Exploit Mitigation Improvements in Windows 8

Ken Johnson, Matt Miller
Microsoft Security Engineering Center (MSEC)

Black Hat USA 2012

Acknowledgements

Many individuals have worked very hard to deliver the improvements we will discuss

Windows	Charles Chen, Greg Colombo, Jason Garms, Stephen Hufnagel, Arun Kishan, Joe Laughlin, Pavel Lebedynskiy, John Lin, Gov Maharaj, Hari Pulapaka, Pierre-Yves Santerre, Neeraj Singh, Evan Tice, Valeriy Tsuryk, Suma Uppuluri, Landy Wang, Matthew Woolman
CLR/Silverlight	Reid Borsuk, Jesse Collins, Jeffrey Cooperstein, Nick Kramer
Visual Studio	Jonathan Caves, Tanveer Gani, Mark Hall, Lawrence Joel, Louis Lafreniere, Mark Levine, Steve Lucco, Mark Roberts, Andre Vachon, YongKang Zhu
Internet Explorer	David Fields, Forbes Higman, Eric Lawrence, Zach Murphy, Justin Rogers
Microsoft Research	Richard Black, Miguel Castro, Manuel Costa, Ben Livshits, Jay Stokes, Ben Zorn
Microsoft Security Engineering Center	Eugene Bobukh, Tim Burrell, Thomas Garnier, Nitin Kumar Goel, Ken Johnson, John Lambert, Matt Miller, Dave Probert, Tony Rice, Richard Shupak, Julien Vanegue, Greg Wroblewski

Windows 8 Security Overview

Secure Development

The development of Windows 8 followed the Security Development Lifecycle (SDL) which defines best practices for secure software design, implementation, and testing.

Securing the Boot

Windows 8 includes new protections against root and boot kits. UEFI systems prevent malware from starting before Windows and protects the remainder of the boot process, including early launch antimalware software.

Securing the Core

Windows 8 includes numerous security features, such as mitigations, that make it difficult and costly for malware authors to develop reliable exploits for vulnerabilities.

Securing After the Boot

Windows Defender is shipped with every edition of Windows 8 and Internet Explorer's Application reputation features have been moved into the platform such that users are protected regardless of the browser they use.

Framing the problem with exploit economics

Attackers want to maximize ROI

Use sandboxing to protect user privacy & data

Find and eliminate vulnerability classes

Attacker Return = (Gains per use X Opportunities to use) - (Cost to acquire vulnerability + Cost to weaponize)

