				edi sub_314623	
				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]	
				edi	
				<pre>[ebp+arg_0], eax sub_31486A</pre>	
				eax, eax	
				short loc_31306D	
				<pre>eax, [ebp+arg_0]</pre>	
			push	eax	
	Miscellaneous C		push push call test jz	edi sub_314623 eax, eax short loc_31306D	
			cmp	[ebp+arg_0], esi	
	Madana Dinany Eve	لمها م	jz	short loc_31308F	
	Modern Binary Exp	lollal	ION	ODh	
				sub 31411B	
	CSCI 1968 - Sprin	σ 201			
	CSCI 4968 - Sprin	S AUL			
	Austin Dall			sub_3140F3	
	Austin Ralls	5		eax, eax short loc 31307D	
				sub 3140F3	
				short loc_31308C	
		loc_31307D:			
			call	sub_3140F3	
			and	eax, OFFFFh	
NADE 04/14/201E	Miccollongous Concente				1
MBE - 04/14/2015	Miscellaneous Concepts	loc 31308C+			CODE XREF: sub 312FD
MBE - 04/14/2015	Miscellaneous Concepts	loc_31308C:		[ebp+var_4], eax	1 CODE XREF: sub_312FE

Lecture Overview

Miscellaneous Concepts

Miscellaneous Concepts

- Integers in C
- Uninitialized data
- Structs

MBE - 04/14/2015

- File Descriptors
- Stack Cookies

		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]
	pusn	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
:_313066:		
		sub_31411B
_31306D:		
		sub_3140F3
		short loc_31307D
		sub_3140F3
		short loc_31308C
: 31307D:		
		sub_3140F3
	and	eax, OFFFFh
		2
313080		· CODE VDEE . and 210ED

Misc Concepts		call test jz cmp jnz mov cmp jb sub push	sub_314623 eax, eax short loc_31306 [ebp+arg_0], eb short loc_31306 eax, [ebp+var_7 eax, [ebp+var_8 short loc_31306 eax, [ebp+var_8 esi	x 6 0] 4] 6
 There's a lot of smaller bits class that are important, by 			sub_31486A	
warrant their own lectures			<pre>eax esi, 1D0h esi [ebp+arg_4] edi sub_314623 eax, eax short loc_31306 [ebp+arg_0], es short loc_31308</pre>	
	loc_313066:		0Dh sub_31411B	
	loc_31306D:		sub_3140F3 eax, eax short loc_31307 sub_3140F3 short loc_31308	
MBE - 04/14/2015 Miscellaneous Concepts	loc_31307D:	call and or	sub_3140F3 eax, 0FFFFh eax, 80070000h	; CODE XREF: sub_312FD8 3 ; CODE XREF: sub 312FD8

Misc Concepts

	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]
pusn	esi

- There's a lot of smaller bits and pieces to this class that are important, but too small to warrant their own lectures
- Also, this lecture should have come before spring break but got displaced till now

; CODE XREF: sub 312FD ; sub_312FD8+59

		loc_31306D:			eax loc_31307D 140F3	
		loc_31307D:		sub_3140F3		
MBE - 04/14/2015	Miscellaneous Concepts	loc 31308C:	or	eax, OFFFTh eax, 80070000h	4 ; CODE XREF: sub 312FD8	
				[ebp+var 4], eax		

Lecture Overview

Miscellaneous Concepts

- Integers in C
- Uninitialized data
- Structs
- File Descriptors
- Stack Cookies

		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
	pusn	esi
		[ebp+arg_0], eax
		sub_31486A
		eax, eax
		short loc_31306D
		eax, [ebp+arg_0]
		[ebp+arg_4] edi
		eul 21/622
		sub_314623
		eax, eax short loc 31306D
		[ebp+arg_0], esi short loc 31308F
		SHOLD TOC_313001
Loc 313066:		
		sub_31411B
Loc_31306D:		
		sub 3140F3
		eax, eax
		short loc_31307D
		sub_3140F3
		short loc 31308C
Loc 31307D:		
		sub 3140F3
	and	eax, OFFFFh
		5
Loc_31308C:		; CODE XREF: sub 312FD8

MBE - 04/14/2015

Integers in C

pushedicallsub_314623testeax, eaxjzshort loc_31306Dcmp[ebp+arg_0], ebxjnzshort loc_313066moveax, [ebp+var_70]cmpeax, [ebp+var_84]jbshort loc_313066subeax, [ebp+var_84]pushesi

• We haven't even mentioned signedness yet

int var1 = 0; unsigned int var2 = 0;

call	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, OFFFTh

Miscellaneous Concepts

Loc_31308C:

6 ; CODE XREF: sub Tebp+var 41, eax

Integers in C

• We haven't even mentioned signedness yet

int var1 = Θ ; unsigned int var2 = Θ ;

SUD_JI400M
short loc_31306D
eax, [ebp+arg_0]
[ebp+arg_4]
sub_314623
short loc_31306D
[ebp+arg_0], esi
short loc_31308F

What's the difference between an

int and an unsign	ed int	jg call jmp	sub_3140F3 eax, eax short loc_31307E sub_3140F3 short loc_313080	
	, loc_31307D:		sub_3140F3	
Miscellaneous Concepts		and or	eax, OFFFFh eax, 80070000h	7

Signed Integers		push call test jz cmp jnz mov cmp jb sub push	short loc_313066	Aub_314623 Hax, eax Hhort loc_31306D Hebp+arg_0], ebx Hhort loc_313066 Heax, [ebp+var_70] Heax, [ebp+var_84] Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat Heat	
 A signed integer can be ir 	nterpret	ed	as pos	sitive	
or negative			eax, eax short loc_31306 esi eax, [ebp+arg_0] eax esi, 1D0h		
 int range: -2,147,483,648 to 3 	2,147,483		esi [ebp+arg_4] edi sub_314623		
	loc_313066:		0Dh sub_31411B		
	loc_31306D: ;		sub_3140F3 eax, eax short loc_31307I sub_3140F3 short loc_313080		
	loc_31307D:	call and or	sub_3140F3 eax, 0FFFFh eax, 80070000h	; CODE XREF: sub_312FD8	
MBE - 04/14/2015 Miscellaneous Conc	epts loc_31308C:		[ebp+var_4], ear	8 ; CODE XREF: sub_312FD8	

Unsigned Integers

	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]
pusn	esi

An unsigned integer is only ever zero and up

• unsigned int - range: 0 to 4,294,967,295

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

		loc_31308C:		[ebp+var_4], eau	
MBE - 04/14/2015	Miscellaneous Concepts				9
			and	eax, OFFFFh	
				sub_3140F3	
		loc_31307D:			
				short loc_313080	
				sub_3140F3	
				short loc_313071	D
				eax, eax	
				sub_3140F3	

Unsigned Integers

	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]
pusn	631

An unsigned integer is only ever zero and up

 unsigned int - range: 0 to 4,294,967,295

		call	sub 31486A	
			short loc_31306	
			eax, [ebp+arg_0]	
signed int			[ebp+arg_4]	
			sub_314623	
naco 0 + a / 20/ 067 205				
ange: 0 to 4,294,967,295			short loc_31306	
			[ebp+arg_0], esi	
			short loc_313081	
	loc 313066:			
			sub_31411B	
	loc 31306 : •			
Twice the range of a sig	Jhed I	nte	oσer	
		call	970 3140F3	
			short loc 313071	

				short loc_3130	8C
		loc_31307D:			
				sub_3140F3	
			and	eax, OFFFFh	
MBE - 04/14/2015	Miscellaneous Concepts				10
		loc 31308C:			

Signedness Naming

MBE - 04/14/2015

	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]
pusn	esi

 The name signed or unsigned comes from whether or not the type can carry a sign (+/-)

			eax, [ebp+arg_0	
			[ebp+arg_4]	
			sub_314623	
			short loc_31306	D
			[ebp+arg_0], es.	
			short loc_31308	
	loc_313066:			
	loc 31306D:			
			sub 3140F3	
			short loc 31307	D
			short loc_31308	C
	loc_31307D:			
			sub_3140F3	
		and	eax, OFFFFh	
S				11
	loc_31308C:			
			Tehnatian Al ear	

Common Names

- Signed
 - int
 - signed int
 - long
- Unsigned
 - uint
 - unsigned int
 - unsigned long

		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	
	pusn	031	
		[ebp+arg_0], eax	
		sub_31486A	
		short loc_31306D	
		<pre>eax, [ebp+arg_0]</pre>	
		[ebp+arg_4]	
		sub_314623	
		short loc_31306D	
		<pre>[ebp+arg_0], esi</pre>	
		short loc_31308F	
loc 313066:			
		sub_31411B	
		bub_orarib	
loc 31306D:			
		sub_3140F3	
		short loc 31307D	
		sub 3140F3	
		short loc_31308C	
loc 31307D:			
		sub_3140F3	
	and	eax, OFFFFh	
			12

Miscellaneous Concept

oc_31308C:

; CODE XREI

Visualizing Sig	nednes	S	test jz cmp jnz mov cmp jb sub push	eax, eax short loc_31300 [ebp+arg_0], el short loc_31300 eax, [ebp+var_1 eax, [ebp+var_2 short loc_31300 eax, [ebp+var_1 esi	ox 56 70] 84] 56
• A signed int uses	the top hi	t to sr		ifv if i	is a
			call	sul 31486A	
positive or negat	ive numbe	r		short loc_3130	
$-0 \times 7FFFFFFF =$				eax, [ebp+arg_ eax	
• 0111111111			11	esi, 1D0h 1 ^{esi} [eto+arg_1]	1
		*****	push call	ed1 sub_314623	±
				eax, eax short loc_3130	
$-0 \times 80000000 =$	-2147483	647		<pre>[ebp+arg_0], es short loc_31308</pre>	31 DF
• 1 000000000	000000000	00000	900	00000	CODE XREF: sub 312FD8 sub 312FD8+55
			push call	0Dh sub_31411B	
		loc_31306D:			
$- 0 \times FFFFFFF =$	-1			sub_3140F3	
• <mark>1</mark> 1111111111	111111111	11111	11	1 1 1 1 1 1	
				short loc_3130	8C
		loc_31307D:	call	sub_3140F3	; CODE XREF: sub_312FD8
MBE - 04/14/2015	Miscellaneous Concepts			eax, 80070000h	12
WDL -04/14/2013		loc_31308C:		[ebp+var_4], ea	; CODE XREF: sub_312FD8

Two's Complement

- To make a number negative:
 Invert all bits
 Add 1

				short loc_31308	C.
		loc_31307D:			
				sub_3140F3	
			and	eax, OFFFFh	
MBE - 04/14/2015	Miscellaneous Concepts				14
		loc_31308C:			
				Tebp+var 41, ea	

Two's Complement

- To make a number negative:
 - Invert all bits
 - Add 1
- eg: 0x00031337
 - = 201527
 - = 00000000000000110001001100111
 - ~= 111111111111001110110011001000
 - **+**= 11111111111100110011001001
 - = -201527 (0xFFFCECC9)

	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]
pusn	031
	[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_312FD8+59

; CODE XREF: sub_312FD; ; sub_312FD8+49

		sub_3140F3 short loc 3130	
		SHOT C 100_3130	
loc_31307D:			
		sub_3140F3	
	and		

Miscellaneous Concepts

bc_31308C:					
- JT JUUC.		12		~	

; CODE XREF: su p+var 4], eax

Tracking S	ignedness			<pre>sub_314623 eax, eax short loc_3130 [ebp+arg_0], e short loc_3130 eax, [ebp+var_ eax, [ebp+var_ short loc_3130 eax, [ebp+var_ esi</pre>	bx 66 70] 84] 66
			pusn push push	esi eax	ר
How does you	our program tra	CK SIg	he	sub 31486A	a
				short loc_3130	
				esi	
				eax, [ebp+arg_ eax	0]
				[ebp+arg_4]	
				edi sub_314623	
				eax, eax	
				short loc_3130	
				[ebp+arg_0], e	
				short loc_3130	81
		loc_313066:			
				sub_31411B	
		loc_31306D:			
				sub_3140F3	
				short loc_3130	70
				sub_3140F3 short loc_3130	
		loc_31307D:			
		100-0100701		sub_3140F3	
			and	eax, OFFFFh	
MBE - 04/14/2015	Miscellaneous Concepts	100 212000.			16 ; CODE XREF: sub 312FD8
		loc_31308C:		[ebp+var_4], e	
				For Forder Till o	

Tracking Signedness

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

- How does your program track signed hess?

