				sub_314623	
				short loc_31306D	
				[epp+arg_0], ebx	
				short loc_313066	
				eax, [epp+var_/u	
				edx, [eop+vdr_04	
				SHOLD IDC 313000	
				esi	
				Tebp+arg 01, eax	
				sub 31486A	
				short loc 31306D	
				eax, [ebp+arg 0]	
			push		
	теар схрю	lai	puch push call test jz	edi sub_314623 eax, eax short loc_31306D	
			cmp jz	<pre>[ebp+arg_0], esi short loc 31308F</pre>	
		1	•		
	Modern Binary Exp	oloitat	Ior		
		004	call	sub_31411B	
	(<u>S(14968 - Sprin</u>	σ J	5		
		D T A T			
				aub 2140F2	
	Markuc Gaacoc			500_5140r5	
	IVIAI KUS GAASEU	Jelen		short loc 31307D	
				sub 3140F3	
				short loc 31308C	
		loc_31307D:			
				sub_3140F3	
			and	eax, OFFFFh	
WIBE - 04/07/2015	Heap Exploitation				, CODE VDEE, and at anno
		0.0 51 51181			

Lecture Overview

Heap Overview

- Heap Exploitation
 - Heap Overflows
 - Use After Free
 - Heap Spraying
 - Metadata Corruption

	push	edi	
		sub_314623	
		short loc_31306D	
		[ebp+arg_0], ebx	
		short loc_313066	
		<pre>eax, [ebp+var_70]</pre>	
		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	
000000			
LOC_313066;			
		UUN	
		SUD_31411B	
oc 31306D:			
		sub 3140F3	
		eax, eax	
		short loc 31307D	
		sub 3140F3	
		short loc 31308C	
.oc_31307D:			
	call	sub_3140F3	
	and	eax, OFFFFh	
			2

Heap Exploitation

			ucan ucan	
			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	
	loc_313066:			
			sub_31411B	
	loc 31306D+			
Desis evenuisuu on dunensis mere		call	sub 3140F3	
Basic overview on dynamic memory al	nd neap structu	rest		
			short loc 31307D	
			sub 3140F3	
			short loc 31308C	
	loc 31307D:			
			sub 3140F3	
		and	eax, OFFFFh	
			eax, 80070000h	
- 04/07/2015 Hean Explo	itation			3
	loc 31308C:			; CODE XREF: sub 312FD8
			[ebp+var 4], eax	

MBE

The Heap

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

- The heap is pool of memory used for dynamic allocations at runtime
 - malloc() grabs memory on the heap
 - free() releases memory on the heap

	sub_314_6A	
	short loc_31306D	
	eax, [ebp+arg_0]	
MOV		
push		
	[ebp+arg_4]	
call	sub_314623	
test		
	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, OFFTTh	
	4	

Heap Exploitation

The Heap



Heap Exploitation

c_31308C:

; CODE)

B	asics of Dynamic N	/lemo	push call test jz cmp Cy 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	
int	<pre>main()</pre>			esi eax edi [ebp+arg_0], eax	
ĺ	char * buffer = NULL;			sub_31486A eax, eax short loc_31306D esi	
	<pre>/* allocate a 0x100 byte buffer */ buffer = malloc(0x100);</pre>	/		<pre>eax, [ebp+arg_0] eax esi, 1D0h esi [ebp+arg_4]</pre>	
	<pre>/* read input and print it */ fgets(stdin, buffer, 0x100); printf("Hello %s!\n", buffer);</pre>			eur sub_314623 eax, eax short loc_31306D [ebp+arg_0], esi short loc_31308F	
	/* destrov our dvnamicallv allocat	ted buffer '	* /push	ODh	
}	<pre>free(buffer); return 0;</pre>	loc_31306D:	call test jg call	sub_3140F3 eax, eax short loc_31307D sub_3140F3	
		; loc_31307D:		short loc_31308C	
MBE -	04/07/2015 Heap Exploitati	ion 10c_31308C:	and or	eax, OFFFTh eax, 80070000h	6 ; CODE XREF; sub_312FD8

Heap vs Stack

cmp [ebp+ jnz short mov eax, cmp eax, jb short sub eax, push esi push esi push eax push edi mov [ebp+

 <u>Dynamic</u> memory allocations at runtime

Heap

MBE - 04/0

- Objects, big buffers, structs, persistence, larger things
- Slower, Manual
 - Done by the programmer
 - malloc/calloc/recalloc/free
 - new/delete

- <u>Fixed</u> memory allocations known at compile time
- Local variables, return addresses, function args
- Ioc_313066:
 Fast, Automatic
 Done by the compiler
 Abstracts away any concept
 - Abstracts away any concept of allocating/de-allocating

		loc_31307D:			
				sub_3140F3	
			and	eax, OFFFFh	
7/2015	Heap Exploitation				7
		loc 31308C:			
				Tebp+var 41, ea	

Stack

Heap Implementations

	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	es1

- Tons of different heap implementations
 - dlmalloc
 - ptmalloc
 - tcmalloc
 - jemalloc
 - nedmalloc
 - Hoard

MBE - 04

short loc_31308C	
short loc_31307D sub_3140F3	
sub_3140F3	
sub_31411B	
uun	
short loc_31308F	
[ebp+arg_0], esi	
short loc_31306D	
sub_314623	
[ebp+arg_4]	

		loc_31307D:				
				sub_3140F3		
			and	eax, OFFFFh		
/07/2015	Heap Exploitation				8	
		loc_31308C:				
				Tehnutran /1 es		

Heap Implementations

	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]
DUSN	esi

- Tons of different heap implementations
 - dlmalloc
 - ptmalloc
 - tcmalloc
 - jemalloc
 - nedmalloc
 - Hoard

Some applications even c	reate	their	own he short loc 31307D	312ED8+49 a p
implementations!			sub_3140F3 short loc_31308C	

		loc 31307D:			
				sub_3140F3	
			and	eax, OFFFFh	
MBE - 04/07/2015	Heap Exploitation				9
		loc_31308C:			
				Tehn+var 41 ex	

Heap Implementations

- 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
- glibc 2.19 is what we have on the Warzone
 - Default for Ubuntu 14.04 (32bit)
 - Its heap implementation is based on ptmalloc2
 - Very fast, low fragmentation, thread safe

WIDE - 04/07/2015		loc_31308C:		Tehnutten (1 en	; CODE XREF: sub_312FD8
	Hoop Evaloitation				10
			and	eax, OFFFFh	
				sub_3140F3	
		loc_31307D:			
				short loc_313080	
				sub_3140F3	
				short loc_313071	
				sub 3140F3	
		100_313000.			
		100 21206D.			
				sub_31411B	
		loc 313066:			
				short loc_31308	
				abant 1ag 212001	

Know Thy Heap Everyone uses the heap (dynamic memory) but few usually know much about its internals • Do you even know the cost of your mallocs? MBE - 04/07/2015 Heap Exploitation 11

Malloc Trivia				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	
 malloc(32); 	How many b	oytes on	push push push the test	esi eax edi [ebp+arg_0], eax PC:30 C eax, eax	e your
• malloc(4):	malloc cl	hunks re	push push push	eax, [ebp+arg_0] eax esi, 1D0h esi	lp?
				<pre>[ebp+arg_4] edi sub_314623 eax, eax short loc_31306D [ebp+arg_0], esi</pre>	
• malloc(20);		loc_313066;		oDh sub_31411B	
 malloc(0); 		loc_31306D:		<pre>sub_3140F3 eax, eax short loc_31307D sub_3140F3 short loc_31308C</pre>	
		;	call and	sub_3140F3	; CODE XREF: sub_312FD
MBE - 04/07/2015	Heap Exploitation	loc_31308C:		eax, 80070000h [ebp+var 4], eax	12 ; CODE XREF: sub_312FD

Malloc Trivia				edi sub_314623 eax, eax short loc_31306E [ebp+arg_0], ebp short loc_31306E eax, [ebp+var_70 eax, [ebp+var_84 short loc_31306E eax, [ebp+var_84 esi	D E 5 J] 1] 5 1]
• malloc(32);	How many b	oytes on	push push mpy test	esi eax [ebp+arg_0], eax nean eax, eax	e your
— 40 bytes	malloc ch	nunks re	eally	taking u	ip ?
• malloc(4)·				eax esi, 1D0h esi	
				[ebp+arg_4] edi sub_314623 eax. eax	
$\bullet malloc(20)$				short loc_31306 [ebp+arg_0], esi short loc_31308H	
		loc_313066:			
				ODh sub_31411B	
		loc_31306D:			
 malloc(0); 				<pre>sub_3140F3 eax, eax short loc_31307I sub_3140F3 =bort loc_313080</pre>	
		loc_31307D:		sub_3140F3	
MBF - 04/07/2015	Heap Exploitation		and	eax, OFFFFh eax, 80070000h	13
		loc_31308C:		[ebp+var 4]. ear	; CODE XREF: sub_312FD

Malloc Trivia		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	D x 6 0] 4] 6 4]
• malloc(32):		push push push	esi eax edi [ebp+arg_0], ea	x
	How many byte	s on the	e neap ar	'e your
— 40 bytes	malloc chun	ks really	/ taking u	<mark>åp?</mark>
• malloc(4):				
			[ebp+arg_4]	
1C by the a			sub_314623	
— Ib byles			eax, eax	n
			[ebp+arg_0], es	
• malloc (20) ·			short loc_31308	
		13066:		
			0Dh sub_31411B	
		1306D:		
• malloc(U):			sub_3140F3	
			eax, eax short loc_31307	D
			sub_3140F3	
			511010 100_31308	
	loc 3	1307D:		
	105_0	call	sub_3140F3	
		and	eax, OFFFFh eax, 80070000h	
MBE - 04/07/2015	Heap Exploitation			14
	loc_3	13080:	Tehn+var 41 es	