We want to minimize ROI

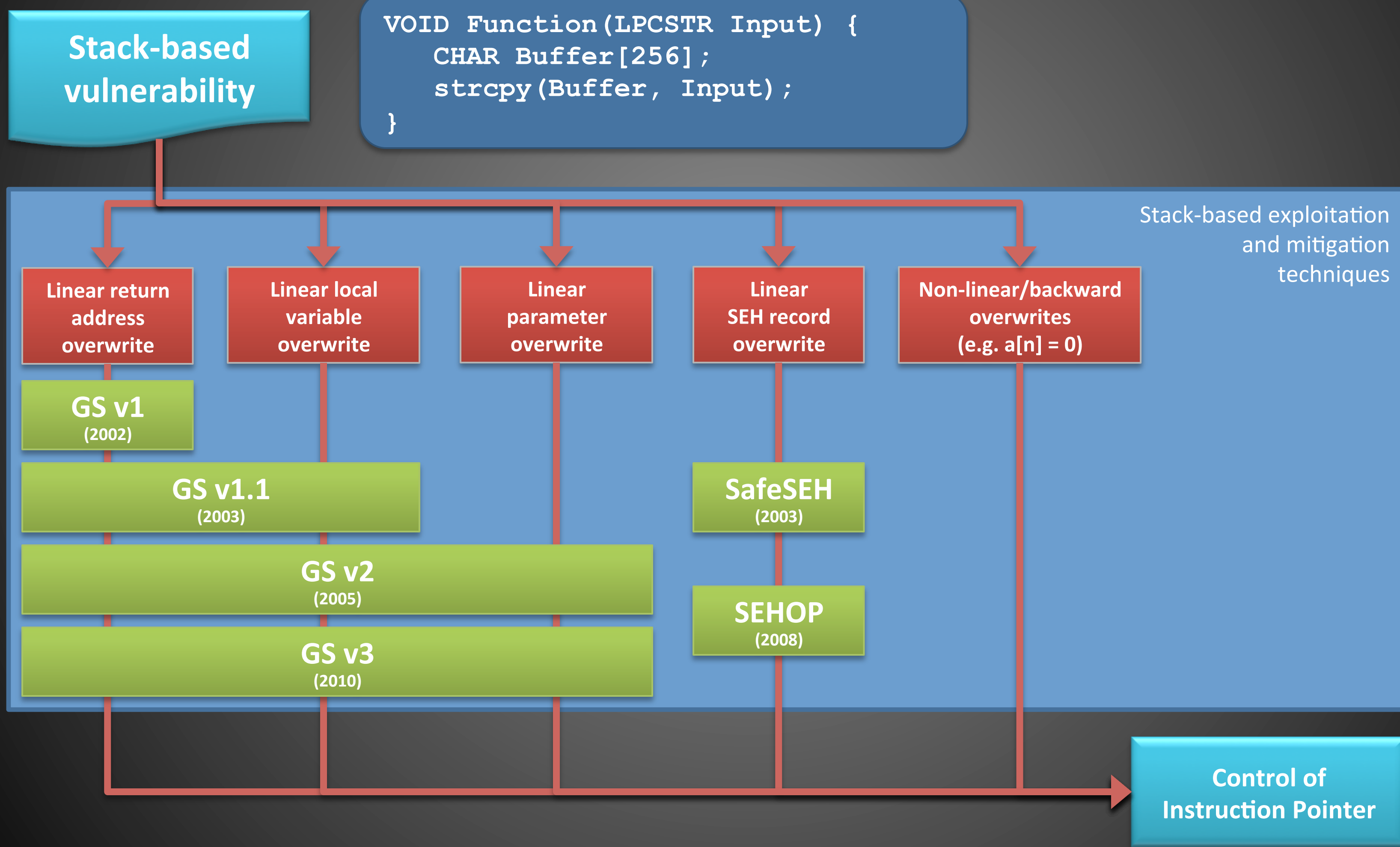
Use telemetry to minimize attack

Break cookbook techniques & make

Today we will describe exploit mitigation improvements in Windows 8 that increase the cost of developing reliable exploits