 Variable types are known at compile time, so signed instructions are compiled in to handle your variable
 with [ebp+arg_4]
 with [ebp+arg_4]
- You probably didn't realize this, but you can determine integer types at the assembly level

MBE 04/14/2013		loc_31308C:			
MBE - 04/14/2015	Miscellaneous Concepts				17
			and	eax, OFFFFh	
				sub_3140F3	
		loc_31307D:			
				sub_3140F3 short loc 3130	80
				short loc_3130	7D
				sub_3140F3	

Signed in	structions 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
 Some com 	mon signed instructions
- IDIV	- Signed divide
- IMUL	- Signed multiply
– SAL	- Shift left, preserve sign
– SAR	- Shift right, preserve sign
- MOVSX	- Move, sign extend
- JL	- Jump if less and 312FD8 + 56
- JLE	- Jump if less or equal
– JG	- Jump if greater
– JGE	- Jump if greater or equal
	jmp short loc_31308C ; loc_31307D: ; CODE XREF: sub_312FDs call sub_3140F3
MBE - 04/14/2015	and eax, OFFFFn or eax, 80070000h Miscellaneous Concepts loc_31308C: ; CODE XREF: sub_312FD2

Unsigned instructions

- MUL

– SHL

– SHR

-JB

-JA

- JAE

– JBE

- Some common unsigned instructions - DIV
 - Unsigned divide
 - Unsigned multiply
 - Shift left
 - Shift right
 - MOVZX Move, zero extend
 - Jump if below³¹³⁰⁶⁶
 - Jump if below or equal
 - Jump if above
 - Jump if above or equal

		loc_31307D:	307D:		
				sub_3140F3	
			and	eax, OFFFFh	
MBE - 04/14/2015	Miscellaneous Concepts				19
		loc_31308C:			

Minimum Size

- Minimum sizes

 char
 bits
 short
 bits
 int
 bits
 int
 bits
 call expenses
 bits
 call sub_314623
 cat expenses
 chore loc_313065
 cat expenses
 chore loc_313065
 cat expenses
 chore loc_313065
 coc_313065
 coc_313065
 coc_313065
 coc_313065
 - These are MINIMUM sizes, can vary from system to system!

		loc_31307D:			
				sub_3140F3	
			and	eax, OFFFFh	
/IBE - 04/14/2015	Miscellaneous Concepts				20
		loc_31308C:			

Fixed Sizes

- Fixed size format

 int[# of bits]_t
 uint[# of bits]_t
- eg int8_t, uint16_t, i
- Guaranteed size across systems

 Defined in stdint.h
 Also check out limits.h

		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	
	pusn	es1	
		[ebp+arg_0], eax	
		sub_31486A	
		short loc_31306D	
		<pre>eax, [ebp+arg_0]</pre>	
		[ebp+arg_4]	
		sub_314623	
.nt3	Test _	eax, eax	
NU 5.	JZ	short loc_31306D	
	спр	[ebp+arg_0], esi	
		short loc_31308F	
loc_313066:			
mc		ODh	
ems		sub_31411B	
00000			
loc_31306D:			
		sub_3140F3	
		eax, eax	
		short loc_31307D	
		sub_3140F3	
		short loc_31308C	
00 212070.			
Loc_31307D:			
		sub_3140F3 eax, OFFFTh	
		eax, 80070000h	
			21
Loc_31308C:			CODE XREF: sub 312FD8
		Lebotvar 41 ear	

push	edi
	sub_314623
	short loc_31306D
	[ebp+arg_0], ebx
	short loc_313066
	<pre>eax, [ebp+var_70]</pre>
	<pre>eax, [ebp+var_84]</pre>
	short loc_313066
	<pre>eax, [ebp+var_84]</pre>

- Imagine a simple uint8_t that is being ++'d
 - -0x00
 - 0x01
 - 0x02

call	31486A
	short loc_31306D
	eax, [ebp+arg_0]
	[ebp+arg_4]
	sub_314623
	short loc_31306D
	[ebp+arg_0], esi
	short loc_31308F

TO	G.		U		

	3141	

300_3140F3
short loc_31307D
sub_3140F3
short loc_31308C

		U	1	

22

MBE - 04/14/2015

push	edi
	sub_314623
	short loc_31306D
	<pre>[ebp+arg_0], ebx</pre>
	short loc_313066
	<pre>eax, [ebp+var_70]</pre>
	<pre>eax, [ebp+var_84]</pre>
	short loc_313066
	<pre>eax, [ebp+var_84]</pre>
	A21

- Imagine a simple uint8_t that is being ++'d
 - 0x00
 - 0x01
 - 0x02
 - ••
 - Øxfe
 - 0xFF - ????

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

loc_313066

push

; CODE XREF: sub 312FD ; sub_312FD8+59

call sub_31

; CODE .

; sub 312FD8+49

call sub_3140F3 test eax, eax ig short loc_31307D call sub_3140F3

jmp short loc

307D:

CODE XREF: sub_312FD8

F3 FFh 70000h 23

MBE - 04/14/2015

Miscellaneous Concepts

oc_31308C:

; COl

push	edi
	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]

- Imagine a simple uint8 t that is being ++'d ightarrow
 - -0000
 - -0x01
 - 0x02
 - • •
 - -0xFE
 - 0xFF
 - 0x00 <-- overflows!
 - -0x01

:all	31486A
	short loc_31306D
	eax, [ebp+arg_0]
	[ebp+arg_4]
	sub_314623
	short loc_31306D
	[ebp+arg_0], esi
	short loc 31308F

	3141

- sub 3140F3

|--|

Cax,	

		U	С	

ebp+var	41.

- pushedicallsub_314623testeax, eaxjzshort loc_31306Dcmp[ebp+arg_0], ebxjnzshort loc_313066moveax, [ebp+var_70]cmpeax, [ebp+var_84]jbshort loc_313066subeax, [ebp+var_84]pushesi
- This obviously applies to any size of integer!
 - ØxFFFFFFD
 - ØxFFFFFFF
 ØxFFFFFFFF
 - 0x0000000
 - 0x00000001
 - 0x0000002

call	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	
	eax, 80070000h	

Miscellaneous Concepts

oc_31308C:

; CODE ;

25

- Don't forget multiplying!
 0x00120000 * 0x00123456
 - $= 0 \times 00000147 AE0C0000$
 - $= 0 \times AE0C0000 (long)$

		26
	and	eax, OFFFTh
7D:	call	; CODE XREF: sub_312F sub 3140F3
		short loc 31308C
		sub_3140F3
		short loc_31307D
		sub 3140F3
6D:		
		sub_31411B
66:		
		short loc 31308F
		[ebp+arg_0], esi
		short loc_31306D
		eax, eax
		sub_314623
		edi
		[ebp+arg_4]
	push	esi
		earling
b 10	lea	eax. [ebp+arg_0]
		esi
0		short loc_31306D
6		eax, eax
		sub_31486A
		[ebp+arg_0], eax
		edi Tehntarg Ol eav
	pusn	051
	push	esi
		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
		1 04 1 500

MBE - 04/14/2015

Miscellaneous Concepts

Loc_31308C:

; CODE : [ebp+var 4], eax

- Don't forget multiplying! 0x00120000 * 0x00123456
 - $= 0 \times 00000 147 AE0 C0000 (lo$
 - 0xAE0C0000 (long)

or

0x40000123 *

- 0x0000048C (long)

	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]
pusn	esi
	[ebp+arg_0], eax
	sub_31486A
	short loc_31306D
	eax_ [ebp+arg_0]
mc v	eax long
push	esi
	[ebp+arg_4]
	sub_314623
	short loc_31306D
	[ebp+arg_0], esi

sub		

sub 3140F3

sub 3140F3

27

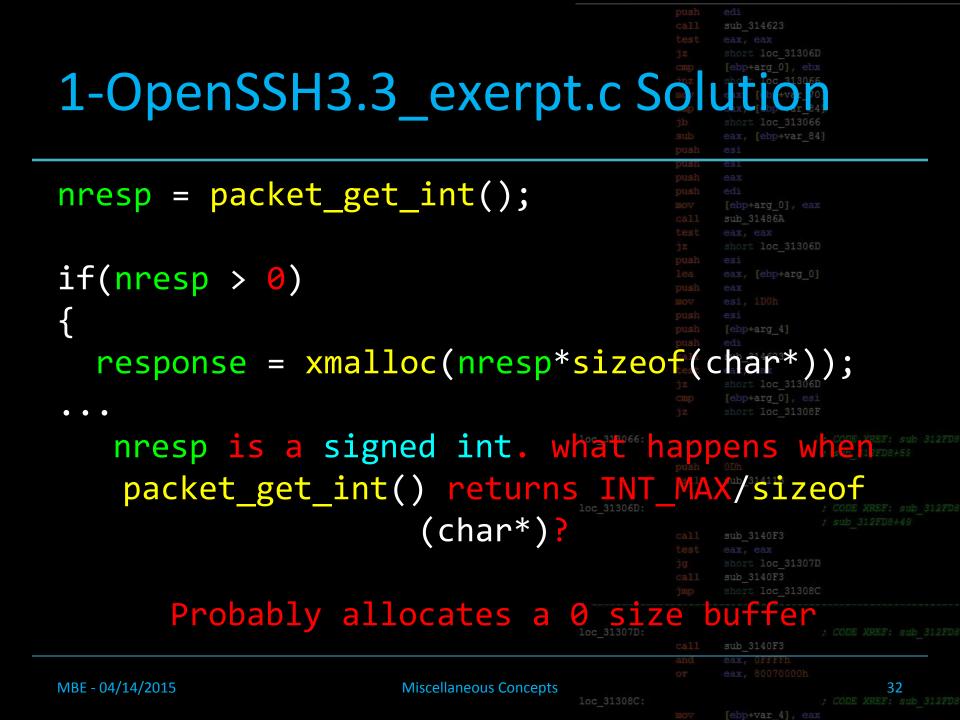
MBE - 04/14/2015





		edi sub 314623	
		short loc 31306	
		[ebp+arg_0], eb:	
1-OpenSSH3.3_exerp	jnz	short loc_31306	
I-UUPUDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD	mov	eax, [ebp+var_7	
	Спр	eax, [ebp+var_8	
		short loc_31306 eax, [ebp+var_8	
		esi	±]
	pusn	esi	
<pre>nresp = packet_get_int();</pre>		edi [ebp+arg 0], ea	
		sub 31486A	
		eax, eax	
		short loc 31306	
if(nresp > 0)		eax, [ebp+arg_0]	
ſ			
~		esi John Jang 41	
		[ebp+arg_4] edi	
<pre>response = xmalloc(nresp*</pre>	iz	char*));
<pre>for (i = 0; i < nresp; i+</pre>	-+) ^{cmp} jz	<pre>[ebp+arg_0], es. short loc_31308</pre>	
			CODE VOEE,
<pre>response[i] = packet_ge</pre>	et strin	g(NULL	; sub 312FD8+59
	push	ODa	/)-
		sub_31411B	
(https://www.owasp.org/index.php	o/Integer_over		
		eax, eax	
		short loc_31307 sub 3140F3	
		short loc_31308	
	loc_31307D:		
		sub_3140F3	
	and	eax, OFFFFh	
MBE - 04/14/2015 Miscellaneous Concepts	loc_31308C:		30 ; CODE XREF: sub_312FD8