Malloc Trivia		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	
• malloc(32);	How many byte	push push push	esi eax edi [ebp+arg_0], eax	
– 40 bytes	malloc chur	iks really	eax, eax ehort toc 31306D taking u eax, febp+arg 01	p?
 malloc(4); 			eax esi, 1D0h esi [ebp+arg_4]	
— 16 bytes			edi sub_314623 eax, eax short loc_31306D	
 malloc(20); 		cmp jz 313066:	[ebp+arg_0], esi short loc_31308F	
— 24 bytes			0Dh sub_31411B	
 malloc(0); 	loc	31306D: call test jg call jmp	sub_3140F3 eax, eax short loc_31307D sub_3140F3 short loc_31308C	
	loc_	_31307D: call	sub_3140F3	: CODE XREF: sub_312FI
MBE - 04/07/2015	Heap Exploitation	and or 31308C:	eax, OFFFTh eax, 80070000h	15 : CODE XREF: sub_312FI

Malloc Trivia			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	D x 6 0] 4] 6 4]
• malloc(32):			push push push	esi eax edi [ebp+arg_0], ea	*
	How many b	lytes on	test		e your
– 40 bytes	malloc cł	hunks re	eally	takingu	lp?
,				eax, [ebp+arg_0 eax	
\bullet malloc(1).					
				esi Tebp+arg 41	
— 16 hytes				sub_314623	
TO Dytes				short loc_31306	
				[ebp+arg_0], es	
• malloc(20):					
		loc_313066:			
— 24 bytes				sub_31411B	
		loc_31306D:			
• malloc(0)·				sub_3140F3	
				sub 3140F3	
— 16 bytes				short loc_31308	с
1					
		loc_31307D:		anh 2140F2	
			and	eax, OFFFFh	
	Lloop Evalation				10
MBE - 04/07/2015	Heap Exploitation	loc 31308C:			: CODE XREF: sub 312FD
				Tebp+var 41. ea	

Malloc Trivia			push call 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 short loc_31306	D x 6 0] 4] 6 4]
 malloc(32); 	How many h	wtoc on	push push push	eax edi [ebp+arg_0], ea	
		ytes on	test	eax, eax	
– 40 bytes	malloc cr	nunks re	lea	eax, [ebp+arg_0	1 p :
• malloc(4);				esi [ebp+arg_4]	
16 by the				edi sub_314623	
– To bytes				eax, eax short loc_31306	D
				<pre>[ebp+arg_0], es short loc_31308</pre>	
		loc_313066:			
24 by too				ODh	
– Z4 Dytes		a alasta		sub_31411B	
		IOC_31306D:			
				eax, eax	
				short loc_31307 sub 3140F3	D
— 16 bytes 	-101W/2			short loc_31308	с
		loc 31307D:			
			call	sub_3140F3	_
MBE - 04/07/2015	Heap Exploitation	loc 31308C:			17 ; CODE XREF: sub 312FD4
				Tehnatian Al ea	

Malloc Trivia

- malloc(32);
 - 40 bytes
- malloc(4);
 - 16 bytes
- malloc(20);
 - 24 bytes
- malloc(0); – 16 bytes – OWat

How many bytes on the head are your malloc chunks really taking up? How many did you get right? Maybe one? right?

sub 3140F3

18

MBE - 04/07/2015

Heap Exploitation

/levels/lecture/heap/sizes

recrnte@watzone:/tevers/recrnte/neaps

lecture@warzone:/levels/lecture/heap\$

prints distance between mallocs (size of chunk)

short loc_31306D
eax, [ebp+arg_0]

ointer

<pre>lecture@warzone:/levels/lecture/heap\$./sizes</pre>									
malloc(32)	is	at	0x086a6008,	40	bytes	to	the	next	pointer
malloc(4)	is	at	0x086a6030,	16	bytes	to	the	next	pointer
malloc(20)	is	at	0x086a6040,	24	bytes	to	the	next	pointer
malloc(0)	is	at	0x086a6058,	16	bytes	to	the	next	pointer
malloc(64)	is	at	0x086a6068,	72	bytes	to	the	next	pointer
malloc(32)	is	at	0x086a60b0,	40	bytes	to	the	next	pointer
malloc(32)	ie	at	0x086a60d8	40	hutes	to	the	nevt	nointer

malloc(32) is at 0x086a6100, 40 bytes to the next pointer malloc(32) is at 0x086a6128, 40 bytes to the next pointer

1-			- / /- A				short loc_31308C		
						loc_31307D:		sub 3140F3	
MBE	- 04/07/	/2015		Неа	ap Exploitation	loc_31308C:	and or mov	eax, OFFFFh eax, 80070000h [ebp+var 4], eax	19 ; CODE XREF; sub_312FD8

Heap Chu	inks		push call test jz cmp jnz mov cmp jb sub 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>unsigned f buffer = n //Out come</pre>	int * buf- malloc(0x: es a heap	<pre>fer = NU 100); chunk</pre>	push push push call test jz push lea push lea push push call test jz cmp jz	ear ear ear cdi Tebp+arg_0], ear sub_31486A ear, ear short loc_31306D esi ear, [ebp+arg_0] ear esi, 1D0h esi [ebp+arg_4] edi sub_314623 ear, ear short loc_31306D [ebp+arg_0], esi short loc_31308F		
	Heap (Chunk				o 312FD8
Previous Chunk Size (4 bytes)	Chunk Size (4 bytes)	(8 + (n ,	Data / 8)*8	8 bytes)		2_312FD8
*(buffer-2)	*(buffer-1)	*buffer	call jmp call	sub_3140F3 short loc_31308C sub_3140F3	; CODE XREF: su	b_312FD8
MBE - 04/07/2015	Heap Expl	loitation	and or	eax, OFFFFh eax, 80070000h	20 ; CODE XREF: su	b 312FD8

Heap Chunks

- Previous Chunk Size
 - Size of previous chunk (if prev chunk is free)
- Chunk Size
 - Size of entire chunk including overhead

	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
test	
z	short loc_31306D
push	
	eax, [ebp+arg_0]
	[ebp+arg_4]
	sub_314623
	short loc_31306D
	[ebp+arg_0], esi



Heap Chunks

call sub_314623 test eax, eax jz short loc_ cmp [ebp+arg_0 jnz short loc_ nov eax, [ebp+ cmp eax, [ebp+ jb short loc_ sub eax, [ebp+ push esi push eax

• Data

sh edi 7 [ebp+arg_0],

11 sub_3148

Your newly allocated memory / ptr returned by malloc

<pre>eax, [ebp+arg_0]</pre>
[ebp+arg_4]
sub_314623
short loc_31306D
<pre>[ebp+arg_0], esi</pre>
short loc_31308F



Heap Chunks

Flags

- 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 mov [ebp+arg_0], eax sub eax
- Because of byte alignment, the lower 3 bits of the chunk size field would always be zero. Instead they are used for flag bits.
- 0x01 PREV_INUSE set when previous chunk is in use
- 0x02 IS_MMAPPED set if chunk was obtained with mmap()
 0x04 NON_MAIN_ARENA set if chunk belongs to a thread arena