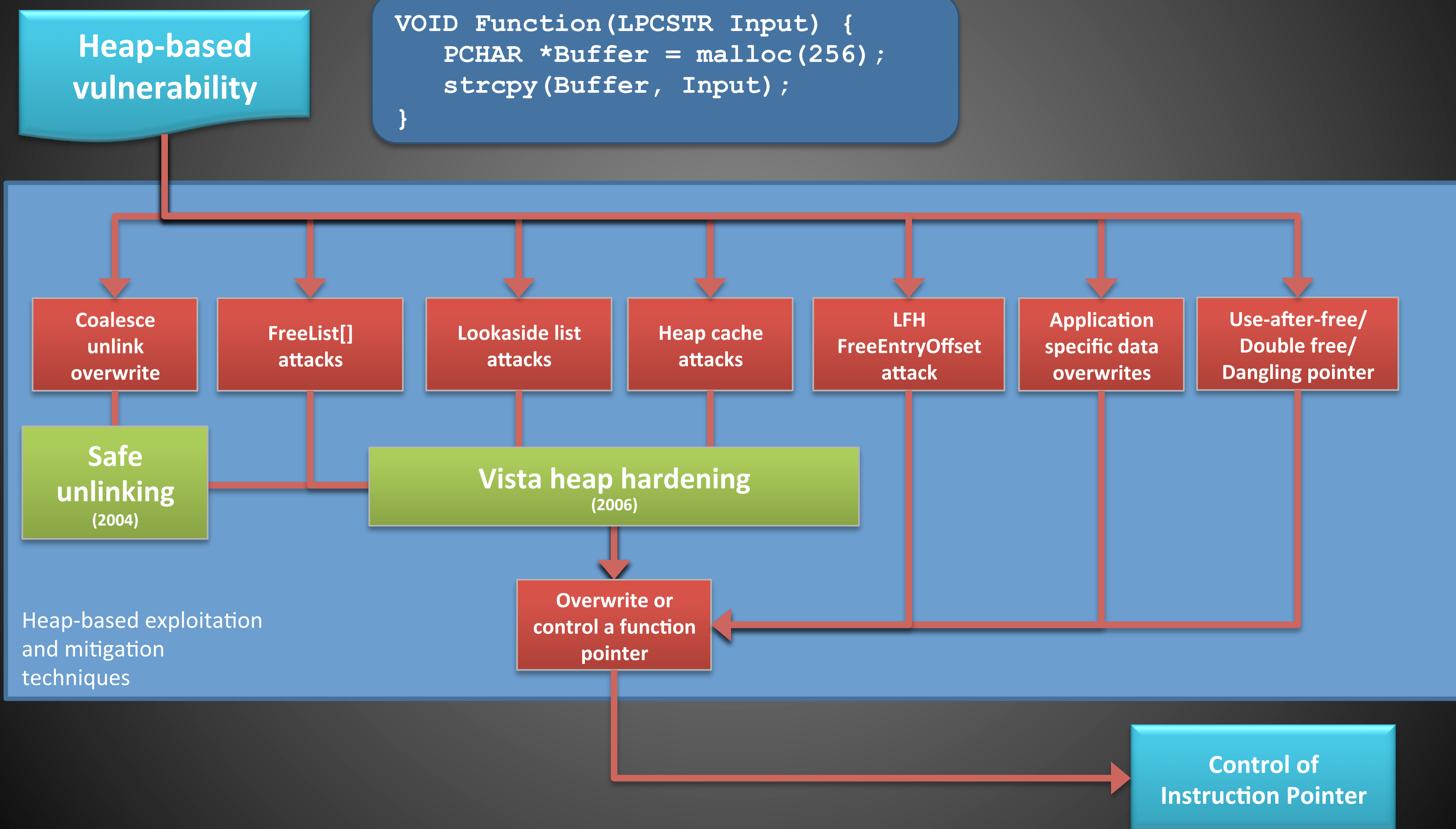
History of exploit mitigations on Windows

```
VOID Function(LPCSTR Input) {
   CHAR Buffer[256];
   strcpy(Buffer, Input);
}
```