1-OpenSSH3.3_e	xerpt.c S	<pre>push edi call sub_314623 test eax, eax jz short loc_31306D cmp [ebp+arg_0], ebx cmp [ebp</pre>	
<pre>nresp = packet_get_in</pre>	t();	push esi push eax push edi mov [ebp+arg_0], eax call sub_31486A test eax, eax jz short loc 31306D	
<pre>if(nresp > 0) { response = xmalloc(</pre>	<pre>nresp*sized</pre>	<pre>push esi lea eax, [ebp+arg_0] push eax mov esi, 1D0h push esi push [ebp+arg_4] push edi</pre>	
<pre>nresp is a signed packet_get_int() (c</pre>		push * ODh MAX/Silzeof ; CODE XREF: sub_3. ; sub_312FD8+49 call sub_3140F3 test eax, eax jg short loc_31307D call sub_3140F3 jmp short loc_31308C ; CODE XREF: sub_3.	
MBE - 04/14/2015 Miscella	neous Concepts	call sub_3140F3 and eax, 0FFFh or eax, 80070000h 31 ; CODE XREF: sub_3 mov [ebp+var_4], eax	



2-variable-length_exe	erpt.	jb sub push	edi sub_314623 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 esi	
<pre>char* processNext(char* strm) char buf[512]; short len = *(short*) strm; strm += sizeof(len); if (len <= 512) { memcpy(buf, strm, len); process(buf); } }</pre>	<pre>{ loc 313066:</pre>	push push mov call test jz push lea push lea push push push call test jz cmp jz	esi eax edi [ebp+arg_0], eax sub_31486A eax, eax short loc_31306D esi eax, [ebp+arg_0] eax esi, 1D0h esi [ebp+arg_4] edi sub_314623 eax, eax short loc_31306D [ebp+arg_0], esi short loc_31308F	
return strm + len;	100_313066;			
<pre>} else { return -1; }</pre>	loc_31306D:			
} (https://www.owasp.org/index.php/Ir	teger_ove			
MBE - 04/14/2015 Miscellaneous Concepts	loc_31308C:	call and or mov	<pre>sub_3140F3 eax, 0FFFFh eax, 80070000h [ebp+var_4], eax</pre>	33 ; CODE XREF: sub_312FD8

2-variable-length_exe	erpt.o	push call test jz cmp inz mov 2p jb sub push	short loc_313066 eax, [ebp+var_84] esi
<pre>char* processNext(char* strm) char buf[512]; short len = *(short*) strm; strm += sizeof(len); if (len <= 512) { memcpy(buf, strm, len); } }</pre>	ł		esi eax edi [ebp+arg_0], eax sub_31486A eax, eax short loc_31306D esi eax, [ebp+arg_0] eax esi, 1D0h esi [ebp+arg_4] edi sub_314623 eax, eax short loc_31306D [ebp+arg_0], esi short loc_31308F
len is signed short, and	10c_313066: NY Neg	at	; CODE XREF: sub 312FD8 ive len ^{b_312FD8+55}
will pass the if-	staten		sub 31411B
		call and	; CODE XREF: sub_312FD8 sub_3140F3 eax, OFFFTh
MBE - 04/14/2015 Miscellaneous Concepts	loc_31308C:		eax, 80070000h 34 ; CODE XREF: sub_312FD8

2-variable-	length_exerpt	jb short loc_313066 sub eax, [ebp+var_84] push esi
<pre>char buf[512] short len = * strm += sizeo if (len <= 51</pre>	<pre>(short*) strm; f(len);</pre>	<pre>push esi push eax push edi mov [ebp+arg_0], eax call sub_31486A test eax, eax jz short loc_31306D push esi lea eax, [ebp+arg_0] push eax mov esi, 1D0h push esi push [ebp+arg_4] push edi call sub_314623 test eax, eax jz short loc_31306D cmp [ebp+arg_0], esi jz short loc_31308F</pre>
	loc_313066	56: ; CODE XREF: sub 312FD8
	gned short, any n	Call Sub_31411B
	pass the if-stat	<pre>call sub_3140F3 test eax, eax jg short loc_31307D</pre>
•	y takes an unsigr	
underflow ->	large copy ->s.tr	call sub_3140F3
MBE - 04/14/2015	Miscellaneous Concepts	and eax, 0FFFFh or eax, 80070000h 35 cc: ; CODE XREF: sub_312FD8 mov [ebp+var_4], eax

Integer Problems

MBE - 04/14/20

	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]
pusn	631

- It's very common to see modern bugs stem from integer confusion and misuse
- Know when to use signed/unsigned

call	sub_31486A
se	short loc_31306D
push	
	eax, [ebp+arg_0]
push	[ebp+arg 4]
	edi
call	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
	36

015	Miscellaneous Concepts		
		loc_31308C:	



Miscellaneous Concepts

Miscellaneous Concepts

- Integers in C
- Uninitialized data
- Structs

- File Descriptors
- Stack Cookies

		sub_314623	
		short loc 31306	D
		[ebp+arg_0], eb	
		short loc 31306	
		eax, [ebp+var_7	
		eax, [ebp+var_8	
		short loc_31306	
		eax, [ebp+var 8	
	pusn	es1	
		[ebp+arg_0], ear	
		sub_31486A	
		eax, eax	
		short loc 31306	D
		esi	
		eax, [ebp+arg_0]	
		eax [coprarg_o	
		[ebp+arg_4]	
		edi	
		sub_314623	
		short loc_31306	
		[ebp+arg_0], es	
		short loc_31308	
22 2120551			
oc_313066:			
		sub_31411B	
oc_31306D:			
.00_31306D:			
		sub_3140F3	
		short loc_31307	0
		sub_3140F3	
		short loc_31308	
oc 31307D:			
		sub 3140F3	
	and	eax, OFFFFh	
			37
oc 31308C:			; CODE XREF: sub 312FD8

Uninitialized Data

	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]
pusn	esi

 Uninitialized data is a subtle vulnerability that can leak information or cause undefined behavior in an application

MBE - 04/14/2015	Miscellaneous Concepts	loc_31308C:		[ebp+var_4], eax	38 : CODE XREF: sub_312FD8
			or	eax, OFFFFh eax, 80070000h	
		loc_31307D:		sub_3140F3	
				sub_3140F3 eax, eax short loc_31307D sub_3140F3 short loc_31308C	

Uninitialized Data

	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]
pusn	esi

- Uninitialized data is a subtle vulnerability that can leak information or cause undefined behavior in an application
- The bug manifests when variables are not properly initialized before use

		loc_31308C:		[ebp+var 4], ea	
MBE - 04/14/2015	Miscellaneous Concepts				39
			and	eax, OFFFFh	
				sub_3140F3	
		loc_31307D:			
				short loc_31308	
				sub_3140F3	
				short loc 31307)
				eax, eax	
				sub_3140F3	
		loc_31306D:			
		1 2120CD-			
				sub_31411B	

Spot the Bug

```
int do_work()
{
    int i;
    char buf[20];
    while(i < 20){
        buf[i] = `A`;
        i++;</pre>
```

```
return 0;
```

		sub 314623
		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]
	pusn	-CS1
		[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
3066:		
		sub_31411B
306D:		
		sub_3140F3
		short loc_31307D
		sub_3140F3
		short loc_31308C
307D:		
	call	sub_3140F3
	and	eax, OFFFTh eax, 80070000h
0000-		40 CODE XREF: sub 312FD8
308C:		

Μ	ΒE	- 04	1/1	4/	20)15

}

Miscellaneous Concepts

loc_31308C:

; CODE XRE

				sub_314623	
				short loc_31306	
				[ebp+arg_0], ebs	
Spot the Bug				short loc_313060	
				eax, [ebp+var_70	
Spot the Dug				eax, [ebp+var_84	1]
				short loc 313060	
				eax, [ebp+var_84	
			pusn	esi	
<pre>int do work()</pre>					
				[ebp+arg_0], eau	
				sub 31486A	
{					
				short loc 31306	
	ic p	ovori	niti		
int i; 🗕 🚽 🚽				Cerr, Lever 0	
char buf[20];	•		mov		
	T	o anyt		Csi	
			push	Cabp+arg 41	
				sub_314623	
while(i < 20){				short loc_31306	2
				[ebp+arg_0], esi	
				short loc_313081	
buf[i] = 'A';					
		loc 313066:			
÷					
i++;					
				sub 31411B	
∫		loc 31306D:			
				sub 3140F3	
				short loc 313071	
return 0;				sub 3140F3	
				short loc_313080	
		loc_31307D:			
				sub 3140F3	
			and	eax, OFFFFh	
MBE - 04/14/2015	Miscellaneous Concepts				41
	miseenaneous concepts	loc_31308C:			; CODE XREF: sub 312FD8
				Tebp+var 41, eau	

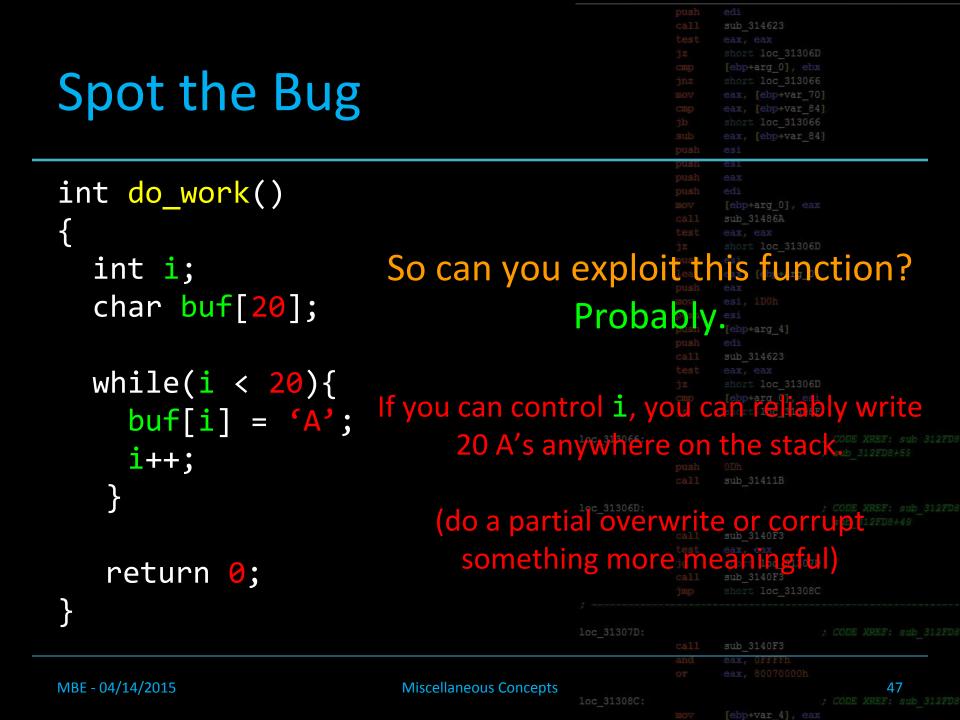
Spot the Bug int do_work() $\{$ short loc_31306D i is never initialized. int i; char buf[20]; to anything ai while(i < 20){ So what is buf[i] = 'A'; i++; return ∅; MBE - 04/14/2015 Miscellaneous Concepts 42

Spot the Bug int do_work() sub 31486A $\{$ i is never initialized int i; char buf[20]; to anythe gest, 100h push gest arg_41 while(i < 20){ So what is i? buf[i] = 'A'; The variable will be whatever data i++; happens to be left on the stack frame from a previous function call of any sort return ∂; MBE - 04/14/2015 Miscellaneous Concepts 43