/levels/lecture/heap/heap

prints heap chunks fields

lecture@warzone:/levels/lecture/heap\$

lecture@warzone:/levels/lecture/heap\$./heap chunks

mallocing

<pre>[prev = 0x00000000][size = 0x00000011][data buffer (0x08276008)>] = from malloc(0) [prev = 0x00000000][size = 0x00000011][data buffer (0x08276018)>] = from malloc(4) [prev = 0x00000000][size = 0x00000011][data buffer (0x08276028)>] = from malloc(16) [prev = 0x00000000][size = 0x00000012][data buffer (0x08276038)>] = from malloc(24) [prev = 0x00000000][size = 0x00000021][data buffer (0x08276050)>] = from malloc(22) [prev = 0x00000000][size = 0x00000029][data buffer (0x08276070)>] = from malloc(32) [prev = 0x00000000][size = 0x00000049][data buffer (0x08276088)>] = from malloc(26) [prev = 0x00000000][size = 0x00000049][data buffer (0x08276168)>] = from malloc(26) [prev = 0x00000000][size = 0x00000049][data buffer (0x08276168)>] = from malloc(256) [prev = 0x00000000][size = 0x0000024][data buffer (0x08276168)>] = from malloc(26) [prev = 0x00000000][size = 0x00000249][data buffer (0x08276188)>] = from malloc(1024) [prev = 0x00000000][size = 0x00000249][data buffer (0x08276478)>] = from malloc(1024) [prev = 0x00000000][size = 0x00000249][data buffer (0x08276478)>] = from malloc(248) [prev = 0x00000000][size = 0x00000249][data buffer (0x08276478)>] = from malloc(248) [prev = 0x00000000][size = 0x00000249][data buffer (0x08277088)>] = from malloc(248) [prev = 0x00000000][size = 0x0000109][data buffer (0x08278048)>] = from malloc(4046) [prev = 0x00000000][size = 0x0000109][data buffer (0x08278048)>] = from malloc(16384)] [prev = 0x00000000][size = 0x0000109][data buffer (0x08278090)>] = from malloc(16384)] [prev = 0x00000000][size = 0x0000109][data buffer (0x08278098)>] = from malloc(16384)] [prev = 0x00000000][size = 0x0000109][data buffer (0x08278098)>] = from malloc(16384)] [prev = 0x00000000</pre>				-														
<pre>[prev - 0x00000000][size - 0x00000011][data buffer (0x08276018)>] - from malloc(4) [prev - 0x00000000][size - 0x0000001][data buffer (0x08276028)>] - from malloc(8) [prev - 0x00000000][size - 0x00000021][data buffer (0x08276038)>] - from malloc(16) [prev - 0x00000000][size - 0x00000029][data buffer (0x08276050)>] - from malloc(24) [prev - 0x00000000][size - 0x00000029][data buffer (0x08276070)>] - from malloc(32) [prev - 0x00000000][size - 0x0000009][data buffer (0x08276098)>] - from malloc(64) [prev - 0x00000000][size - 0x0000009][data buffer (0x08276168)>] - from malloc(256) [prev - 0x00000000][size - 0x0000009][data buffer (0x08276170)>] - from malloc(512) [prev - 0x00000000][size - 0x00000209][data buffer (0x08276180)>] - from malloc(512) [prev - 0x00000000][size - 0x00000209][data buffer (0x08276478)>] - from malloc(1024) [prev - 0x00000000][size - 0x00000009][data buffer (0x08276478)>] - from malloc(2048) [prev - 0x00000000][size - 0x00000009][data buffer (0x08277088)>] - from malloc(2048) [prev - 0x00000000][size - 0x00000009][data buffer (0x08276090)>] - from malloc(1024) [prev - 0x00000000][size - 0x00000009][data buffer (0x08276080)>] - from malloc(1024) [prev - 0x00000000][size - 0x00000009][data buffer (0x08277088)>] - from malloc(2048) [prev - 0x00000000][size - 0x00000009][data buffer (0x08278090)>] - from malloc(8192) [prev - 0x00000000][size - 0x00000009][data buffer (0x08278090)>] - from malloc(16384) lecture@warzone:/levels/lecture/heap\$ Ceture@warzone:/levels/lecture/heap\$ Ceture@warzone:/levels/lecture/heap\$ Ceture@warzone:/levels/lecture/heap\$ Ceture@warzone:/levels/lecture/heap\$ Ceture@warzone:/levels/lecture/heap\$ Ceture@warzone:/levels/lecture/heap\$ Ceture@warzone:/levels/lecture/heap\$ Ceture@warzone:/levels/lectu</pre>	[prev		0x000	00000][size		0x00000011][data	buffer	(0x08276008)	>] –	from	malloc(0)	
<pre>[prev - 0x00000000][size - 0x00000011][data buffer (0x08276028)>] - from malloc(8) [prev - 0x00000000][size - 0x00000019][data buffer (0x08276038)>] - from malloc(16) [prev - 0x00000000][size - 0x00000029][data buffer (0x08276070)>] - from malloc(24) [prev - 0x00000000][size - 0x00000019][data buffer (0x08276070)>] - from malloc(32) [prev - 0x00000000][size - 0x00000019][data buffer (0x08276098)>] - from malloc(64) [prev - 0x00000000][size - 0x00000019][data buffer (0x08276098)>] - from malloc(128) [prev - 0x00000000][size - 0x0000019][data buffer (0x08276098)>] - from malloc(128) [prev - 0x00000000][size - 0x0000019][data buffer (0x08276168)>] - from malloc(512) [prev - 0x00000000][size - 0x00000209][data buffer (0x08276178)>] - from malloc(512) [prev - 0x00000000][size - 0x0000019][data buffer (0x08276478)>] - from malloc(1024) [prev - 0x00000000][size - 0x00000109][data buffer (0x08276180)>] - from malloc(2048) [prev - 0x00000000][size - 0x0000109][data buffer (0x08277088)>] - from malloc(2048) [prev - 0x00000000][size - 0x0000109][data buffer (0x08277088)>] - from malloc(4096) 112702 [prev - 0x00000000][size - 0x0000109][data buffer (0x08278090)>] - from malloc(1024) [prev - 0x00000000][size - 0x0000109][data buffer (0x08278090)>] - from malloc(16384) lecture@warzone:/levels/lecture/heap\$ lecture@warzone:/levels/lecture/heap\$</pre>	[prev		0x000	00000][size		0x00000011][data	buffer	(0x08276018)	>] –	from	malloc(4)	
<pre>[prev - 0x00000000][size - 0x00000019][data buffer (0x08276038)>] - from malloc(16) [prev - 0x00000000][size - 0x00000021][data buffer (0x08276050)>] - from malloc(24) [prev - 0x00000000][size - 0x00000029][data buffer (0x08276070)>] - from malloc(32) [prev - 0x00000000][size - 0x00000089][data buffer (0x08276098)>] - from malloc(64) [prev - 0x00000000][size - 0x00000089][data buffer (0x082760e0)>] - from malloc(128) [prev - 0x00000000][size - 0x00000109][data buffer (0x08276168)>] - from malloc(256) [[prev - 0x00000000][size - 0x00000209][data buffer (0x08276270)>] - from malloc(512) [prev - 0x00000000][size - 0x00000009][data buffer (0x08276478)>] - from malloc(1024) [prev - 0x00000000][size - 0x00000009][data buffer (0x08276880)>] - from malloc(2048) [prev - 0x00000000][size - 0x00000009][data buffer (0x08277088)>] - from malloc(2048) [prev - 0x00000000][size - 0x00001009][data buffer (0x08277088)>] - from malloc(4096) [12708 [prev - 0x00000000][size - 0x00001009][data buffer (0x08278090)>] - from malloc(4096) [12708 [prev - 0x00000000][size - 0x00001009][data buffer (0x08278090)>] - from malloc(1024) [prev - 0x00000000][size - 0x00001009][data buffer (0x08278090)>] - from malloc(16384) [prev - 0x00000000][size - 0x00000009][data buffer (0x08278098)>] - from malloc(16384)]ecture@warzone:/levels/lecture/heap\$</pre>	[prev		0x000	00000][size		0x00000011][data	buffer	(0x08276028)	>] –	from	malloc(8)	
<pre>[prev - 0x00000000][size - 0x00000021][data buffer (0x08276050)>] - from malloc(24) [prev - 0x00000000][size - 0x00000029][data buffer (0x08276070)>] - from malloc(32) [prev - 0x00000000][size - 0x00000049][data buffer (0x0827608)>] - from malloc(64) [prev - 0x00000000][size - 0x00000089][data buffer (0x08276080)>] - from malloc(128) [prev - 0x00000000][size - 0x00000009][data buffer (0x08276168)>] - from malloc(256) [prev - 0x00000000][size - 0x00000209][data buffer (0x08276270)>] - from malloc(512) [prev - 0x00000000][size - 0x00000209][data buffer (0x08276478)>] - from malloc(1024) [prev - 0x00000000][size - 0x00000009][data buffer (0x08276480)>] - from malloc(1024) [prev - 0x00000000][size - 0x0000109][data buffer (0x08276880)>] - from malloc(2048) [prev - 0x00000000][size - 0x0000109][data buffer (0x08277088)>] - from malloc(2048) [prev - 0x00000000][size - 0x0000109][data buffer (0x08278090)>] - from malloc(4096) [12704 [prev - 0x00000000][size - 0x0000109][data buffer (0x08278090)>] - from malloc(16384) [prev - 0x00000000][size - 0x0000109][data buffer (0x08278090)>] - from malloc(16384) [prev - 0x00000000][size - 0x00000009][data buffer (0x08278090)>] - from malloc(16384) [prev - 0x00000000][size - 0x00000009][data buffer (0x08278090)>] - from malloc(16384) [prev - 0x00000000][size - 0x00000009][data buffer (0x08278090)>] - from malloc(16384) [prev - 0x00000000][size - 0x00000009][data buffer (0x08278090)>] - from malloc(16384)</pre>	[prev		0x000	00000][size		0x00000019][data	buffer	(0x08276038)	>] –	from	malloc(16)	
<pre>[prev - 0x00000000][size - 0x00000029][data buffer (0x08276070)>] - from malloc(32) [prev - 0x00000000][size - 0x00000049][data buffer (0x08276098)>] - from malloc(128) [prev - 0x00000000][size - 0x00000109][data buffer (0x08276168)>] - from malloc(256) [prev - 0x00000000][size - 0x00000209][data buffer (0x08276270)>] - from malloc(512) [prev - 0x00000000][size - 0x00000409][data buffer (0x08276478)>] - from malloc(1024) [prev - 0x00000000][size - 0x00000409][data buffer (0x08276880)>] - from malloc(2048) [prev - 0x00000000][size - 0x0000109][data buffer (0x08277088)>] - from malloc(2048) [prev - 0x00000000][size - 0x0000109][data buffer (0x08277088)>] - from malloc(4096) [2770 [prev - 0x00000000][size - 0x0000109][data buffer (0x08277088)>] - from malloc(1024) [prev - 0x00000000][size - 0x0000109][data buffer (0x08277088)>] - from malloc(1024) [prev - 0x00000000][size - 0x0000109][data buffer (0x08278090)>] - from malloc(16384) [prev - 0x00000000][size - 0x0000109][data buffer (0x08278090)>] - from malloc(16384) [prev - 0x00000000][size - 0x0000109][data buffer (0x08278090)>] - from malloc(16384) [prev - 0x00000000][size - 0x0000109][data buffer (0x08278090)>] - from malloc(16384)</pre>	[prev		0x000	00000][size		0x00000021][data	buffer	(0x08276050)	>] –	from	malloc(24)	
<pre>[prev - 0x0000000][size - 0x00000049][data buffer (0x08276098)>] - from malloc(64) [prev - 0x00000000][size - 0x00000089][data buffer (0x082760e0)>] - from malloc(128) [prev - 0x00000000][size - 0x00000109][data buffer (0x08276168)>] - from malloc(256) [prev - 0x00000000][size - 0x00000209][data buffer (0x08276270)>] - from malloc(512) [prev - 0x00000000][size - 0x00000409][data buffer (0x08276478)>] - from malloc(1024) [prev - 0x00000000][size - 0x00000809][data buffer (0x08276880)>] - from malloc(2048) [prev - 0x00000000][size - 0x00001009][data buffer (0x08277088)>] - from malloc(2048) [prev - 0x00000000][size - 0x00001009][data buffer (0x08278090)>] - from malloc(4096) [2770 [prev - 0x00000000][size - 0x00001009][data buffer (0x08278090)>] - from malloc(16384) [prev - 0x00000000][size - 0x00001009][data buffer (0x08278090)>] - from malloc(16384) [prev - 0x00000000][size - 0x0000409][data buffer (0x08278098)>] - from malloc(16384) [prev - 0x00000000][size - 0x0000409][data buffer (0x08278098)>] - from malloc(16384) [prev - 0x00000000][size - 0x00004099][data buffer (0x08278098)>] - from malloc(16384) [prev - 0x00000000][size - 0x00004099][data buffer (0x08278098)>] - from malloc(16384)</pre>	[prev		0x000	00000][size		0x0000029][data	buffer	(0x08276070)	>] –	from	malloc(32)	
<pre>[prev - 0x00000000][size - 0x00000089][data buffer (0x082760e0)>] - from malloc(128) [prev - 0x00000000][size - 0x00000109][data buffer (0x08276168)>] - from malloc(256) [prev - 0x00000000][size - 0x00000209][data buffer (0x08276270)>] - from malloc(512) [prev - 0x00000000][size - 0x00000409][data buffer (0x08276478)>] - from malloc(1024) [prev - 0x00000000][size - 0x00000809][data buffer (0x08276880)>] - from malloc(2048) [prev - 0x00000000][size - 0x00001009][data buffer (0x08277088)>] - from malloc(4096) [1270] [prev - 0x00000000][size - 0x00001009][data buffer (0x08278090)>] - from malloc(4096) [1270] [prev - 0x00000000][size - 0x00001009][data buffer (0x08278090)>] - from malloc(4096) [1270] [prev - 0x00000000][size - 0x00001009][data buffer (0x08278090)>] - from malloc(16384) [prev - 0x00000000][size - 0x00004009][data buffer (0x08278098)>] - from malloc(16384) [prev - 0x00000000][size - 0x00004009][data buffer (0x08278098)>] - from malloc(16384)</pre>	[prev		0x000	00000][size		0x00000049][data	buffer	(0x08276098)	>] –	from	malloc(64)	
<pre>[prev - 0x00000000][size - 0x00000109][data buffer (0x08276168)>] - from malloc(256) [prev - 0x00000000][size - 0x00000209][data buffer (0x08276270)>] - from malloc(512) [prev - 0x00000000][size - 0x00000009][data buffer (0x08276478)>] - from malloc(1024) [prev - 0x00000000][size - 0x00000009][data buffer (0x08276880)>] - from malloc(2048) [prev - 0x00000000][size - 0x00001009][data buffer (0x08277088)>] - from malloc(4096) [27D [prev - 0x00000000][size - 0x00002009][data buffer (0x08278090)>] - from malloc(8192) [prev - 0x00000000][size - 0x00000009][data buffer (0x08278090)>] - from malloc(16384) lecture@warzone:/levels/lecture/heap\$ lecture@warzone:/levels/lecture/heap\$</pre>	[prev		0x000	00000][size		0x0000089][data	buffer	(0x082760e0)	>] –	from	malloc(128)	
<pre>[prev - 0x00000000][size - 0x00000209][data buffer (0x08276270)>] - from malloc(512) [prev - 0x00000000][size - 0x00000409][data buffer (0x08276478)>] - from malloc(1024) [prev - 0x00000000][size - 0x00001009][data buffer (0x08276880)>] - from malloc(2048) [prev - 0x00000000][size - 0x00001009][data buffer (0x08277088)>] - from malloc(4096) [12700 [prev - 0x000000000][size - 0x00002009][data buffer (0x08278090)>] - from malloc(8192) [prev - 0x00000000][size - 0x00001009][data buffer (0x08278090)>] - from malloc(8192) [prev - 0x00000000][size - 0x00004009][data buffer (0x08278098)>] - from malloc(16384) lecture@warzone:/levels/lecture/heap\$ lecture@warzone:/levels/lecture/heap\$</pre>	[prev		0x000	00000][size		0×00000109][data	buffer	(0x08276168)	>] –	from	malloc(256)	
<pre>[prev - 0x00000000][size - 0x00000409][data buffer (0x08276478)>] - from malloc(1024) [prev - 0x00000000][size - 0x00000809][data buffer (0x08276880)>] - from malloc(2048) [prev - 0x00000000][size - 0x00001009][data buffer (0x08277088)>] - from malloc(4096) [prev - 0x00000000][size - 0x00002009][data buffer (0x08278090)>] - from malloc(8192) [prev - 0x00000000][size - 0x00004009][data buffer (0x08278090)>] - from malloc(8192) [prev - 0x00000000][size - 0x00004009][data buffer (0x08278098)>] - from malloc(16384) lecture@warzone:/levels/lecture/heap\$ </pre>	[prev		0x000	00000][size		0x00000209][data	buffer	(0x08276270)	>] –	from	malloc(512)	
<pre>[prev - 0x00000000][size - 0x00000809][data buffer (0x08276880)>] - from malloc(2048) [prev - 0x00000000][size - 0x00001009][data buffer (0x08277088)>] - from malloc(4096) [prev - 0x00000000][size - 0x00002009][data buffer (0x08278090)>] - from malloc(8192) [prev - 0x00000000][size - 0x00004009][data buffer (0x0827a098)>] - from malloc(16384) lecture@warzone:/levels/lecture/heap\$ lecture@warzone:/levels/lecture/heap\$ lecture@warzone:/levels/lecture/heap\$ </pre>	[prev		0x000	00000][size		0x00000409][data	buffer	(0x08276478)	>] –	from	malloc(1024)	
<pre>[prev - 0x00000000][size - 0x00001009][data buffer (0x08277088)>] - from malloc(4096) [prev - 0x00000000][size - 0x00002009][data buffer (0x08278090)>] - from malloc(8192) [prev - 0x00000000][size - 0x00004009][data buffer (0x0827a098)>] - from malloc(16384) lecture@warzone:/levels/lecture/heap\$ lecture@warzone:/levels/</pre>	[prev		0x000	00000][size		0x00000809][data	buffer	(0x08276880)	>] –	from	malloc(2048)	
<pre>[prev - 0x00000000][size - 0x00002009][data buffer (0x08278090)>] - from malloc(8192) [prev - 0x00000000][size - 0x00004009][data buffer (0x0827a098)>] - from malloc(16384) lecture@warzone:/levels/lecture/heap\$ lecture@warzone:/levels/lecture@warzone:/levels/le</pre>	[prev		0x000	00000][size		0x00001009][data	buffer	(0x08277088)	>] –	from	malloc(4096)	
<pre>[prev - 0x00000000][size - 0x00004009][data buffer (0x0827a098)>] - from malloc(16384) lecture@warzone:/levels/lecture/heap\$ call sub_3140F3 call sub_3140F3 call sub_3140F3</pre>	[prev		0x000	00000][size		0x00002009][data	buffer	(0x08278090)	>] –	from	malloc(8192)	
lecture@warzone:/levels/lecture/heap\$ call sub_3140F3 call sub_3140F3	[prev		0x000	00000][size		0x00004009][data	buffer	(0x0827a098)	>] –	from	malloc(16384)	
call sub_3140F3	1	ecture	≥@ĭ	warzon	e:/lev	vel:	s/lect	uı	ce/heap\$									
				io maton	•• /1 ••				o /boon¢							3140F3		
															- bab			