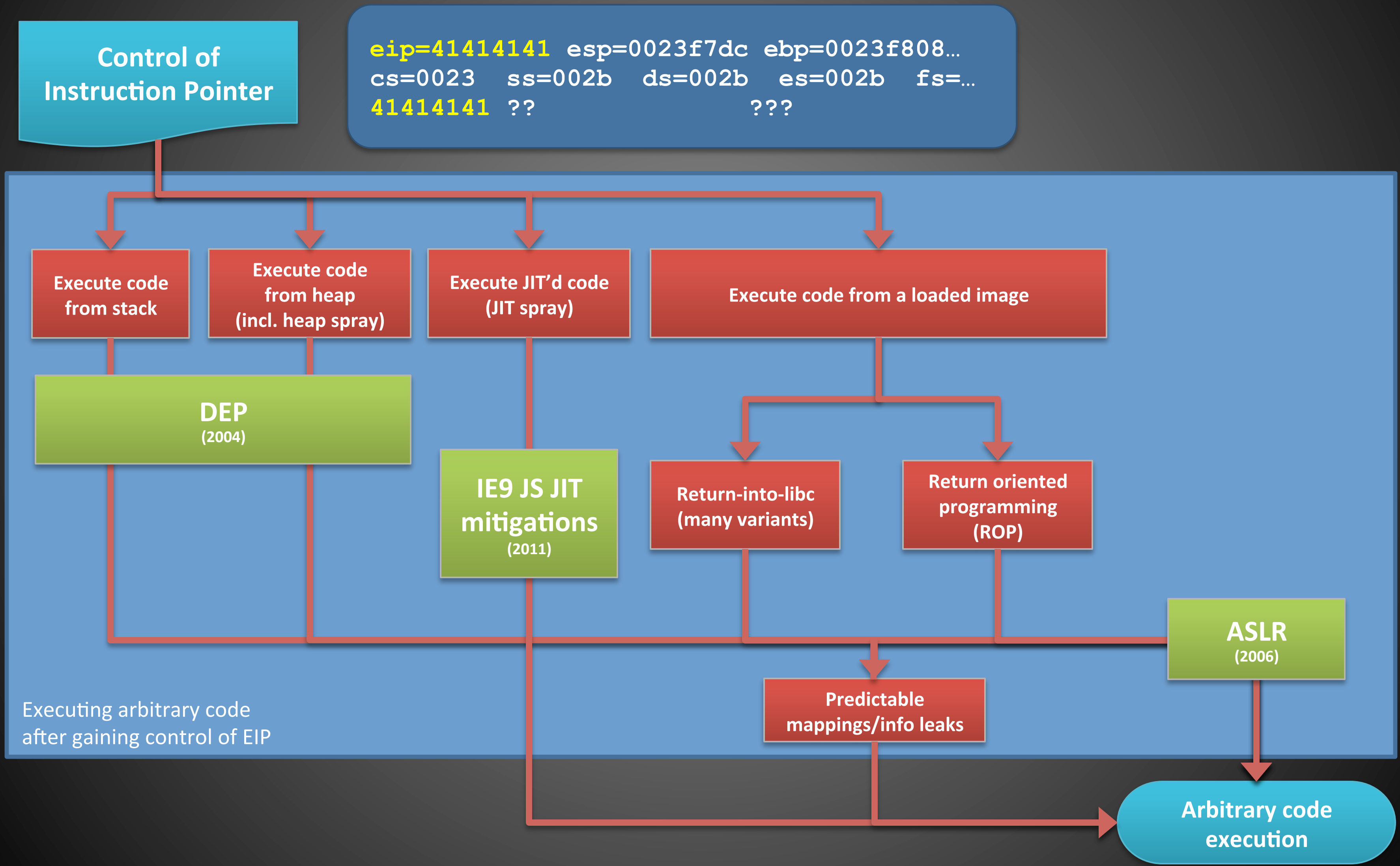


History of exploit mitigations on Windows

```
VOID Function(LPCSTR Input) {
   PCHAR *Buffer = malloc(256);
   strcpy(Buffer, Input);
}
```



History of exploit mitigations on Windows



The state of memory safety exploits

Most systems are not compromised by exploits	• About 6% of MSRT detections were likely caused by exploits [29] • Updates were available for more than a year for most of the exploited issues [29]
Most exploits target third party applications	• 11 of 13 CVEs targeted by popular exploit kits in 2011 were for issues in non-Microsoft applications [27]
Most exploits target older versions of Windows (e.g. XP)	• Only 5% of 184 sampled exploits succeeded on Windows 7 [28] • ASLR and other mitigations in Windows 7 make exploitation costly [30]
Most exploits fail when mitigations are enabled	• 14 of 19 exploits from popular exploit kits fail with DEP enabled [27] • 89% of 184 sampled exploits failed with EMET enabled on XP [28]
Exploits that bypass mitigations & target the latest products do exist	• Zero-day issues were exploited in sophisticated attacks (Stuxnet, Duqu) • Exploits were written for Chrome and IE9 for Pwn2Own 2012