Spot the Bug			push call test jz cmp jnz mov cmp jb sub push	edi sub_314623 eax, eax short loc_31306 [ebp+arg_0], eb short loc_31306 eax, [ebp+var_7 eax, [ebp+var_8 short loc_31306 eax, [ebp+var_8 esi esi	ж 6 0] 4] 6
<pre>int do_work() { int i;</pre>	So can you	ovnloi	push push mov call test jz pust	eax edi [ebp+arg_0], ea sub_31486A eax, eax short loc 31306	D
char buf[20];	So carryou	Схрю		esi, 1D0h esi [ebp+arg_4] edi	
<pre>while(i < 20){ buf[i] = `A';</pre>		lan adama		<pre>sub_314623 eax, eax short loc_31306 [ebp+arg_0], es short loc_31308</pre>	
i++; }		loc_313066: loc_31306D:		ODh sub_31411B	
return 0;				<pre>sub_3140F3 eax, eax short loc_31307 sub_3140F3 short loc_31308</pre>	; sub_312FD8+49
}		;; loc_31307D:	call and or	sub_3140F3 eax, 0FFFFh eax, 80070000h	; CODE XREF: sub_312FD8
MBE - 04/14/2015	Miscellaneous Concepts	loc_31308C:		Tohn your Al	44 ; CODE XREF: sub_312FD8

			push	edi	
				sub_314623	
				short loc_31306D	
				[ebp+arg_0], ebx	
Spot the Bug				short loc_313066	
Shot the Rild				eax, [ebp+var_70	
Spot the Dug				eax, [ebp+var_84	
				short loc_313066	
				eax, [ebp+var_84	
			pusn	esi	
int do work()					
				[ebp+arg_0], eax	
ſ				sub_31486A	
{					
	-	•	jz	short loc_31306D	
int i;		ovnloi	pust	nic fun	-tion?
و⊥ ااا⊥	So can you	CAPIO	lea		
	•			Cd.X.	
char <pre>buf[20];</pre>			BOY	esi, 1D0h	
		Proba		esi	
			pdBd		
				sub_314623	
while (i < 20)					
while(i < 20){				short loc_31306D	
· · · · ·				[ebp+arg_0], esi	
buf[i] = ' A';				short loc_31308F	
		loc 313066:			
•					
<u>i</u> ++;					
_ · · J				sub 31411B	
٦				oup_orario	
Ĵ		loc_31306D:			
				sub_3140F3 eax, eax	
				short loc 31307D	
return 0;				sub 3140F3	
				short loc 31308C	
		loc_31307D:			
				sub 3140F3	
			and	eax, OFFFFh	
MBE - 04/14/2015	Miscellaneous Concepts				45
		loc_31308C:			; CODE XREF: sub 312FD8
				Tebruar /1 ear	

				sub_314623	
				short loc_3130	6D
				[ebp+arg_0], el	
Coot the Dug				short loc_3130	66
Spot the Bug				eax, [ebp+var_]	
				<pre>eax, [ebp+var_]</pre>	
•				short loc_3130	
				eax, [ebp+var_	84]
			push	esi	
			pusn	es1	
int do work()					
				[ebp+arg_0], ea	
5				sub_31486A	
					(D)
	C]2	short loc_3130	
int i;	So can you	explo	Los	his tiin	rtion r
		CAPIO	nush		
char buf[20];		Droh	h	esi	
		Proba		·Tebp+arg_41	
			push	edi	
				sub 314623	
while(i < 20){				short loc 3130	
• • •			CIIID	[ebp+arg [], e	
	If you can cont	roi L, y		Caldrt 162 16	
<pre>buf[i] = 'A';</pre>					
	20 A's ar	1/19-31-3955 rd		the sta	CODE XREF: sub 312FD:
i++;				i the sta	312FD8+59
و ۱۱ ۲					
				sub_31411B	
>					
		loc_31306D:			
				sub_3140F3	
				eax, eax	
return 0;				short loc_3130 sub 3140F3	
				short loc 3130	
				51010 100_3130	
		loc_31307D:			
		100_010000		sub 3140F3	
				eax, OFFFFh	
MBE - 04/14/2015	Miscellaneous Concepts				46
	wiscenarieous concepts	loc 31308C:			; CODE XREF: sub 312FD:
				Tebp+var 41. e	



More Subtle

}

MBE - 04/14/2015

```
#include <stdio.h>
#include <stdlib.h>
void take_ptr( int *bptr ) {
    printf( "%lx", *bptr );
}
int main( int argc, char **argv.)
    int b;
    take_ptr( &b );
    printf( "%lx", b );
```

(https://www.blackhat.com/presentations/bh-europe-06/bh-eu-06-Flake.pdf)

Miscellaneous Concepts

48

#-uninitialized_data.c

- on warzone
- http://www.exploit-db.com/docs/99.pdf

MBE - 04/14/2015 M	liscellaneous Concepts	loc_31308C:		[ebp+var_4], eax	49 ; CODE XREF: sub_312FD8
			and	eax, OFFFFh eax, 80070000h	
		loc_31307D:		sub_3140F3	
				<pre>sub_3140F3 eax, eax short loc_31307D sub_3140F3 short loc_31308C</pre>	

#-uninitialized_data.c - Solution

- char *err, *mesg;
- easy to exploit with ASLR off

		short loc_31306D	
	cmb	[ebp+arg_0], ebx	
		thent loc 313066	
	blu	ax. [eb + ar 70]	
		short loc_313066	
		eax, [ebp+var_84]	
	pusn	es1	
		[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	
313066:			
		sub 31411B	
		500_JITID	
31306D:			
313000.			
		sub 3140F3	
	test		
		eax, eax short loc 31307D	
		sub 3140F3	
		short loc 31308C	
		SHOLF TOC_212080	
31307D:			
	call	sub 3140F3	
	and	eax, OFFFTh	
			50

MBE - 04/14/2015

Miscellaneous Concepts

Loc_31308C:

; CODE

Uninitialized Data

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

sub 3140F3

- Keep in mind this can happen on the heap too!
- There's no knowing what's going to be on the other end of the pointer you get back from something like malloc()

				short loc_31307D sub_3140F3 short loc_31308C	
		, loc_31307D:	_31307D:	sub_3140F3	
MBE - 04/14/2015	Miscellaneous Concepts	loc_31308C:	and or mov		51 ; CODE XREF: sub_312FD8

Uninitialized Data

	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]
pusn	031

- Pretty common in amateur development, smaller software projects, CTF problems
- Less common in industry as this is an easy issue to detect statically (in source and binary)

MBE - 04/14/2015	Miscellaneous Concepts	loc_31308C:		[ebp+var_4], ead	52 ; CODE XREF: sub_312FD: x
				eax, 0FFFFh eax, 80070000h	
			call	sub_3140F3	_
		loc_31307D:			
				eax, eax short loc_31307 sub_3140F3 short loc_31308	
		loc_31306D:		sub_3140F3	
				0Dh. sub_31411B	

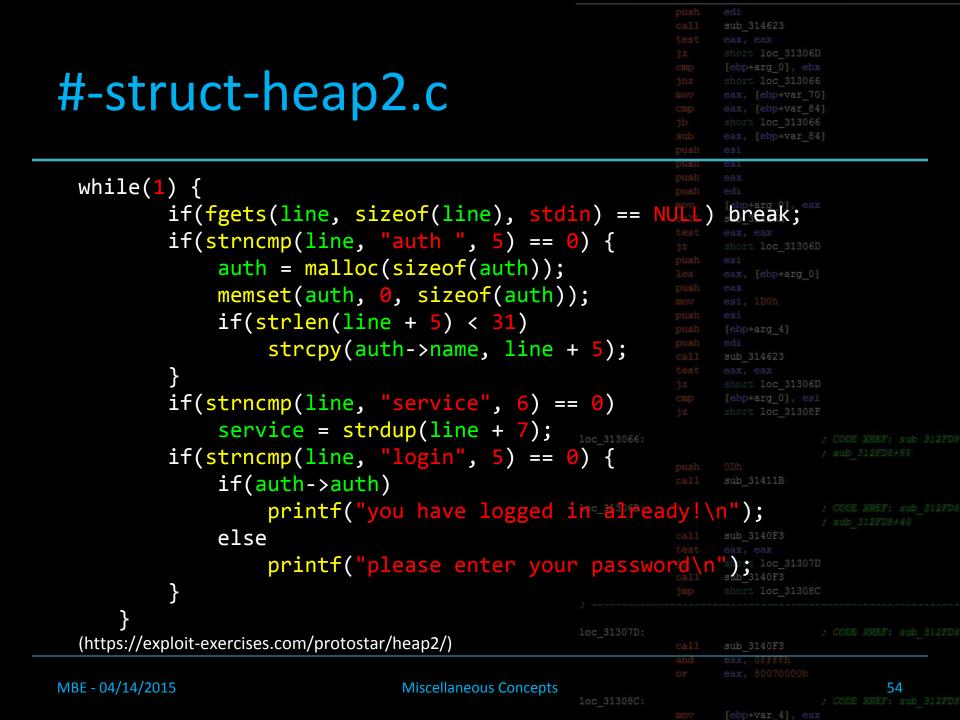
Miscellaneous Concepts

Miscellaneous Concepts

- Integers in C
- Uninitialized data
- Structs

- File Descriptors
- Stack Cookies

		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]	
	pusn	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	
loc 313066:			
		sub 31411B	
loc_31306D:			
		sub_3140F3	
		short loc_31307D	
		sub_3140F3	
		short loc_31308C	
Loc_31307D:			
		sub_3140F3	
	and	eax, OFFFFh	
			53
loc_31308C:			
		Tebratian Al apar	



	push call test jz	edi sub_314623 eax, eax short loc_31306D
#-struct-heap2.c		[ebp+arg_0], ebx short loc_313066 eax, [ebp+var_70]
		eax, [ebp+var_84] short loc_313066 eax, [ebp+var_84]
	push pusn	esi esi
• • •		
if(strncmp(line, "auth ", 5) == 0) {		[ebp+arg_0], eax sub 31486A
<pre>auth = malloc(sizeof(auth));</pre>		short loc_31306D
<pre>memset(auth, 0, sizeof(auth));</pre>		esi eax, [ebp+arg 0]
if(strlen(line + 5) < 31)		
<pre>strcpy(auth->name, line + 5);</pre>		esi [ebp+arg 4]
}		sub_314623
if(strncmp(line, "service", 6) == 0)		eax, eax short loc 31306D
		[ebp+arg_0], esi
<pre>service = strdup(line + 7);</pre>		short loc_31308F
• • •		
 sizeof(auth) doesn't return siz 	e o	fstruct
 In this case, it returns 16 		; CODE XREF: sub_312F ; sub_312FD8+49 sub_3140F3
 service makes new buffer over 	na jmp	
 Use sizeof(struct auth) 	call	; CODE XREF: sub_3127 sub_3140F3
	or	eax, OFFFTh eax, 80070000h
MBE - 04/14/2015 Miscellaneous Concepts		55
loc_31308C:		; CODE XREF: sub_312F [ebp+var_4], eax

Miscellaneous Concepts

Miscellaneous Concepts

- Integers in C
- Uninitialized data
- Structs

- File Descriptors
- Stack Cookies

		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]	
	pusn	esi	
		[ebp+arg_0], eax	
		sub_31486A	
		short loc_31306D	
		eax, [ebp+arg_0]	
		[ebp+arg_4]	
		sub_314623	
		short loc_31306D	
		<pre>[ebp+arg_0], esi</pre>	
		short loc_31308F	
100 2120551			
loc_313066:			
		sub 31411B	
loc_31306D:			
		sub 3140F3	
		eax, eax	
		short loc_31307D	
		sub_3140F3	
		short loc 31308C	
loc 31307D:			
		sub 3140F3	
	and	eax, OFFFFh	
			56
loc_31308C:			CODE XREF: sub 312FD8