				short loc_3130	BC
		loc 31307D:			
				sub_3140F3	
			and	eax, OFFFFh	
MBE - 04/07/2015	Heap Exploitation				24
		loc_31308C:			

KS

sub 31486A



Heap Allocations

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]



Heap Allocations

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]



Heap Allocations

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 edi call sub_314623 test eax, eax jz short loc_31306D cmp [ebp+arg_0], ebx jhz short loc_313066 mov eax, [ebp+var_70] cmp eax, [ebp+var_84] jb short loc_313066 sub eax, [ebp+var_84] push esi

- Heap grows DOWN towards higher memory
- Stack grows UP towards lower memory

Heap Exploitation

MBE - 04/07/2015



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]



- Heap grows DOWN towards higher memory
- Stack grows UP towards lower memory

Heap Exploitation

Any ideas why?

MBE - 04/07/2015

pusn	es1
	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



- Heap grows DOWN towards higher memory
- Stack grows UP towards lower memory
- Any ideas why?
 - Probably historical reasons, gave low memory systems more room to fluctuate

		06D	ows
ap Se	egment	eax	hj
pus	esi		
		+var_84]	
	short loc	: 313066	
	eax, [ebn	+var 841	
	eax, [ebn	+var 701	
	short loc	313066	
	[ebp+arg	01. ebx	
	short loc	: 31306D	
	t eax, eax		
	1 sub 31462	3	



MBE - 04/07/2015

Heap Exploitation

3C:

Hea

; CODE AREF: S

32

Heap Chunks – In Use

- Heap chunks exist in two states
 - in use (malloc'd)
 - free'd

	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



Heap Chunks – Freed

free(buffer);

- Forward Pointer
 - A pointer to the next freed chunk
- Backwards Pointer
 - A pointer to the previous freed chunk

	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



/levels/lecture/heap/print_fre

pusn esi	
push eax	
lecture/warzone;/lecture/heapy	
mallocing	
[prev = 0x00000000][size = 0x00000011][data builer (0x0225000) ====>] = Chunk 0x02255000 = In use	
[prev - 0x00000000] [size - 0x00000011] [data burler (0x08265018)>] - Chunk 0x08265010 - In use	
[prev = 0x00000000][size = 0x00000011][data burler (0x08265028) ====>] = Chunk 0x08265020 = In use	
[prev = 0x00000000][size = 0x00000019][data burler (0x08265038) ====>] = Chunk 0x08265030 = In use	
[prev - 0x00000000][size - 0x00000021][data burler (0x08265050)>] - Chunk 0x08265048 - In use	
[prev - 0x00000000][size - 0x00000029][data burler (0x08265070)>] - Chunk 0x08265068 - In use	
[prev - 0x00000000][size - 0x00000049][data buffer (0x08265098)>] - Chunk 0x08265090 - In use	
[prev - 0x000000000][size - 0x00000089][data buffer (0x082650e0)>] - Chunk 0x082650d8 - In use	
[prev - 0x000000000][size - 0x000000109][data buffer (0x08265168)>] - Chunk 0x08265160 - In use	
[prev - 0x000000000][size - 0x000000209][data buffer (0x08265270)>] - Chunk 0x08265268 - In use	
[prev - 0x000000000][size - 0x000000409][data buffer (0x08265478)>] - Chunk 0x08265470 - In use	
[prev - 0x000000000][size - 0x00000809][data buffer (0x08265880)>] - Chunk 0x08265878 - In use	
[prev - 0x000000000][size - 0x00001009][data buffer (0x08266088)>] - Chunk 0x08266080 - In use	
[prev - 0x000000000][size - 0x00002009][data buffer (0x08267090)>] - Chunk 0x08267088 - In use	
[prev - 0x000000000][size - 0x00004009][data buffer (0x08269098)>] - Chunk 0x08269090 - In use	
freeing every other chunk	
[prev - 0x000000000][size - 0x000000011][fd - 0x08265020][bk - 0x08265048] - Chunk 0x08265000 - Freed	
[prev - 0x00000010][size - 0x00000010][data buffer (0x08265018)>] - Chunk 0x08265010 - In use	
[prev - 0x00000000][size - 0x00000011][fd - 0x08266080][bk - 0x08265000] - Chunk 0x08265020 - Freed	
[prev - 0x00000010][size - 0x00000018][data buffer (0x08265038)>] - Chunk 0x08265030 - In use	
[prev - 0x00000000][size - 0x00000021][fd - 0x08265000][bk - 0xb7730450] - Chunk 0x08265048 - Freed	
[prev - 0x00000020][size - 0x00000028][data buffer (0x08265070)>] - Chunk 0x08265068 - In use	
[prev - 0x00000000][size - 0x00000049][fd - 0xb7730450][bk - 0x08265160] - Chunk 0x08265090 - Freed	
[prev - 0x00000048][size - 0x00000088][data buffer (0x082650e0)>] - Chunk 0x082650d8 - In use	
[prev - 0x000000000][size - 0x00000109][fd - 0x08265090][bk - 0x08265470] - Chunk 0x08265160 - Freed	
[prev - 0x00000108][size - 0x00000208][data buffer (0x08265270)>] - Chunk 0x08265268 - In use	
[prev - 0x000000000][size - 0x000000409][fd - 0x08265160][bk - 0x08266080] - Chunk 0x08265470 - Freed	
[prev - 0x00000408][size - 0x00000808][data buffer (0x08265880)>] - Chunk 0x08265878 - In use	
[prev - 0x000000000][size - 0x00001009][fd - 0x08265470][bk - 0x08265020] - Chunk 0x08266080 - Freed	
[prev - 0x00001008][size - 0x00002008][data buffer (0x08267090)>] - Chunk 0x08267088 - In use	
<pre>lecture@warzone:/levels/lecture/heap\$</pre>	
call sub 3140F3	