Bottom line: we must continue to increase the cost of exploitation for attackers

Objectives & focus areas in Windows 8

Objectives

Mitigate entire classes of vulnerabilities

Break known exploitation techniques

Increase the cost of exploitation in key domains

- Code Generation Security
- Address Space Layout Randomization
- Windows Heap
- Windows Kernel
- Default Settings

Enhanced /GS, range checks, sealed optimization, and virtual table guard

CODE GENERATION

Enhanced /GS

- Enhanced GS stack buffer overrun protection
  - Released with Visual Studio 2010 [1]
  - Windows 8 is built with this enabled
- GS heuristics now protect more functions
  - Non-pointer arrays and POD structures
- GS optimization removes unnecessary checks
  - Safety proof means no check is needed
- Closes gaps in protection
  - MS04-035, MS06-054, MS07-017 (ANI)

Range Checks

- Compiler-inserted array bounds check (via /GS)

```
CHAR Buffer[256];
UINT i; // possibly attacker controlled

if (i >= ARRAYSIZE(Buffer)) {     ← compiler inserted
   __report_rangecheckfailure();
}

Buffer[i] = 0;
```

- Completely mitigates certain vulnerabilities
  - CVE-2009-2512, CVE-2010-2555
- Bounds check insertion is limited to specific scenarios
  - Assignment of NUL to a fixed-size stack/global array

Sealed optimization

- Optimization for "sealed" C++ types & methods

```
class COptionElement sealed : public CElement
{
    DECLARE_CLASS_TYPES(COptionElement, CElement)
```

- Virtual method calls become direct calls

```
COptionElement *optionElement;
optionElement->IsEnabled();
```

Without sealed

```
mov  rax,qword ptr [rcx]
call qword ptr [rax+920h]
```

With sealed

```
call CElement::IsEnabled
```

- Eliminating indirect calls reduces exploitation attack surface
  - Helps mitigate vulnerabilities like CVE-2011-1996
  - Devirtualized ~4,500 calls in mshtml.dll and ~13,000 in mso.dll

Virtual Table Guard

Probabilistic mitigation for vulnerabilities that enable vtable ptr corruption

IE10 has enabled this for a handful of key classes in mshtml.dll

Enabled by adding an annotation to a C++ class

CElement::`vftable'
VirtualMethod1
VirtualMethod2
VirtualMethod3
VirtualMethod4
…
_vtguard

Extra entry added to vtable.
ASLR makes this entry's value unknown to the attacker.

```
mshtml!CElement::Doc:
63700e70 mov     eax,dword ptr [ecx]
63700e72 cmp     [eax+1B8h],offset mshtml!_vtguard
63700e7c jne     mshtml!CElement::Doc+0x18 (63700e88)
63700e7e call    dword ptr [eax+0ACh]
63700e84 mov     eax,dword ptr [eax+0Ch]
63700e87 ret
63700e88 call    mshtml!__report_gsfailure
```

Check added at virtual method call sites.
If vtable[vtguard_vte] != vtguard then terminate the process.

Force ASLR, bottom-up/top-down randomization, and high entropy

ADDRESS SPACE LAYOUT RANDOMIZATION

Retrospective: ASLR

- ASLR was first introduced in Windows Vista
  - Led to a big shift in attacker mentality
- Attackers now depend on gaps in ASLR
  - EXEs/DLLs not linked with /DYNAMICBASE [2]
  - Address space spraying (heap/JIT) [3]
  - Predictable memory regions [4]
  - Information disclosures [5]
- ASLR has been substantially improved in Windows 8

Force ASLR

- Many exploits depend on non-ASLR DLLs

```
0:000> !dh dirapi
...
4B4C111C time date stamp Mon Jan 11 22:05:16 2010
...
    1000 base of code
         ----- new -----
68000000 image base
    1000 section alignment
...
00001000 size of heap commit
       0  DLL characteristics
  187A80 [    402F] address [size] of Export Directory
  1869F4 [      C8] address [size] of Import Directory
  19B000 [   12178] address [size] of Resource Directory
```