File Descriptors

- In Linux, everything is a file
- When opening a file, it gets a number
- You use some frequently
 - 0 STDIN
 - 1 STDOUT
 - 2 STDERR
- open returns a file descriptor

	push	edi	
		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]	
	pusn	esi	
		[ebp+arg 0], eax	
		sub 31486A	
		short loc 31306D	
	area in	est	
a nu			
	push	leax	
		[ebp+arg_4]	
		sub_314623	
		short loc_31306D	
		[ebp+arg_0], esi	
		short loc 31308F	
.oc_313066;			
		sub_31411B	
.oc_31306D:			
r			
ノ		sub_3140F3	
	test		
		short loc_31307D	
		sub 3140F3	
		short loc_31308C	
.oc_31307D:			
		sub_3140F3	
	and	eax, OFFFFh	
			57

MBE - 04/14/20	1	[
----------------	---	---

Miscellaneous Concepts

loc_31308C:

Tebr+var 41, eax

#-fd.c

• On the warzone

		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]	
	pusn	esi	
		[ebp+arg 0], eax	
		sub_31486A	
		short loc 31306D	
		eax, [ebp+arg_0]	
		eax	
		[ebp+arg_4]	
		edi	
		sub_314623	
		short loc_31306D	
		[ebp+arg_0], esi	
		short loc_31308F	
13066:			
		sub_31411B	
12060-			
1306D:			
		sub_3140F3	
		eax, eax	
		short loc_31307D	
		sub_3140F3	
		short loc_31308C	
1307D:			
	call	sub 3140F3	
		-	
		eax, OFFFFh eax, 80070000h	
		our, ocovououn	58
			58 CODE XREF: sub 312FD8
1308C:			

MBE - 04/14/2015

Miscellaneous Concepts

#-fd.c - Solution

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

file descriptors from parent processes are inherited by children

fd to password file wasn't closed

	ses al e
st	eax, eax
	short loc_31306D
	eax, [ebp+arg_0]
	[ebp+arg_4]
	sub_314623
	short loc_31306D
	[ebp+arg_0], esi
	short loc_31308F

				short loc_31307D sub_3140F3 short loc_31308C	
		loc_31307D:)7D: call	sub_3140F3	
			and	eax, OFFFFh	
MBE - 04/14/2015	Miscellaneous Concepts	loc_31308C:			59 : CODE XREF: sub_312FD8

Miscellaneous Concepts

Miscellaneous Concepts

- Integers in C
- Uninitialized data
- Structs

- File Descriptors
- Stack Cookies

	push	edi
		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]
	pusn	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
loc 313066:		
106_313000.		
		Sub Statio
loc_31306D:		
		sub 3140F3
		eax, eax
		short loc 31307D
		sub_3140F3
		short loc 31308C
loc_31307D:		
		sub_3140F3
	and	eax, OFFFFh
		60
loc_31308C:		

			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	
			eax, eax	
			short loc 31306D	
			eax, [ebp+arg_0]	
Stack Canarie				
JIALN CAHAIR				
	р		[ebp+arg_4]	
			sub_314623	
	j	Z	short loc_31306D	
		щp	[ebp+arg_0], esi	
			short loc_31308F	
Modern Binary Exploi	tati			
	lall	JU		
			0Dh sub 31411B	
CCCL 10CQ Coving 2			300_314110	
CSCI 4968 - Spring 2	CLU.)		
			sub_3140F3	
Sophia D'Antoine				
			short loc_31307D	
			sub_3140F3	
			short loc_31308C	
loc_313				
		all	sub_3140F3	
		nd	eax, OFFFFh	
				C1
MBE - 04/14/2015 Stack Canaries				61
loc_313				
			[ebp+var_4], eax	

1. How do we protect against overflows?

Stack Canaries

- 2. Different Types
- 3. Guarding the Stack
- 4. Ways to Leak Information
- 5. When All Else Fails

		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]	
	pusn	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	
_313066:			
		0Dh	
		sub_31411B	
2120 CD+			
_31306D:			
		sub 3140F3	
	call test		
		eax, eax	
		short loc_31307D sub 3140F3	
		short loc 31308C	
31307D:			
		sub 3140F3	
	and	eax, OFFFFh	
		62	
31308C:		; CODE XREF: sub 3	

Overflow Protections

Before the Overflow (program and compiler)

- program well
- validate input
- static/ dynamic analysis

After the Overflow (OS level)

- intercept function calls
- turn off execution
- randomize the addresses

		sub_314623	
		short loc_3130	
		[ebp+arg_0], @	
		short loc_3130	
		eax, [ebp+var]	
		eax, [ebp+var]	_84]
		short loc_3130	066
		eax, [ebp+var]	_84]
	push	esi	
	pusn	esi	
		[ebp+arg_0], (
piler)		sub_31486A	
	Jz	short loc_3130	160
r <mark>cpy v</mark> s	strno	c <mark>py v st</mark> i	VQJI
SCII			
		esi Lohmong 41	
VM or 3	ςΔτ	Solvers	
		sub 314623	
		eax, eax	
		short loc 3130	
		[ebp+arg_0], (
		short loc_3130	
loc 313066:	-		
nk Libsa	مfد		
	push		
		sub 31411B	
(bit / L)FP		
{ 			
SLR		sub 3140F3	
		short loc_3130	07D
		sub_3140F3	
		short loc_3130	08C
4			
g/wiki/Libsa	ate		
loc_31307D:			
		sub_3140F3	
	- Devel	ABY OFFF	

Stack Canaries

Avaya Labs \Rightarrow http://directory.fsf.org

loc_31308C:

st

Lir

N)

AS

; CODE XREI ebo+var 41. eax

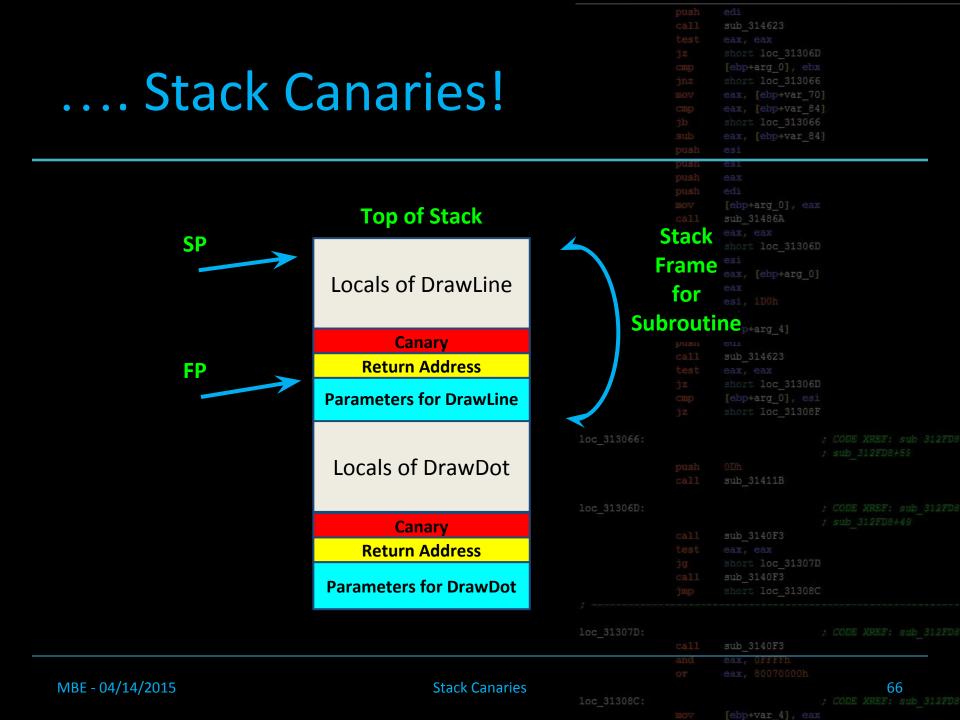


.... Stack Canaries!

- + After the Overflow (Compiler and OS level)
 - sometimes called Stack Guards or Cookies
 - embed random "canaries" in stack frames
 - verify their integrity PRIOR to Function RET

		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]
	pusn	esi
-		[ebp+arg_0], eax
el)		sub_31486A
CIJ		
		short loc_31306D
		eax, [ebp+arg_0]
	push	
ΟΚ	ies	
		[ebp+arg_4]
		sub_314623
an	nes	
an	IC3	short loc_31306D
		[ebp+arg_0], esi
		short loc_31308F
n	RFT	URN ; sub_312ED8+59
	push	
		sub 31411B
D:		
		sub_3140F3

				eax, eax short loc_313071 sub_3140F3 short loc_313080	
		loc_31307D:		sub 3140F3	
			and or	eax, OFFFFh eax, 80070000h	
MBE - 04/14/2015	Stack Canaries	loc_31308C:		Tehnisten /1 eas	65 ; CODE XREF: sub_312FD



.... Stack Canaries!

+ What is a canary?

MBE - (

- its a random integer
- pushed onto stack after certain triggers are pushed
- popped off stack and checked before the trigger is read from
- valued saved as global variable padded by unmapped pages

14/2015	Stack Canaries	loc_31308C:		[ebp+var 4], ea	67 ; CODE XREF: sub_312FD8
				eax, OFFFFh eax, 80070000h	67
			call	sub_3140F3	
		loc_31307D:			
				short loc_31307 sub_3140F3 short loc_31308	
				sub_3140F3 eax, eax	
		loc_31306D:			
pages				0Dh sub_31411B	

sub 31486A

.... Stack Canaries!

Drawbacks ╋

MBE - 04

- adds overhead (huge cache footprint) 1.
- only defends against stack overflows 2.
- NULL canaries can potentially be abused 3.
- Random canaries can potentially be learned 4.
 - format string vulns a.

h	information leak				
Ν.	mormation icak				
				sub_31411B	
		loc_31306D:			
				sub 3140F3	
				short loc 313071	
				sub_3140F3	
				short loc_313080	
		loc 31307D:			
				sub_3140F3	
			and	eax, OFFFFh	
/14/2015	Stack Canaries	loc_31308C:			68 ; CODE XREF: sub 312FD8
				[ebp+var 4], eas	

1. How do we protect against overflows?

Stack Canaries

- 2. Different Types
- 3. Guarding the Stack
- 4. Ways to Leak Information
- 5. When All Else Fails

		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
	pusn	esi
2		
?		[ebp+arg_0], eax
		sub_31486A
		eax, eax
		short loc_31306D
		eax, [ebp+arg_0]
		esi Labatang (1
		[ebp+arg_4]
		edi sub 314623
		sub_314623
		eax, eax
		<pre>short loc_31306D [ebp+arg_0], esi</pre>
		short loc_31308F
		20016 100 210001
oc 313066:		
00_010000		
		sub_31411B
oc_31306D:		
		sub 3140F3
		short loc_31307D
		sub_3140F3
		short loc 31308C
oc 31307D:		
		sub 3140F3
	and	eax, OFFFFh
		69
oc_31308C:		; CODE XREF: sub 312FD8

Terminato	or Canaries		push call test jz cmp jnz mov cmp jb sub sub	edi sub_314623 eax, eax short loc_313066 [ebp+arg_0], ebx short loc_313066 eax, [ebp+var_70 eax, [ebp+var_84 short loc_313066 eax, [ebp+var_84 esi	6 5 0] 1] 5
				esi eax edi [ebp+arg_0], eax sub_31486A eax, eax	
The Canary = 0	(null), newline, linef	feed, EO	jz pusi	short loc_31306D esi	
- targets	string functions will stop copying at the			<pre>eax, [ebp+arg_0] eax esi, 1D0h esi [ebp+arg_4]</pre>	
	rs cannot use string				nck
vector		lanction	jz cmp jz	short loc_313061 [ebp+arg_0], esi short loc_313088	
- ignores	rest of program sec	urity			
		-		0Dh sub_31411B	
		loc_31306D:		sub_3140F3 eax, eax short loc_31307D sub_3140F3 short loc_313080	
		loc_31307D:		sub_3140F3	
MBE - 04/14/2015	Stack Canaries		and or	eax, OFFFFh eax, 80070000h	70
		loc_31308C:			

