eax	
		sub 31486A	
		[ebp+arg 0], eax	
	push	631	
		eax, [epp+var_84]	
		short Loc_313066	
		eax, [ebp+var_84]	
		eax, [ebp+var_70]	
		short loc_313066	
		[ebp+arg_0], ebx	
		short loc_31306D	
		sub_314623	

### In Closing

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

- Like other mitigation technologies, ASLR is a 'tack on' solution that only makes things harder
- The vulnerabilities and exploits become both more complex and precise the deeper down the rabbit hole we go

		loc_31306D:			
				sub_3140F3 eax, eax short loc_31307 sub_3140F3 short loc_31308	
		loc_31307D:		sub_3140F3	
			and	eax, OFFFFh eax, 80070000h	
MBE - 03/31/15	ASLR	loc_31308C:		[ebo+var 4] ea	67 ; CODE XREF: sub_312FD8

# Modern Exploit Mitigations

- DEP & ASLR are the two main pillars of modern exploit mitigation technologies
- Congrats, being able to bypass these mean that you're probably capable of writing exploits for real vulnerabilities

		eax, 80070000h	
		eax, OFFFFh	
	call	sub_3140F3	
loc_31307D:			
		sub_3140F3 short loc_313080	
		short loc_313071	
		sub 3140F3	
loc_31306D:			
	loc_31306D: ; loc_31307D:	loc_31306D: call test jg call jmp ; loc_31307D: call and	loc_31306D: call sub_3140F3 test eax, eax jg short loc_31307D call sub_3140F3 jmp short loc_31308C ;