Heap Exploitation

MBE - 04/07/2015

35

/levels/lecture/heap/print_fr

eax, eax short loc_31306D [ebp+arg_0], ebx hort loc_313066 eax, [ebp+var_84] short loc_313066 eax, [ebp+var_84] short esi

prints heap chunks in their different states

est eax, eax short loc_31306D

freeing every other chunk ...

MBE - 04/07/2015

<pre>[prev - 0x00000000][size - 0x00000011][fd - 0x08265020][bk - 0x0826</pre>	65048] -	Chunk 0x08265000 -	- Freed
<pre>[prev - 0x00000010][size - 0x00000010][data buffer (0x08265018);</pre>	>] -	Chunk 0x08265010 -	- In use
<pre>[prev - 0x00000000][size - 0x00000011][fd - 0x08266080][bk - 0x0826</pre>	65000] -	Chunk 0x08265020 -	- Freed
<pre>[prev - 0x00000010][size - 0x00000018][data buffer (0x08265038)></pre>	>] -	Chunk 0x08265030 -	- In use
<pre>[prev - 0x00000000][size - 0x00000021][fd - 0x08265000][bk - 0xb773</pre>	30450] -	Chunk 0x08265048 -	- Freed
<pre>[prev - 0x00000020][size - 0x00000028][data buffer (0x08265070)></pre>	>] -	Chunk 0x08265068 -	- In use
<pre>[prev - 0x00000000][size - 0x00000049][fd - 0xb7730450][bk - 0x0820</pre>	65160] -	Chunk 0x08265090 -	- Freed
<pre>[prev - 0x00000048][size - 0x00000088][data buffer (0x082650e0)></pre>	>] -	Chunk 0x082650d8 -	- In use
<pre>[prev - 0x00000000][size - 0x00000109][fd - 0x08265090][bk - 0x0826</pre>	65470] -	Chunk 0x08265160 -	- Freed
<pre>[prev - 0x00000108][size - 0x00000208][data buffer (0x08265270)></pre>	>] -	Chunk 0x08265268 -	- In use 12FDs
<pre>[prev - 0x00000000][size - 0x00000409][fd - 0x08265160][bk - 0x0820</pre>	66080] -	Chunk 0x08265470 -	- Freed
<pre>[prev - 0x00000408][size - 0x00000808][data buffer (0x08265880)></pre>	>] -	Chunk 0x08265878 -	- In use
<pre>[prev - 0x00000000][size - 0x00001009][fd - 0x08265470][bk - 0x0820</pre>	65020] -	Chunk 0x08266080 -	- Freed
<pre>[prev - 0x00001008][size - 0x00002008][data buffer (0x08267090)></pre>	>] -	Chunk 0x08267088 -	- In use
<pre>lecture@warzone:/levels/lecture/heap\$</pre>			
leature Augmanas /leature /hears			
		short loc_31307D	
		sub_3140F3	
		short loc_31308C	
loc 313077			
		sub 3140F3	
	and	eax. OFFFFh	

Heap Exploitation

c_31308C:

; CODE XR

36


Heap Implementations

- Heaps go way deeper
 - Arenas, Binning
 - Chunk coalescing
 - Fragmentation

	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
	[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

 The details regarding these are heavily implementation reliant, and more relevant when attempting to exploit heap metadata

		loc_31307D:	loc 31307D:			
				sub_3140F3		
			and	eax, OFFFFh		
MBE - 04/07/2015	Heap Exploitation				38	
		loc 31308C:				
				Tebretter Al es		

Heap Implementations

MBE - 04

	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

- If you want to read more about the specifics of the glibc heap implementation...
- <u>https://sploitfun.wordpress.</u>
 <u>https://sploitfun.wordpress.</u>
 <u>com/2015/02/10/understanding-glibc-malloc/</u>

		100_313066;		0Dh sub_31411B	
r read the sour	ce!	loc_31306D:		sub_3140F3 eax, eax short loc_31307I sub_3140F3 short loc_313080	
		;		sub_3140F3	
07/2015	Heap Exploitation	loc_31308C:	and or mov	eax, OFFFTh eax, 80070000h [ebp+var 4], eau	39 ; CODE XREF: sub_312FD8

Lecture Overview

- Heap Overview
- Heap Exploitation
 - Heap Overflows
 - Use After Free
 - Heap Spraying
 - Metadata Corruption

		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	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	
loc 313066:			: CODE XREF: sub 312FD8
		sub 31411B	
loc 31306D:			
		sub 3140F3	
	test	eax, eax	
		short loc 31307D	
		sub 3140F3	
		short loc 31308C	
loc 31307D:			
		sub 3140F3	
	and	eax, OFFFFh	
			40

				sub 314623	
				short loc 31306	D
				[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
				Tehntang 01 eas	
				aub 21/967	
				SUD_JI400A	
				can, can	
				SHOLF TOC_21200	
				eax, [epp+arg_0	
				Cd.L	
				[epp+arg_4]	
				sub_314623	
				short 10C_31306	D
				[ebp+arg_0], es.	
				short loc_31308	
		loc 313066:			
				sub 31411B	
	UNATION	loc_31306D:			
Common boon related	concents as used in a	wolaitatia	call	sub_3140F3	
common neap related	concepts as used in e	xpionatic	test		
				short loc_31307	D
				sub_3140F3	
				short loc_31308	c
		loc_31307D:			
			call	sub_3140F3	
			and	eax, OFFFFh	
5 04/07/2045					
E - 04/07/2015	Heap Exploitation				41
		Loc 31308C:			

Lecture Overview

- Heap Overview
- Heap Exploitation
 - Heap Overflows
 - Use After Free
 - Heap Spraying
 - Metadata Corruption

	push	edi	
		sub_314623	
		short loc_31306	
		[ebp+arg_0], ebt	
		short loc_31306	
		eax, [ebp+var_7	
		eax, [ebp+var_8	4]
		short loc_31306	
		eax, [ebp+var_8	4]
	pusn	631	
		[ebp+arg_0], eas	
		sub_31486A	
		short loc_31306	
		eax, [ebp+arg_0]	
		[ebp+arg_4]	
		sub_314623	
		short loc_31306	D
		[ebp+arg_0], es:	
		short loc_31308.	
00 313066			
		sub 31411B	
oc 31306D:			
		sub 3140F3	
	test	eax, eax	
		short loc 31307	
		sub 3140F3	
		short loc 31308	
OC_31307D:		aub 21/072	
	Call	SUD_SI4UPS	
		dan, oduroudii	42
			42

MBE - 04/07/2015

	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

Buffer overflows are basically the same on the

Heap Exploitation

heap as they are on the stack

	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
	43
	Tehn+var 41 eav

Heap Overflows		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	
 Buffer overflows are basically the heap as they are on the stack 	push push call test jz push lea	esi eax sub_31486A eax, eax short loc_31306D esi eax, febreard 01	the

Heap cookies/canaries aren't a the

	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_4]	
push		
ning	sub_314623	
ie st	eax, eax	
jz	short loc_31306D	
	[ebp+arg_0], esi	
	short loc_31308E	
	sub_31411B	
	sub 3140F3	
	short loc 31307D	
	sub 3140F3	
	short loc_313080	
	sub_3140F3	
and	eax, OFFFFh	
		44

MBE - 04/07/2015

	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	e51

- Buffer overflows are basically the same on the heap as they are on the stack
- Heap cookies/canaries aren't a thing
 - No 'return' addresses to protect

		45
	eax, OFFFFh	
	sub_3140F3	
	sub_3140r3	
	BHOIL 10C_313U/U	
	eax, eax	
	sub_3140F3	
	sub_31411B	
	short loc 31308F	
	[ebp+arg 0], esi	
۲,	short loc 31306D	
18	ALL J14023	
	cul aub 21/622	
	[epp+arg_4]	
	eax, [ebp+arg_0]	
	short loc_31306D	

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]



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]