Terminator Canaries

How to Defeat This:

MBE - 04/1

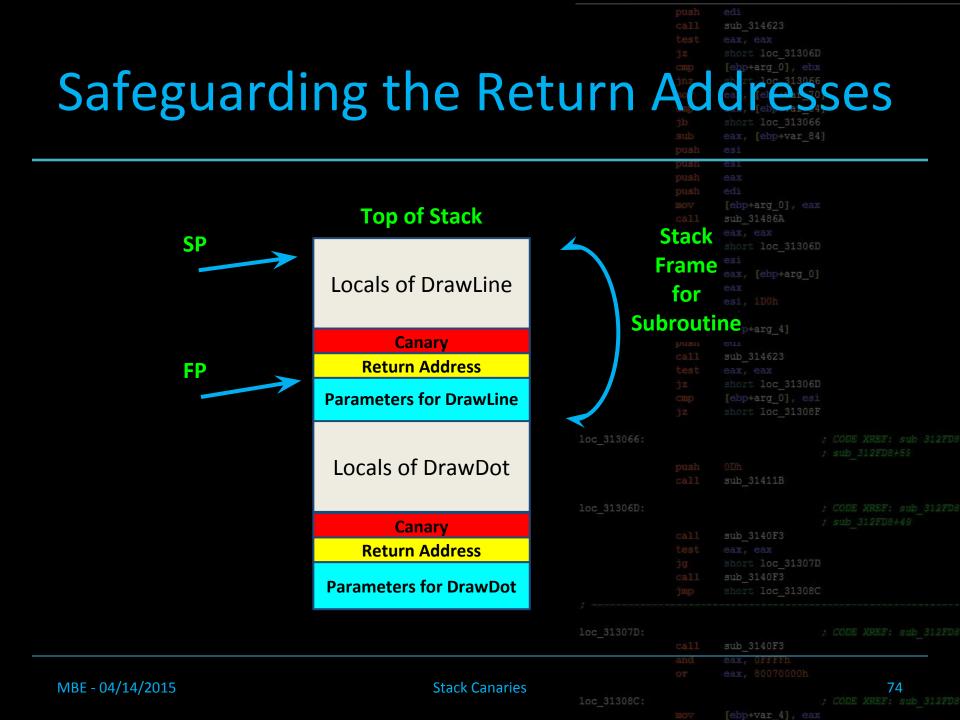
test eax, eax jz short loc_31306D cmp [ebp+arg_0], ebx jnz short loc_313066 mov eax, [ebp+var_70] cmp eax, [ebp+var_84] jb short loc_313066 sub eax, [ebp+var_84] push eax push esi push eax push edi mov [ebp+arg_0], eax call sub_31486A test eax, eax jz short loc_31306D push esi lea eax, [ebp+arg_0] push eax

- if input is treated as binary data and not text
- overwrite the canary with its known value, passing the canary check code
 - control information with mismatched values

- 6	executed soon before the	ne return	instru	uction	
				sub_31411B	
		loc_31306D:			
				sub_3140F3	
				short loc_313071	
				sub_3140F3	
				short loc_313080	
		loc_31307D:		sub_3140F3	
			and	eax, OFFFFh	
4/2015	Stack Canaries				71
+/2013	Stack Cananes	loc_31308C:			

			sub_314623	
			short loc_31306D	
			[ebp+arg_0], ebx	
			short loc_313066	
Torminator Canaria			eax, [ebp+var_70 eax, [ebp+var_84	
Terminator Canarie			short loc 313066	
			eax, [ebp+var 84	
		pusn	esi	
			[ebp+arg_0], eax	
			sub_31486A	
			eax, eax short loc 31306D	
Seems like a bad idea, who wou	ild use this	jz push	esi	
		lea	eax, [ebp+arg_0]	
			eax	
			[ebp+arg_4]	
GCO				
			sub_314623	
			short loc_31306D	
			<pre>[ebp+arg_0], esi short loc 31308F</pre>	
"If a random generator can'	t be usea, tr	ie pi	rotector	
switches the guard to the te	rminator car	narv	Dh.	
		ron y	Sub_31411B	
	loc_31306D:			
			sub 3140F3	
			short loc_31307D	
			sub_3140F3	
			short loc_31308C	
	loc_31307D:			
			sub_3140F3	
		and	eax, OFFFFh	
MBE - 04/14/2015 Stack Cana				72
MBE - 04/14/2015 Stack Cana	aries			72 ; CODE XREF: sub_312FD8

Randomized Canaries	call test jz cmp jnz mov cmp jb sub sub	<pre>sub_314623 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] esi</pre>	
		esi eax edi [ebp+arg_0], eax	
Most popular (GCC uses them)		sub_31486A eax, eax short loc_31306D	
 random number chosen at program 		rtup ^{bp+arg_0}	
- attacker must be dynamic		esi [ebp+arg_4]	
 inserts into every stack frame 		edi sub_314623 eax, eax	
 trigger: return addresses 		<pre>short loc_31306D [ebp+arg_0], esi short loc 31308F</pre>	
- Some possibilities			
- NULL canaries		0Dh sub_31411B	
loc_31306D:			
 gcc on a typical 32-bit machine 		sub_3140F3 eax, eax	
is \Rightarrow 4 byte canary		short loc_31307D sub_3140F3 short loc_31308C	
loc_31307D:		; sub_3140F3	
MBE - 04/14/2015 Stack Canaries	and or	eax, OFFFTh eax, 80070000h	73 CODE XREF: sub 312FD8
10C_31308C:		; [ebp+var 4], eax	



Randomized Canaries	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
GCC -fstack-protector-all -fstack-protector	<pre>push es1 push eax push edi mov [ebp+arg_0], eax call sub_31486A test eax, eax jz short loc_31306D push esi lea eax, [ebp+arg_0] push eax mov esi, 1D0h push esi</pre>
 + char array of 8 bytes or more declared on the stack +param=ssp-buffer-size=N 	<pre>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>
-fstack-protector-strong	; CODE XREF: sub 312FD8 ; sub_312FD8+59 push ODh call sub_31411B
 + declaration of type or leng arrays + local var addresses or loca variables 	test eax, eax jg short loc_31307D l register 1308C ; CODE XREF: sub_312FD8 call sub_3140F3
MBE - 04/14/2015 Stack Canaries	and eax, OFFFFh or eax, 80070000h 75 ; CODE XREF: sub_312FD8 mov [ebp+var_4], eax

Randomized Canaries

sub_314623
short loc_31306D
[ebp+arg_0], ebx
short loc_313066
<pre>eax, [ebp+var_70]</pre>
<pre>eax, [ebp+var_84]</pre>
short loc_313066
<pre>eax, [ebp+var_84]</pre>



76

MBE - 04/14/2015

Stack Canaries

Random XOR Canaries

- They are still random!
- XOR-ed with all or part of the control data
 - if altered, the canary value is immediately invalidated
 - same vulnerabilities as random canaries in reading off stack



313066

sub 31486A

Random XOR Canarie	25	push call test jz cmp jnz mov cmp jb sub push	edi sub_314623 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 esi	
- To Bypass: read value from the sta	ck	push push push mov call test	esi eax edi [ebp+arg_0], eax sub_31486A	
- get the canary value			eax, eax short loc_31306D esi eax, [ebp+arg_0] eax	
- the control data			esi, 1D0h esi [ebp+arg_4] edi	
- the algorithm			<pre>sub_314623 eax, eax short loc_31306D [ebp+arg_0], esi</pre>	
\Rightarrow RE the XOR-ed canary	loc_313066;		short loc_31308F	
\Rightarrow spoof custom canary	for shellc	ode	0Dh sub_31411B	
	loc_31306D:		sub_3140F3 eax, eax short loc_31307D sub_3140F3 short loc_31308C	
	loc_31307D:	call and or	sub_3140F3 eax, 0FFFTh eax, 80070000h	; CODE XREF: sub_312FD8
MBE - 04/14/2015 Stack Canaries	loc_31308C:			78 ; CODE XREF: sub_312FD8

Random	Canaries

- Benefits
 - same protection as basic random canaries
 - defends against specific attacks involving control data or retur
 value changes without overflowing the canary (invalidates it)
 - XOR's the canary with the return address
 - protect against overflowing buffers in structures

_	attacker	tries to	о таке	e pointer	point at	control	oata

				sub_3140F3 short loc_313	08C
		loc 31307D:			
				sub_3140F3	
			and	eax, OFFFFh	
MBE - 04/14/2015	Stack Canaries				79
		loc 31308C:			
				Tehn+var 41	

	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
pusn	esi
	esi eax
pusn push push	esi eax
push push push mov	esi eax edi
push push push mov call	esi eax edi [ebp+arg_0], eax
push push push mov call test	esi eax edi [ebp+arg_0], eax sub_31486A
push push push mov call test	esi edi [ebp+arg_0], eax sub_31486A eax, eax short loc_31306D
push push push mov call test jz push	esi edi [ebp+arg_0], eax sub_31486A eax, eax short loc_31306D
push push push mov call test jz push	es1 eax edi [ebp+arg_0], eax sub_31486A eax, eax short loc_31306D esi eax, [ebp+arg_0]

Random XOR Canaries

- Downsides
 - + more overhead means more security
 - # of canaries (StackGuard uses 128 static) & complexity of algorithm
 - + only protects control data from being altered

IF the attacker is overwriting pointers

- + still allows overwrite of data and the pointers themselves
 - function pointers can be victimized

	edi sub_314623 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]
	<pre>eax, eax short loc_31306D [ebp+arg_0], ebx short loc_313066 eax, [ebp+var_70] eax, [ebp+var_84] short loc_313066</pre>
	<pre>short loc_31306D [ebp+arg_0], ebx short loc_313066 eax, [ebp+var_70] eax, [ebp+var_84] short loc_313066</pre>
	<pre>[ebp+arg_0], ebx short loc_313066 eax, [ebp+var_70] eax, [ebp+var_84] short loc_313066</pre>
	<pre>short loc_313066 eax, [ebp+var_70] eax, [ebp+var_84] short loc_313066</pre>
	<pre>eax, [ebp+var_70] eax, [ebp+var_84] short loc_313066</pre>
	<pre>eax, [ebp+var_84] short loc_313066</pre>
	short loc_313066
	eax, [ebp+var_84]
pusn	631
	[ebp+arg_0], eax
	sub_31486A
	short loc_31306D
	eax, [ebp+arg_0]
	lexity of algorithm
	oush oush nov call test jz oush lea push

; CODE XREF: sub 312FD: ; sub_312FD8+59

⇒ overflow into them and call to execute shellcode

 Stack Canaries
 Stack Canaries

 MBE - 04/14/2015
 Stack Canaries

 Stack Canaries
 80

 loc_31308C:
 ; CODE XREF: sub_312FD8

 mov
 [ebb+var 41, eax

Random (XOR) Canaries	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
	al of the Stor relies on goo	mov esi, 1D0h
For Both Random an security element rel generation. Pseudor learned. Cryptograp	ies on good ran andom sequen	dom number sub 314118 ber sub 314118 core xREF: sub 312708 sub 312708 sub 312708 sub 312708 sub 312708 sub 312708
MBE - 04/14/2015	loc_3130 Stack Canaries	call sub_3140F3 and eax, OFFFFh or eax, 80070000h 81