Images are not randomized unless the DYNAMIC_BASE bit is set

- Processes can now force non-ASLR images to be randomized
  – Behaves as if an image's preferred base is not available
  – Bottom-up randomization provides entropy for these images
- Processes must opt-in to receive this behavior
  – Also supported on Windows 7 with KB2639308 installed

Outcome: attackers can no longer rely on non-ASLR images

Bottom-up & top-down randomization

Windows 7

- Heaps and stacks are randomized
- PEBs/TEBs are randomized, but with limited entropy
- VirtualAlloc and MapViewOfFile are not randomized
- Predictable memory regions can exist as a result

Windows 8

- All bottom-up/top-down allocations are randomized
- Accomplished by biasing start address of allocations
- PEBs/TEBs now receive much more entropy
- Both are opt-in (EXE must be dynamicbase)

Outcome: predictable memory regions have been eliminated

High entropy ASLR for 64-bit processes

ASLR in Windows 8 takes advantage of the large address space (8TB) of 64-bit processes

High entropy bottom-up randomization

- 1 TB of variance in bottom-up start address
- Breaks traditional address space spraying (heap/JIT)
- Processes must opt-in to receive this behavior

High entropy top-down randomization

- 8 GB of variance in top-down start address
- Automatically enabled if top-down randomization is on

High entropy image randomization

- Images based above 4 GB receive more entropy
- All system images have been moved above 4 GB

Outcome: probability of guessing an address is decreased and disclosures of memory addresses must include more than the low 32 bits

ASLR entropy improvements

Entropy (in bits) by region	Windows 7		Windows 8		
	32-bit	64-bit	32-bit	64-bit	64-bit (HE)
Bottom-up allocations (opt-in)	0	0	8	8	24
Stacks	14	14	17	17	33
Heaps	5	5	8	8	24
Top-down allocations (opt-in)	0	0	8	17	17
PEBs/TEBs	4	4	8	17	17
EXE images	8	8	8	17*	17*
DLL images	8	8	8	19*	19*
Non-ASLR DLL images (opt-in)	0	0	8	8	24

* 64-bit DLLs based below 4GB receive 14 bits, EXEs below 4GB receive 8 bits

ASLR entropy is the same for both 32-bit and 64-bit processes on Windows 7

64-bit processes receive much more entropy on Windows 8, especially with high entropy (HE) enabled

Removal of information disclosure vectors

- Information disclosures can be used to bypass ASLR
- Disclosure via an arbitrary read is now less reliable
  - Predictable mappings have been eliminated
- SharedUserData is still predictable, but less useful
  - Image pointers have been removed, breaking known techniques [4,6]

```
0:000> u ntdll!NtWriteVirtualMemory L6
ntdll!NtWriteVirtualMemory:
6a214724 b802000000      mov     eax,2
6a214729 e803000000      call    ntdll!NtWriteVirtualMemory+0xd
(6a214731)
6a21472e c21400          ret     14h
6a214731 8bd4            mov     edx,esp
6a214733 0f34            sysenter
6a214735 c3              ret
0:000> dd 7ffe0300 L1
7ffe0300  00000000
```