• In the real world, lots of cool and complex things like objects/structs end up on the heap of the intervence of the in	Heap Overflow	۷S		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_80 short loc_313060 eax, [ebp+var_80 eax, [ebp+var_80	0 6 5 0] 4] 6 4]
MBE - 04/07/2015 Heap Exploitation Image of the set of the s	 In the real world, things like objects 	lots of coo s/structs e	ol and end up	call correction call test Or	mplex the h	eap
Ioc_313066: ; CODE XREF: sub_312FD push ODh call sub_31411B Ioc_31306D: ; CODE XREF: sub_312FD ; sub_312FD8+49 ; sub_312FD8+49 call sub_3140F3 test ear, eax jg short loc_31307D call sub_3140F3 test ear, eax jg short loc_31308C /					<pre>eax; [ebp+arg_0] eax esi; 1D0h esi [ebp+arg_4] edi sub_314623 eax, eax short loc_313061 [ebp+arg_0], esi short loc_313081</pre>	
loc_31306D: ; CODE XREF: sub_312FD8:49 call :sub_3140F3 test :eax, eax jg :short loc_31307D call :sub_3140F3 jmp :short loc_31307D call :sub_3140F3 jmp :short loc_31307D call :sub_3140F3 jmp :short loc_31308C / :call :sub_3140F3 and :eax, 07FFFn or :eax, 80070000h Heap Exploitation 48 loc_31308C: : CODE XREF: sub_312FD			loc_313066:		0Dh sub_31411B	
loc_31307D: ; CODE XREF: sub_312FD. call sub_3140F3 and eax, 0FFFTh or eax, 80070000h MBE - 04/07/2015 Heap Exploitation Loc_31308C: ; CODE XREF: sub_312FD.			loc_31306D:		sub_3140F3 eax, eax short loc_313071 sub_3140F3 short loc_31308(
	MBE - 04/07/2015	Heap Exploitation	loc_31307D: loc_31308C:	call and or	sub_3140F3 eax, OFFFFh eax, 80070000h	; CODE XREF: sub_312FD8 48 ; 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]
pusn	esi

 In the real world, lots of cool and complex things like objects/structs end up on the heap
 Anything that handles the data you just corrupted is now viable attack surface in the application

test	
	short loc_31306D
	<pre>[ebp+arg_0], esi</pre>
	short loc_31308F

 It's common to put function pointers in structs which generally are malloc'd on the heap

				short loc_31308C		
		loc_31307D:				
				sub_3140F3		
			and	eax, OFFFFh		
MBE - 04/07/2015	Heap Exploitation				50	
		loc_31308C:				
				Tehn+var 41 ex		

	sub_314623
	short loc_31306D
	[ebp+arg_0], ebx
	short loc_313066
	<pre>eax, [ebp+var_70]</pre>
	eax, [ebp+var_84]
	short loc_313066
	<pre>eax, [ebp+var_84]</pre>
pusn	es1

 In the real world, lots of cool and complex things like objects/structs end up on the heap
 Anything that handles the data you just corrupted is now viable attack surface in the application

test	
	short loc_31306D
	[ebp+arg_0], esi
	short loc_31308F

 It's common to put function pointers in structs which generally are malloc'd on the heap

– Overwrite a function pointer on the heap, and force a codepath to call that object's function!

		loc 31307D:			
				sub_3140F3	
			and	eax, OFFFFh	
MBE - 04/07/2015	Heap Exploitation				51
		loc_31308C:			
				Lohn gron Al	

Heap Overflov	VS		edi sub_314623 eax, eax short loc_31306f [ebp+arg_0], ebx short loc_313066 eax, [ebp+var_70 eax, [ebp+var_84 short loc_313066 eax, [ebp+var_84 esi	
struct toystr void (* m char buff	{ lessage)(c er[20]; ^{100_313061}	push push push mov call test jz push lea push nov push push call cmp jz	esi eax edi [ebp+arg_0], eax sub_31486A eax, eax short loc_31306f esi eax, [ebp+arg_0] eax esi, 1D0h esi [ebp+arg_4] edi sub_314623 eax, *x short loc_1308f [ebp+arg_0], csi short loc_31308f	; ; CODE XREF: sub 312FD8 ; sub_312FD8+55
};	loc_31306 : loc_31307	call test jg call jmp): call	sub_3140F3 eax, eax short loc_31307I sub_3140F3 short loc_313080 sub_3140F3 eax. 0FFFb	; CODE XREF: sub_312FD8 ; sub_312FD8+49 ; ; ; CODE XREF: sub_312FD8
MBE - 04/07/2015	Heap Exploitation			52 ; CODE XREF: sub_312FD8

Heap Over	flows			<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]</pre>	
			push pusn	esi esi	
<pre>coolguy = malloc(s</pre>	<pre>izeof(struct toystr))</pre>	;		eax edi [ebp+arg_0], eax sub_31486A eax, eax	
<pre>lameguy = malloc(s</pre>	<pre>izeof(struct toystr))</pre>	•		<pre>short loc_31306D esi eax, [ebp+arg 0]</pre>	
<pre>coolguy->message = lameguy->message =</pre>	<print_cool; &print_meh;</print_cool; 			eax	
<pre>printf("Input cool fgets(coolguy->buf coolguy->buffer[st</pre>	<pre>guy's name: "); fer, 200, stdin); // rcspn(coolguy->buffer</pre>	oopz . "\n")]	call test jz cmp j ^z = 0:	<pre>sub_314623 eax, eax short loc_31306D [ebp+arg_0], esi short loc_31308F</pre>	
<pre>printf("Input lame</pre>	<pre>guy's name: ");</pre>	loc_313066;	push call	0Dh sub_31411B	
fgets(lameguy->buf lameguy->buffer[st	fer, 20, stdin); rcspn(lameguy->buffer	, "\n")]	= 0;	sub_3140F3	
<pre>coolguy->message(c lameguy->message(l</pre>	<pre>oolguy->buffer); ameguy->buffer);</pre>			short loc_31307D sub_3140F3 short loc_31308C	
		loc_31307D:	call	sub_3140F3	CODE XREF: sub_312FD8
MBE - 04/07/2015	Heap Exploitation	loc_31308C:	or	eax, Orrith eax, 80070000h	53 CÓDE XREF: sub_312FD8
				[ebp+var 4], eax	



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]



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





				edi sub_314623	
/levels/lecture	e/heap/	'heap	jz cmp inz yv jb sub push	short loc_313060 [ebp+arg_0], ebx Sx Dbp v210 short loc_313066 eax, [ebp+var_84 esi	sh
tov function n	ointer ove	rwrita	push push	hoan	
	onner ove	loc_313066;	call test jz push lea push push push call test jz cmp jz	<pre>[epp arg_0, gradewidth="bordewidth"] sub_31486A 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_31308F 0Dh</pre>	
		loc_31306D:		sub_3140F3 eax, eax short loc_31307E sub_3140F3 short loc_31308C	
MBE - 04/07/2015	Heap Exploitation	loc_31308C:	call and or	sub_3140F3 eax, 0FFFFh eax, 80070000h	58 ; CODE XREF: sub_312FD8

Lecture Overview

- Heap Overview
- Heap Exploitation
 Heap Overflows
 - Use After Free
 - Heap Spraying
 - Metadata Corruption

		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	
	pusn	081		
		[ebp+arg 0], ea		
		sub 31486A		
		eax, eax		
		short loc 31306	D	
		eax. [ebp+arg 0		
		eav [out] [oup] arg_o		
		Tebo+arg 41		
		edi		
		sub 314623		
		ear ear		
		short loc 31306		
		[ebp+arg 0], es		
		short loc 31308		
loc 313066				
		sub 31411B		
loc 31306D:				
		sub 3140F3		
		short loc 31307		
		sub 3140F3		
		showt loc 31308		
loc 31307D:				
_		sub 3140F3		
	and	eax, OFFFFh		
		eax, 80070000h		
				59
Loc 31308C -				CODE VEET: auto 312ED2

Course Terminology

• Use After Free

MBE - 04/07/2015

- [ebp+var 84]
- A class of vulnerability where data on the heap is freed, but a leftover reference or 'dangling pointer' is used by the code as if the data were still valid
- Most popular in Web Browsers, complex programs
- Also

known as UAF	loc_313066:		0Dh sub_31411B	
	loc_31306D:			
			sub_3140F3	
			short loc_31307	D
			sub_3140F3	
			Short 10C_31308	
	loc_31307D:			
			sub_3140F3	
		and	eax, OFFFFh	
Heap Exploitation				60
	loc_31308C:			
			epp+var 4], eau	



Heap Exploitation

oc_31308C:

; CODE XREF: sub_31

61



Heap Exploitation

31308C:

; CODE XREF: sub_31

62

Big and a state of a sta



MBE - 04/07/2015

Heap Exploitation

0xFFFFFFF

loc_31308C:

63 ; CODE XREF: sub_3

ODE XREF: sub_312FD8



Heap Exploitation

308C:

; CODE XREF: sub_3

64

Course Terminology

- Dangling Pointer
 - A left over pointer in your code that references free'd data and is prone to be re-used
 - As the memory it's pointing at was freed, there's no guarantees on what data is there now
 - Also known as stale pointer, wild pointer

VIBE - 04/07/2015		loc_31308C:			; CODE XREF: sub_312FD8
	Lloop Evaloitation			eax, 80070000h	6F
			call	sub_3140F3	
		loc_31307D:			
				sub_3140F3 eax, eax short loc_31307F sub_3140F3 short loc_313080	
		loc_31306D:			



Heap Exploitation

308C:

; CODE XREF: sub

66

Use After Free

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]



Use After Free

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]



MBE - 04/07/2015

Use After Free

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]



MBE - 04/07/2015

Heap Exploitation

loc 31308C:

; CODE XR

69

Exploiting a Use After Free

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 exploit a UAF, you usually have to allocate a different type of object over the one you just freed

MBE - 04/07/2015	Heap Exploitation	loc_31308C:		; [ebp+var 4], eax	70 CODE XREF: sub_312FD8
			and	eax, OFFFFh eax, 80070000h	
		loc_31307D:		; sub_3140F3	
				eax, eax short loc_31307D sub_3140F3 short loc_31308C	

Exploiting a Use After Free

```
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 exploit a UAF, you usually have to allocate a different type of object over the one you just freed

```
struct toystr { st
void (* message)(char *);
char buffer[20];
};
};
```

```
struct person {
     inits: favorite_num; core XREF: sub 312FD
      int ages; ODh
sub_31411B
     char name[16];
                    sub 3140F3
                    sub 3140F3
                                     71
```

Exploiting a Use After Free