Lecture Overview

1. How do we protect against overflows?

Stack Canaries

- 2. Different Types
- 3. Guarding the Stack
- 4. Ways to Leak Information
- 5. When All Else Fails

		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	
	pusn	es1	
2			
?		[ebp+arg_0], eax	
		sub_31486A	
		short loc_31306D	
		eax, [ebp+arg_0]	
		[ebp+arg_4]	
		edi	
		sub_314623	
		short loc_31306D	
		[ebp+arg_0], esi	
		short loc_31308F	
a 212055.			
c_313066:			
		sub 31411B	
		SUD_JIAIID	
c_31306D:			
0_010000.			
		sub 3140F3	
	test	eax, eax	
		short loc_31307D	
		sub_3140F3	
		short loc 31308C	
c 31307D:			
		sub 3140F3	
	and	eax, OFFFFh	
			82
c_31308C:			: CODE XREF: sub 312FD8

Guarding the Stack	<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>	
StackGuard - Used in patc	hesi push eax push edi hesi jz short loc_31306D push eax lea eax, [ebp+arg_0] push eax	
 started in 1998 as static of original prototype writter intern 	n in a few days by an in a few days by an [ebp+arg_0], esi short loc_313066: ; CODE XREF: sub 3128	
 promptly patched into 		
 first canary was harded 	test eax, eax jg short loc_31307D call sub_3140F3 jmp short loc_31308C	
MBE - 04/14/2015 Stack Canaries	loc_31307D: ; CODE XREF: sub_3128 call sub_3140F3 and eax, 0FFFFh or eax, 80070000h loc_31308C: ; CODE XREF: sub_3128 mov [ebp+var 4], eax	

Guarding the Stack	te ja cn jr mo cn jr sr pr	est eas mp [e] nz sho mp eas p eas b eas nsh es	i —	
StackGuard - The first cana	ary ^{pr} ^{cz} ^{cz} ^j ^{cz} ^{pr} ^{cz} ^{pr} ^{cz} ^{pr} ^{cz} ^{pr} ^{cz} ^{pr} ^{cz} ^{pr} ^{cz} ^{pr} ^{cz} ^{pr} ^{cz} ^{pr} ^{cz} ^{pr} ^{cz} ^{pr} ^{cz} ^{pr} ^{cz} ^{pr} ^{cz} ^{pr} ^{cz} ^{pr} ^{cz} ^{pr} ^{cz} ^{pr} ^{cz} ^{pr} ^{cz} ^{pr} ^{cz} ^{pr} ^{cz} ^{pr} ^{cz} ^{pr} ^{cz} ^{pr} ^{cz} ^{pr} ^{cz} ^{pr} ^{cz} ^{pr} ^{cz} ^{pr} ^{cz} ^{pr} ^{cz} ^{pr} ^{pr} ^{cz} ^{pr} ^{cz} ^{pr} ^{cz} ^{pr} ^{cz} ^{pr} ^{pr} ^{cz} ^{pr} ^{pr} ^{cz} ^{pr} ^{pr} ^{cz} ^{pr} ^{pr} ^{cz} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{cz} ^{pr} ^{pr} ^{pr} ^{pr} ^{cz} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^{pr} ^p}		x i bp+arg_0], eax b_31486A x, eax brt loc_31306E i x, [ebp+arg_0] x i, 1D0h	
ØxDEADB	ca jz cm jz 100_3.0006;		<pre>b_314623 k, eax ort loc_31306E op+arg_0], esi ort loc_31308E</pre>	
	te je ce jn ; loc_31307D:		b_3140F3 k, eax brt loc_31307E b_3140F3 brt loc_313080	
MBE - 04/14/2015 Stack Canaries	ar 01 loc_31308C: mc	id ear ear	x, OFFFFh x, 80070000h	84 ; CODE XREF: sub_312FD8

Guarding the Stack

StackGuard

- terminator canary
 CR, LF, 00, -1
 single random canary
 using /dev/random
- single XOR random canary
 - xor-ed return address

		sub_314623	
		short loc_31306D	
		[ebp+arg_0], ebx	
		short loc_313066	
		<pre>eax, [ebp+var_70]</pre>	
		<pre>eax, [ebp+var_84]</pre>	
		short loc_313066	
		<pre>eax, [ebp+var_84]</pre>	
	pusn	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	
12055.			
313066:			
		0Dh sub 31/11B	
		sub_31411B	
31306D:			
H1000D:			
		, sub 3140F3	
		eax, eax short loc_31307D	
		sub_3140F3	
		short loc 31308C	
31307D:			
		sub 3140F3	
	and	eax, OFFFTh	
		eax, 80070000h	
			85
31308C:			CODE XREF: sub 312FD8

Stack Canaries

Guarding the Stack	pus cal tes jz cmp jnz mov cmp jb suk pus	<pre>11 sub_314623 st eax, eax short loc_31306D [ebp+arg_0], ebx s short loc_313066 r eax, [ebp+var_70] eax, [ebp+var_84] short loc_313066 eax, [ebp+var_84] short loc_313066 eax, [ebp+var_84]</pre>	
StackGuard - Extra Benefit	pus pus S cal tes jz pus lea pus	sh eax sh edi 7 [ebp+arg_0], eax 11 sub_31486A st eax, eax short loc_31306D sh esi a eax, [ebp+arg_0]	
 implemented as modified single XOR random canary stores the valid return 	cal tes jz	short loc_31306D (ebp+arg_0], esi	
memory	pus cal loc_31306D: cal tes jg cal jmp ; 	<pre>Il sub_31411B Il sub_3140F3 st eax, eax short loc_31307D Il sub_3140F3 short loc_31308C</pre>	
MBE - 04/14/2015 Stack Canaries	and or loc_31308C:	eax, OFFFFh eax, 80070000h	86 ; CODE XREF: sub_312FD8

Guarding the Stack	<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>
Modding StackGuard - PointGu	push eax push edi [ebp+arg_0], eax
	Sallsub_31486Atesteax, eaxjzshort loc_31306D
	push esi lea eax, [ebp+arg_0] push eax
 does everything StackGuard do 	
newer and slower	push edi call sub_314623 test eax, eax
- allows canaries to be added to	different data
items,	
 automatically: FP and longju 	
 requires users to specify wh 	call sub 3140F3
they think will be exploited	call sub_3140F3 jmp short loc_31308C
	; CODE XREF: sub_312FD8
MBE - 04/14/2015 Stack Canaries	call sub_3140F3 and eax, 0FFFFh or eax, 80070000h 87 ; CODE XREF: sub_312FD8

Guarding the Stack

- **Modding StackGuard ProPolice**
 - also at the compiler level (a patch to GC
 - does everything StackGuard does
 - enhancements:
 - variable sorting

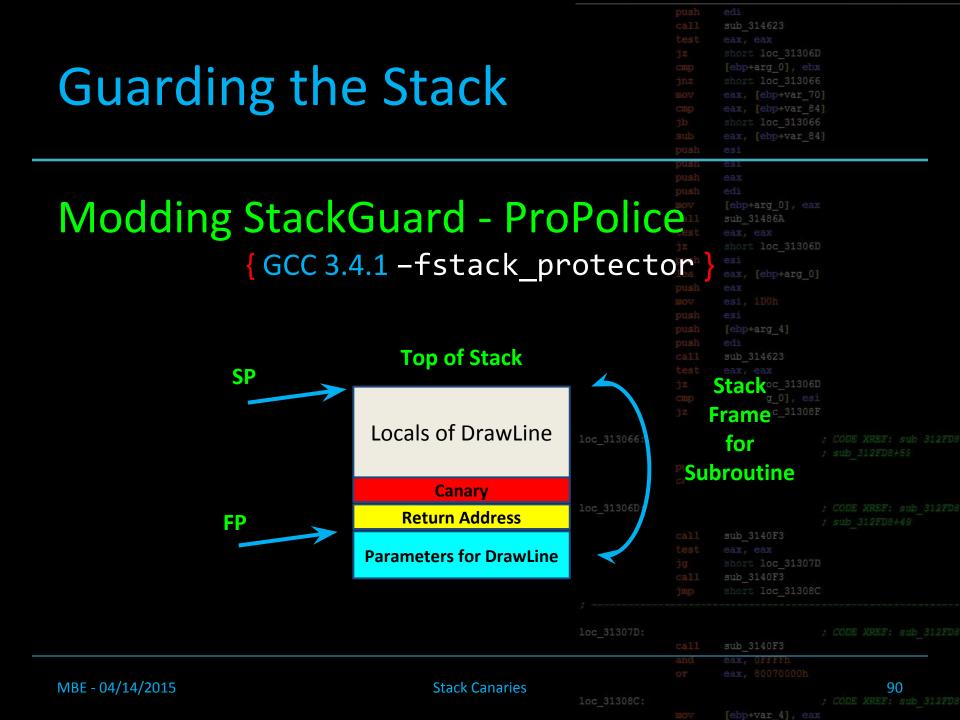
 \rightarrow buffers sorted to top of local variables, means are subjected as the subject of local variables and the subject of local variables are subject of local variables.