```
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 exploit a UAF, you usually have to allocate a different type of object over the one you just freed assume dangling pointer exists

```
1. free()
struct toystr {
```

};

void (* message)(char *); char buffer[20];

```
struct person [e{+arg_0], esi
short loc_31306D
ioinstef favorite_num;cops XREF: sub 312FD
int age;; 0Dh
sub_31411B
lochar name[16]; cops XREF: sub_312FD
sub_312F
```

			jg call jmp 1307D: call	short loc_31307 sub_3140F3 short loc_31308	7D 3C
		loc_31307D:		sub_3140F3	
			and or	eax, OFFFFh eax, 80070000h	
MBE - 04/07/2015	Heap Exploitation	loc_31308C:			72 ; CODE XREF: sub_312FD8
Exploiting a Use After Free

```
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 exploit a UAF, you usually have to allocate a different type of object over the one you just freed assume dangling pointer exists
 free()
 free()
 free()
 malloc(1)
 struct toystr {
 void (* message)(char *);

char buffer[20];

int age; ^{aub_31411B} •char name[16];

sub_3140F3
eax, eax
short loc_31307D
sub_3140F3

short loc 313080

: 31307D:

; CODE XREF: sub_31 sub 3140F3

73

};

Heap Exploitation

};

loc_31308C:

; CODE XF

Exploiting a Use After Free

```
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 exploit a UAF, you usually have to allocate a different type of object over the one you just freed assume dangling pointer exists 2. malilo $G_{b}(314)$ 23 1. free() struct toystr { struct person { void (* message)(char *); idinates: favorite_num; code XREF: sub 312 char buffer[20]; int age; aub_31411B }; char name[16]; sub 3140F3 }; sub 3140F3

3. Set favorite_num = 0x41414141

		loc_31307D:				
				sub_3140F3		
			and	eax, OFFFFh		
MBE - 04/07/2015	Heap Exploitation				74	
		loc_31308C:				
				Tebretter Al en		

Exploiting a Use After Free

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 exploit a UAF, you usually have to allocate a different type of object over the one you just freed assume dangling pointer exists 2. malilo $G_{b}(314)$ 23 1. free() struct person { struct toystr { void (* message)(char *); idinates: favorite_num; cods XREF: sub 312 char buffer[20]; int age; aub_31411B }; char name[16]; }; 4. Force dangling pointer sub 3140F3 3. Set favorite num = 0x41414141to call 'message()'

MBE - 04/07/2015

Heap Exploitation

1308C:

; CODE XR [ebp+var 4], eax 75

				edi sub_314623	
/levels/lect	cure/heap/	'hea	jz cmp inz jvv jb sub push	short loc_31306D [ebp+arg_0], ebx short loc_613066 [a], Dp var_70 var_84 short loc_313066 ear, [ebp+var_84 esi	
your	very first use a	ifter f		esi eax edi [ebp+arg_0], eax sub_31486A	
				eax, eax short loc_31306D	
				eax, [ebp+arg_0]	
				esi, 1D0h	
				esi Tehntang (1	
				edi	
				sub_314623	
				eax, eax short loc 31306D	
				[ebp+arg_0], esi	
				short loc_31308F	
		loc_313066:			
				sub_31411B	
		loc_31306D:			
				sub_3140F3	
				eax, eax	
				sub 3140F3	
				short loc_31308C	
		loc_31307D:			
			call	sub_3140F3	
				eax, 80070000h	
MBE - 04/07/2015	Heap Exploitation				76
		10C_31308C:		Tebritvar 41 eav	



UAF in the Wil	d			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	
• The 'hot' vulneral	bility nowa	adays,	push push push call test	most e	very
modern browser	exploit lev	'erage	es a		
		loc 313066:		<pre>eax, [ebp+arg_0] eax esi, 1D0h esi [ebp+arg_4] edi sub_314623 eax, eax short loc_3 [ebp+arg_0], short loc_31308F</pre>	CODE XREF: sub 312FD8
				sub_31411B	
		loc_31306D:		sub_3140F3 eax, eax short loc_31307D sub_3140F3 short loc_31308C	
		loc_31307D:			
			call and	sub_3140F3 eax, OFFFFh	
MBE - 04/07/2015	Heap Exploitation	loc_31308C:		eax, 80070000h [ebp+var 4], eax	78 : CODE XREF: sub_312FD8

IE CVE Statistics



http://blog.tempest.com.br/breno-cunha/perspectives-on-exploitdevelopment-and-cyber-attacks.html

		loc_31307D:			
				sub_3140F3	
			and	eax, OFFFFh	
MBE - 04/07/2015	Heap Exploitation				79
	and the second sec	loc 31308C:			
				Tebp+var 41 ea	

IE CVE Statistics



MBE - 04/07/2015

Heap Exploitation

loc_31308C:

; CODE X

80

UAF in the Wild		push call test jz cmp jnz mov cmp jb sub push	edi sub_314623 eax, eax short loc_313066 [ebp+arg_0], ebp short loc_313066 eax, [ebp+var_84 short loc_313066 eax, [ebp+var_84 esi	D 5 5 1 1 5 1 1
 The 'hot' vulnerability nowa modern browser exploit lev 	adays, erage		est most e sub_31486A	every
 Why are they so well liked? 			eax esi, 1D0h esi [ebp+arg_4] edi sub_314623 eax, eax short loc_3	
	loc_313066:		[ebp+arg_0], short loc_313081 ODh sub_31411B	; CODE XREF: sub 312FD8 ; sub_312FD8+59
	loc_31306D:		<pre>sub_3140F3 eax, eax short loc_31307I sub_3140F3 short loc_313080</pre>	
MBE - 04/07/2015 Heap Exploitation	loc_31307D: loc_31308C:	call and or	sub_3140F3 eax, OFFFFh eax, 80070000h	; CODE XREF: sub_312FD8 81 ; CODE XREF: sub_312FD8



UAF in the Wild		test jz cmp jnz mov cmp jb sub push	eax, eax short loc_31306E [ebp+arg_0], ebx short loc_313066 eax, [ebp+var_84 short loc_313066 eax, [ebp+var_84 esi	
 The 'hot' vulnerability nowa 	adays	, al	most e	every
modern browser exploit lev	/erage		eax, eax	
 Why are they so well liked? Doesn't require any memory 	/ corru	push mov push push call test jz cmp	eax esi, 1D0h esi [ebp+arg_4] edi sub_314623 eax, eax short loc_3 [ebp+arg_0], short loc_313085 ON to us	e e
 Can be used for info leaks 	100_313066;	push call	0Dh sub_31411B	
	loc_31306D:		sub_3140F3 eax, eax short loc_31307I sub_3140F3 short loc_313080	
	, loc_31307D:	call	sub_3140F3	; CODE XREF: sub_312FD8
MBE - 04/07/2015 Heap Exploitation	loc_31308C:		eax, 80070000h [ebp+var 4], eap	83 ; CODE XREF: sub_312FD8

UAF in the Wil	d		test jz cmp jnz mov cmp jb sub push	<pre>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>	
 The 'hot' vulneral 	bility now	adays,	push push call	nost e	very
modern browser	exploit lev	verage	test S a	eax, eax short Ac 31306D eax, [ebptarg 0]	
 Why are they so vertication Doesn't require and the second s	well liked?	y corrup	push mov push push call test jz cmp j tic	eax esi, 1D0h esi [ebp+arg_4] edi sub_314623 eax, eax short loc_3 [ebp+arg_0], short loc_31308F ON to us	e CODE XREF: sub 312FD8
 Can be used for i 	nfo leaks			0Dh sub_31411B	
 Can be used to tr control of EIP 	igger mem	ory corr	TUP test jg call jmp	short loc_31307D sub_3140F3 short loc_31308C	code XREF: sub_312FD8
		loc_31307D:	call	sub_3140F3	CODE XREF: sub_312FD8
MBE - 04/07/2015	Heap Exploitation	loc_31308C:	and or mov	eax, Orrrrh eax, 80070000h [ebp+var 4], eax	84 CODE XREF: sub_312FD8





Detecting UAF Vulnerabilities

 From the defensive perspective, trying to detect use after free vulnerabilities in complex applications is very difficult, even in industry

• Why?

- UAF's only exist in certain states of execution, so statically scanning source for them won't go far
- They're usually only found through crashes, but symbolic execution and constraint solvers are helping find these bugs faster

		loc_31307D:				
				sub_3140F3		
			and	eax, OFFFFh		
MBF - 04/07/2015	Heap Exploitation				87	
		loc 31308C:				
				Tebretvar 41 eat		

Lecture Overview

- Heap Overview
- Heap Exploitation
 Heap Overflows
 - Lico Aftor Eroo
 - Use After Free
 - Heap Spraying
 - Metadata Corruption

		sub_314623	
		short loc_31306	D
		[ebp+arg_0], eb	
		short loc_31306	
		eax, [ebp+var_7	0]
		eax, [ebp+var_8	4]
		short loc_31306	
		eax, [ebp+var_8	4]
	pusn	esi	
		[ebp+arg_0], ea	
		sub_31486A	
		short loc_31306	D
		eax, [ebp+arg_0	
		[ebp+arg_4]	
		sub_314623	
		short loc_31306	D
		[ebp+arg_0], es	
		short loc_31308	
oc_313066:			
		sub_31411B	
oc 31306D:			
		sub 3140F3	
		short loc 31307	D
		sub 3140F3	
		short loc_31308	C
oc_31307D:			
		sub_3140F3	
	and	eax, OFFFFh	
			88

Course Terminology

Heap Spraying

Μ

- A technique used to increase exploit reliability, by filling the heap with large chunks of data relevant to the exploit you're trying to land
- It can assist with bypassing ASLR
- A heap spray is not a vulnerability or security flaw

		loc_31306D:		sub_3140F3 eax, eax short loc_31307 sub_3140F3 short loc_31308	
		loc_31307D:		sub_3140F3	
BE - 04/07/2015	Heap Exploitation	loc_31308C:	and or	eax, OFFFFh eax, 80070000h	89 ; CODE XREF: sub_312FD8
				Tebp+var 41, east	