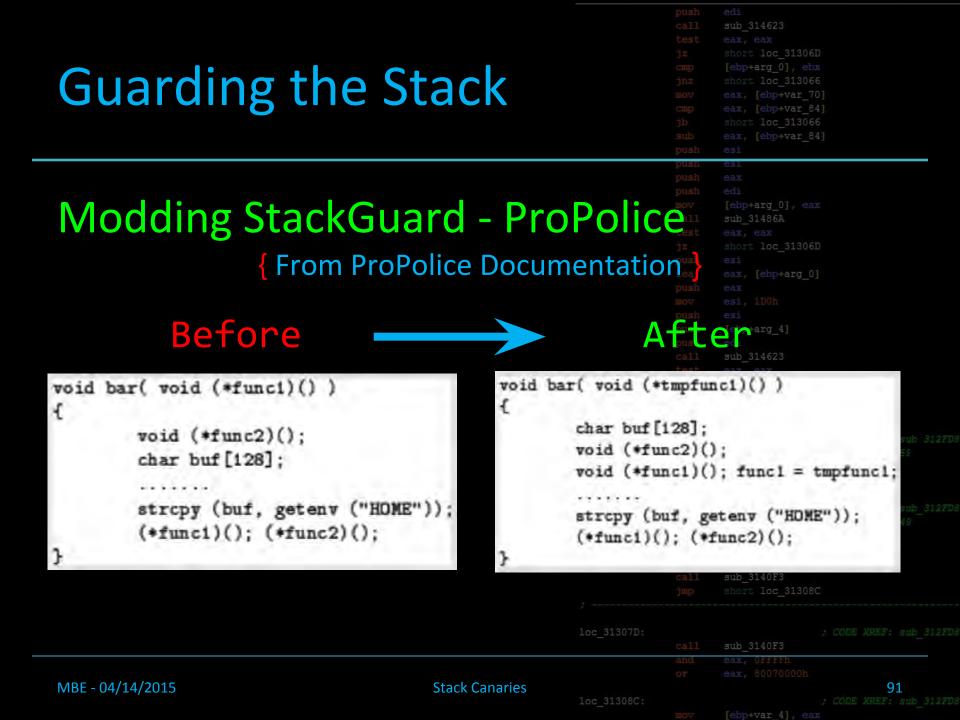
they can't overflow important values

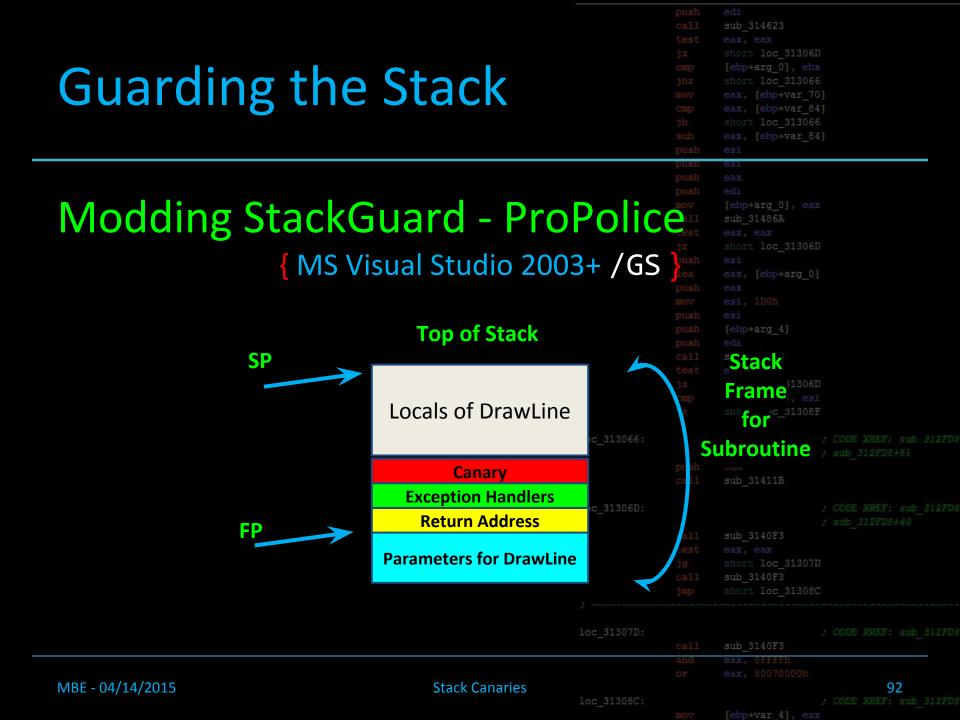
		sub_314623
		eax, eax
		short loc_31306D
		[ebp+arg_0], ebx short loc_313066
		eax, [ebp+var_70]
		eax, [ebp+var_70] eax, [ebp+var_84]
		short loc 313066
		eax, [ebp+var_84]
	pusn	es1
	MOV	[ebp+arg_0], eax
olic		sub_31486A
	Cest	
		short loc_31306D
		eax, [ebp+arg_0]
nat	plsh	
pal	CI	to GCC)
	push	
	Call	sub_314623
do	ρς	eax, eax short loc 31306D
		[ebp+arg_0], esi
		short loc 31308F
313066:		
		sub_31411B
31306D:	•	; CODE XREF: sub_312FD8
al vai	riab	CODE XREF: sub_312FD8
	Call	sub_3140F3
		eax, eax
es		short loc_31307D
		sub_3140F3
		short loc_31308C
31307D:		
		sub_3140F3
	and	eax, OFFFFh
		88

Stack Canaries

Guarding the Stack	pushedicallsub_314623testeax, eaxjzshort loc_31306Dcmp[ebp+arg_0], ebxjnzshort loc_313066moveax, [ebp+var_70]cmpeax, [ebp+var_84]jbshort loc_313066subeax, [ebp+var_84]jbshort loc_313066subeax, [ebp+var_84]pushesi	
Modding StackGuard - Pro	push eax push edi [ebp+arg_0], eax sub_31486A eax, eax jz short loc_31306D push esi lea eax, [ebp+arg_0]	
so this sounds like a good idea - Visual Studios 2003 an - GCC uses it with the fe -fstack prote	d higher cmp jz ature ioc_313066; cmp jz short loc_31306D [ebp+arg_0], esi short loc_31308F	CODE XREF: sub 312FD8 sub_312FD8+59
	loc_31306D: ;	
MBE - 04/14/2015 Stack Canaries	call sub_3140F3 and eax, OFFFTh or eax, 80070000h	CODE XREF: sub_312FD8 89 CODE XREF: sub_312FD8







Lecture Overview

1. How do we protect against overflows?

Stack Canaries

- 2. Different Types
- 3. Guarding the Stack
- 4. Ways to Leak Information
- 5. When All Else Fails

		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]	
	pusn	-631	
ק		[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	
_313066:			
		sub_31411B	
_31306D:			
		sub_3140F3	
		eax, eax	
		short loc_31307D	
		sub_3140F3	
		short loc_31308C	
:_31307D:			
		sub_3140F3	
	and	eax, OFFFFh	
		93	
313080.			

- Focus on Random Canaries
- Overwrite the Canary with the same value
 value

Stack Canaries

- brute force

- learnable random numbers_313066
- unprotected data type
- reading off of the stack

	push	edi
		sub_314623
		short loc_31306D
		[ebp+arg_0], ebx
		short loc_313066
Iry		eax, [ebp+var_70]
		eax, [ebp+var_84]
		short loc_313066
		eax, [ebp+var_84]
	push	esi
	pusn	e51
$\mathbf{O}\mathbf{C}$		[ebp+arg_0], eax
es		sub_31486A
		eax, eax
		short loc_31306D
h th		
		same
	mov	esi, 1D0h
		[ebp+arg_4]
		edi
		sub_314623
		short loc_31306D
		[ebp+arg_0], esi
		short loc_31308F
:_313066:		
		sub_31411B
c_31306D:		
		sub_3140F3
		short loc_31307D
		sub_3140F3
		short loc_31308C
a 21202D.		
c_31307D:		; CODE XREF: sub_312FD:
	call	sub_3140F3
	and	eax, OFFFFh eax, 80070000h
		94
c 31308C:		; CODE XREF: sub 312FD
		Tehnuttan /1 ear

- Brute Force

- cool example attack: http://www.esi.iboh

 <u>eu/persistence-1/</u>
 <u>bush (ebp+arg_4)</u>
 <u>push (edical)</u>
 <u>sub_314623</u>
- requires same canary for each thread so can't call execve()

Stack Canaries

overwrite canary byte by byte

loc_31308C:				5 sub_312FD8
		eax, OFFFFh eax, 80070000h		
		sub_3140F3		
, loc_31307D:				
		short loc_31308C		
		sub_3140F3		
		short loc_31307D		
	call			
	Ϳϥͺ			
l∝by⊧k	$\gamma \tau$			
		sub_31411B		
loc_313066;				

sub 31486A

Wavs	to	Leak	the	Canary

- Learnable Random Numbers
 - GS calculate the canary 2007

	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]
pusn	esi
	[ebp+arg_0], eax
	sub_31486A
	short loc_31306D
	eax, [ebp+arg_0]
	and a post-

- http://uninformed.org/?v=7&a=2&t=sumry
- bad crypto for random generator
- if /dev/random is not found, sometimes

- 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2015 - 04/14/2005 - 04/14/2005 - 04/14/2005 - 04/14/2005 - 04/14/2005 - 04/14/2005 - 04/14/2005 - 04/14/2005 - 04/14/2005 - 04/14/2005 - 04/14/2005 - 04/14/2005 - 04/14/2005 - 04/14/2005 - 04/14/2005 - 04/14/2005 - 04/14/2005 - 04/14/2005 - 04/14/2005 - 04/14/2005 - 04/14/2005 - 04/14/2005 - 04/14/2005 - 04/14/2005 - 04/14/2005 - 04/14/2005 - 04/14/2005 - 04/14/2005 - 04/14/2005 - 04/14/2005 - 04/14/2005 - 04/14/2005 - 04/14/2005 - 04/14/2005 - 04/14/2005 - 04/14/2005 - 04/14/2005 - 04/14/2005 - 04/14/2005 - 04/14/2005 - 04/14/2005 - 04/14/2005 - 04/14/2005 - 04/14/2005 - 04/14/2005 - 04/14/2005 - 04/14/2005 - 04/

96

	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]
pusn	esi
	[ebp+arg_0], eax
	sub_31486A
jz GC	short loc_31306D
	esi

- Unprotected Data Item
 - usually if it isn't a string buffer, there will not be a canary

ABE - 04/14/2015 Stack Canaries	loc_31307D: loc_31308C:	call and or mov	sub_3140F3 eax, 0FFFTh eax, 80070000h	; CODE XREF: sub_312FD: 97 ; CODE XREF: sub_312FD:
	loc_31306D:			
	loc_313066:			
			<pre>[ebp+arg_0], esi short loc_31308F</pre>	

- Reading Off of the Stack

- buffer overflow
 - overwrite null terminator
 - \rightarrow read past the end of array
 - format string vulnerabilities

	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]
usn	851
	[ebp+arg_0], eax
	sub_31486A
	short loc_31306D
	eax, [ebp+arg_0]
	[ebp+arg_4]
	sub_314623
st	eax, eax
Z	short loc_31306D
	[ebp+arg_0], esi
	short loc_31308F

; CODE XREF: sub 312FD8 ; sub 312FD8+59

http://www.exploit-monday.com/2011/06/leveraging-

- Reading Off of the Stack
 - information leaks /memory leaks (out of scope)
 - more complicated attack
 - useful against the stack reordering done by StackGuard/ ProPolice
 - pointers dangling /writing or reading after free

40F3

- http://phrack.org/issues/56/5.htm

		jg short loc_31307D call sub_3140F3 jmp short loc_31308C		
	, loc_31307D:		sub 3140F3	
		and	eax, OFFFFh	
Stack Canaries	loc_31308C:			99 CODE XREF: sub_312FD8
	Stack Canaries	Stack Canaries	call jmp ; loc_31307D: call and or Stack Canaries loc_31308C:	call sub_3140F3 jmp short loc_31308C ;

Terminator Can	aries		push call test jz cmp jnz mov cmp jb sub sub	edi sub_314623 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] esi
Ex Terminator	ercis Cana		push push mov call test jz push lea push mov push push call to to cush	esi eax edi [ebp+arg_0], eax sub_31486A eax, eax short loc_31306D esi eax, [ebp+arg_0] eax esi, 1D0h esi [ebp+arg_4] edi sub_314623) (ebp+arg_0], esi short loc_31308F
		loc_313066:		
				0Dh. sub_31411B
ssh <u>lecture@</u>)warz	20 <u>10</u> c_31306D:		; CODE XREF: sub_312FD8 sb_31453 C C_312FD8+49 short loc_31307D sub_3140F3 short loc_31308C
		;		; CODE XREF: sub_312FD8 sub_3140F3
MBE - 04/14/2015	Stack Canaries	loc_31308C:	and or mov	eax, OFFFFh eax, 80070000h ; CODE XREF: sub_312FD8 [ebp+var_4], eax

Lecture Overview

1. How do we protect against overflows?

Stack Canaries

- 2. Different Types
- 3. Guarding the Stack
- 4. Ways to Leak Information
- 5. When All Else Fails

		sub_314623	
		short loc_31306D	
		[ebp+arg_0], ebx	
		short loc_313066	
		eax, [ebp+var_70]	
		<pre>eax, [ebp+var_84]</pre>	
		short loc_313066	
		<pre>eax, [ebp+var_84]</pre>	
	pusn	esi	
,		<pre>[ebp+arg_0], eax</pre>	
		sub_31486A	
		short loc_31306D	
		<pre>eax, [ebp+arg_0]</pre>	
		[ebp+arg_4]	
		sub_314623	
		short loc_31306D	
		[ebp+arg_0], esi	
		short loc_31308F	
010055			
_313066;			
		0Dh	
		sub_31411B	
31306D:			
		sub_3140F3	
	test	eax, eax	
		short loc_31307D	
		sub_3140F3	
		short loc_31308C	
_31307D:			
		sub_3140F3	
	and	eax, OFFFFh	
			101

Remember...

No canaries on the heap

	-	[ebp+arg_0], es: short loc_31308)	
loc_313066:			
		sub_31411B	
loc_31306D:			
		sub_3140F3	
		short loc_313071	
		sub_3140F3	
		short loc_313080	
loc_31307D:			
		sub_3140F3	
	and	eax, OFFFFh	
			100

Stack Canaries

	102				
Lohn tran 11					