Heap Spraying in the Wild

MBE - 04/07/2015

es 1
eax, [ebp+var_84]
short loc_313066
eax, [ebp+var_84]
<pre>eax, [ebp+var_70]</pre>
short loc_313066
[ebp+arg_0], ebx
short loc_31306D
sub_314623

 Generally found in browser exploits, rare in CTF and wargames but still something you should be aware of

		eax, [ebp+arg_0	
		[ebp+arg 4]	
		sub 314623	
		short loc 31306	
		[ebp+arg 0], es	
		short loc_31308	
loc 313066:			: CODE XREF: sub 312FD
		sub_31411B	
Log 21206D.			
100_010000.			
		anh 3140F3	
		300_J14013	
		short log 31307	
		anb 2140F2	
		abort log 21209	
		SHOPE 106_31306	
loc_31307D:			
		sub_3140F3	
	and	eax, OFFFFh	
			95
loc_31308C:			

Heap Spraying in the Wild

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] eas

- Generally found in browser exploits, rare in CTF and wargames but still something you should be aware of
- Usually heap sprays are done in something like javascript placed on a malicious html page

memory = new Array();
for(i = 0; i < 0x100; i++)
 memory[i] = ROPNOP + ROP;
 id = aub_31411B
 id = aub_31411B
 id = aub_3140F3
 id = aub_3140F3

 loc_31307D:
 ; CODE XREF: sub_312FD8

 call
 sub_3140F3

 and
 eax, 0FFFh

 or
 eax, 80070000h

 MBE - 04/07/2015
 Heap Exploitation

 1oc_31308C:
 ; CODE XREF: sub_312FD8

Heap Spraying on 32bit

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

 On 32bit systems your address space is at maximum 4GB (2³² bytes)

WIBE - 04/07/2015		loc_31308C:		[ebp+var_4], eax	; CODE XREF: sub_312FD8
	Lloon Evaluitation		and	eax, OFFFFh eax, 80070000h	07
		loc_31307D:		sub_3140F3	
				sub_3140F3 eax, eax short loc_31307D sub_3140F3 short loc_31308C	

Heap Spraying on 32bit

	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	es1

- On 32bit systems your address space is at maximum 4GB (2³² bytes)
- Spray 3GB of A's onto the heap?

– +75% chance of 0x23456789 being a valid pointer!

MBE - 04/07/2015	Heap Exploitation	loc_31308C:		[ebp+var 4], eas	98 ; CODE XREF: sub_312FD: ;
			or	eax, OFFFFh eax, 80070000h	
		loc_31307D:	call	sub_3140F3	
				sub_3140F3 short loc_313080	
				sub_3140F3 eax, eax short loc_31307E	
		loc_31306D:			
				sub_31411B	

Η	leap	Spray	ying	on	32bit
---	------	-------	------	----	-------

	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

- On 32bit systems your address space is at maximum 4GB (2³² bytes)
- Spray 3GB of A's onto the heap?
 - +75% chance of 0x23456789 being a valid pointer!
 - Note: It's unlikely you would ever need to spray
 3GB of anything as heap locations can be sub 3140F3
 somewhat predictable, even with ASL R sub 3140F3

	somewhat predictable, eve		call jmp	sub_3140F3 short loc_31308C	
		;		sub_3140F3	
MBE - 04/07	7/2015 Heap Exploitation	loc_31308C:	or mov	eax, OFFFFh eax, 80070000h [ebp+var 4], eax	99 ; CODE XREF: sub_312FD2

Heap Spraying on 64bit

	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]
ousn	esi

• On 64bit heap spraying can't really be used to bypass ASLR

				short 186_31307 sub_3140F3 short 10c_31308(
		loc_31307D:		sub_3140F3	
			and	eax, OFFFFh	
MBE - 04/07/2015	Heap Exploitation	loc_31308C:		[ebp+var 4], ear	100 ; CODE XREF: sub_312FD8

incup spraying on oton	Η	leap	Spray	/ing	on	64bit
------------------------	---	------	-------	------	----	-------

MBE - 04/07/2015

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]
A81

• On 64bit heap spraying can't really be used to bypass ASLR

Heap Exploitation

Good luck spraying anywhere near 2

al	IV I	be used	to
	test	sub_31486A	
		short loc 31306D	
		eax, [ebp+arg 0]	
	pur a		
	764	hytoc	
	ash	Dytes	
		[ebp+arg_4]	
		sub_314623	
		short loc_31306D	
		<pre>[ebp+arg_0], esi</pre>	
		short loc_31308F	
		sub_31411B	
		sub_3140F3	
		short loc_31307D	
		sub_3140F3	
		short loc_31308C	
	call	sub_3140F3	
	and	eax, OFFFFh	
			101
			101

Heap Spraying on 64bit

	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
	eax, [ebp+var_84]
ousn	esi

- On 64bit heap spraying can't really be used to bypass ASLR
 - Good luck spraying anywhere near 2 (spoiler: that's ~18446744 terabytes)

	ly k	be use	d to
		short loc_31306D	
		esi	
	nuela	eax, [epp+arg_0]	
~	7 64	hytor	
	Lash	bytes	
		[ebp+arg_4]	
	puth		
	Ja 1	sub_314623	
		Short loc_31306D	
		short loc 31308F	
		sub_31411B	
		sub 3140F3	
		short loc_31307D	
		sub_3140F3	
		short loc_31308C	
		sub 3140F3	
	and	eax, OFFFFh	
			102
			A TYTER VERE AND 210TRO

Heap Exploitation

loc_31308C:

; CODE X. Tebp+var 41. eax

Heap Spraying on 64bit

- push edi call sub_314623 test eax, eax jz short loc_31306D cmp [ebp+arg_], 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
- On 64bit heap spraying can't really be used to bypass ASLR

 Good luck spraying anywhere near 2 (spoiler: that's ~18446744 terabytes)

	sub_31486A
	short loc_31306D
	<pre>eax, [ebp+arg_0]</pre>
	eax
54	hytoc
	Dyles
	[ebp+arg_4]
	sub_314623
	short loc_31306D
	[ebp+arg_0], esi
	short loc_31308F

 Targeted sprays are still useful in scenarios that you have a partial heap ptr overwrite or need to do some heap grooming

		loc_31307D:					
				sub_3140F3			
			and	eax, OFFFFh			
IBE - 04/07/2015	Heap Exploitation				103		
		loc_31308C:					
				[ebp+var 4], ea			

Heap Spray Payloads

MBE - 04/07/2015

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]

 Pretty common to spray some critical value for your exploit, fake objects, or ROP chains

		eax, [ebp+arg_0	
		[ebp+arg_4]	
		sub_314623	
		short loc_31306	
		[ebp+arg_0], es	
		short loc 31308	
313066:			
		sub_31411B	
31306D:			
		sub 3140F3	
		short loc 31307	
		sub 3140F3	
		short loc 31308	C
31307D:			
		sub_3140F3	
	and	eax, OFFFFh	
			104
31308C:			; CODE XREF: sub 312FD8

Lecture Overview

- Heap Overview
- Heap Exploitation
 - Heap Overflows
 - Use After Free
 - Heap Spraying
 - Metadata Corruption

		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	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
:_31306D:		
		sub_3140F3
		short loc_31307D
		sub_3140F3
		short loc_31308C
_31307D:		
		sub_3140F3
	and	eax, OFFFTh
		105
212000.		

MBE - 04/07/2015

Metadata Corruption

busn	631
push	esi
	<pre>eax, [ebp+var_84]</pre>
	short loc_313066
	<pre>eax, [ebp+var_84]</pre>
	<pre>eax, [ebp+var_70]</pre>
	short loc_313066
	<pre>[ebp+arg_0], ebx</pre>
	short loc_31306D
	sub_314623

- Metadata corruption based exploits involve corrupting heap metadata in such a way that you can use the allocator's internal functions to cause a controlled write of some sort
- Generally involves faking chunks, and abusing its different coalescing or unlinking processes

		loc 31307D:			
				sub_3140F3	
			and	eax, OFFFFh	
MBE - 04/07/2015	Heap Exploitation				106
		loc_31308C:			




Metadata Corruption

	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	e31

- The 'hello world' of heap metadata exploits is an example taught using the heap unlink() process when freeing a chunk
 - This is a dated and long since patched technique that is well documented of the answer to an and the and
- https://sploitfun.wordpress.loc_31306D:
 com/2015/02/26/heap-overflow-using to be and the sub_3140F3

		loc_31307D:			
				sub_3140F3	
			and	eax, OFFFFh	
MBE - 04/07/2015	Heap Exploitation				109
		loc_31308C:			
				Tebretvar 41 ea	



glibc Metadata Corruption

- <u>https://kitctf.de/writeups/0ctf2015/freenote/</u>
- <u>https://sploitfun.wordpress.com/2015/03/04/heap-overflow-using-malloc-maleficarum/</u>
- <u>http://acez.re/ctf-writeup-hitcon-ctf-2014-stkof-or-modern-</u> heap-overflow/

Heap Exploitation

- <u>http://wapiflapi.github.io/2014/11/17/hacklu-oreo-with-</u> ret2dl-resolve/
- <u>http://phrack.org/issues/66/10.html</u>

MBE - 04/07/2015

 <u>http://dl.packetstormsecurity.</u> <u>net/papers/attack/MallocMaleficarum.txt</u>

	sub 314623	
	short loc_31306	
	[ebp+arg_0], eb:	
inz.	short loc_31306	
1 077	eax, [ebp+var_7]	
gao	eax, [ebp+var_8	4]
	short loc_31306	
	eax, [ebp+var_8	4]
pusn	esi	
push		
te/		
	[ebp+arg_0], eas	
	sub_31486A	
$\Omega I / h$		rflow
<u>04/11</u>	eap-ove	
	esi	
	eax, [ebp+arg_0]	
<u>stkof</u>	<u>-or-moc</u>	<u>lern-</u>
	sub_314623	
	short loc_31306	D
klu-o	<u>reo-with</u>	
	SUD_31411B	
	3UD_5140F3	
	CdX, CdX	
	SHOFE LOC_313U/I	
	SUD_3140F3	
	SHOPE LOC_31308	
	aub 21/0E2	
Call	C1UPIC_ODE	
	eax, urffrn	
	eax, aud/uuuuh	141