Integrity checks, guard pages, and allocation order randomization

WINDOWS HEAP

Retrospective: Windows Heap

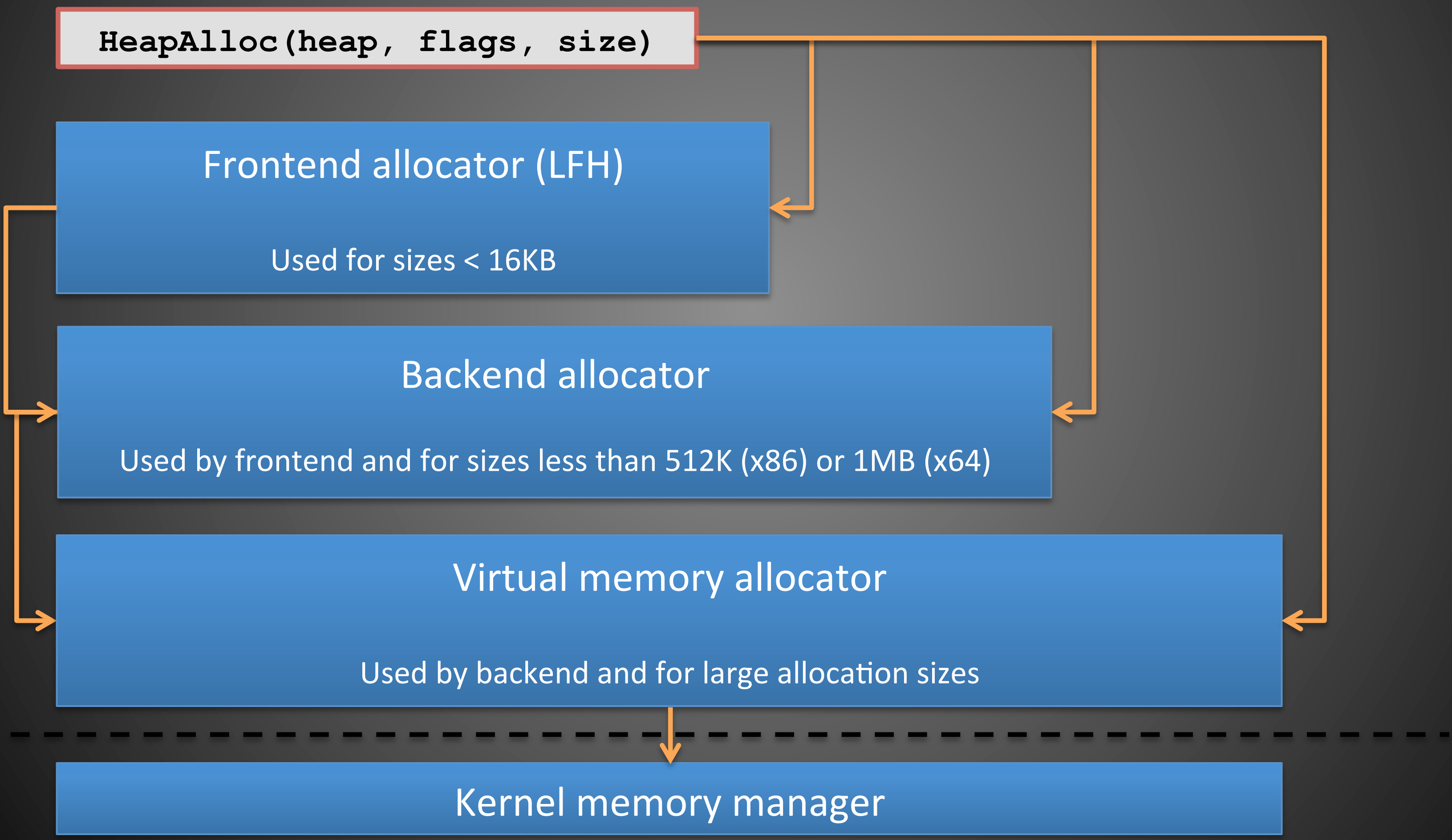
- Windows Vista heap hardening was very effective [11]
  - Only one documented exploit that corrupts metadata [9]

- New attacks have been proposed by researchers
  - Corrupting the HEAP data structure [7]
  - LFH bucket overwrite [7]
  - LFH FreeEntryOffset corruption and depth desync [8,12]

- Real-world exploits target app data on the heap [10]
  - No heap safeguards exist today for this

Windows 8 heap architecture

The general design of the Windows heap is unchanged in Windows 8



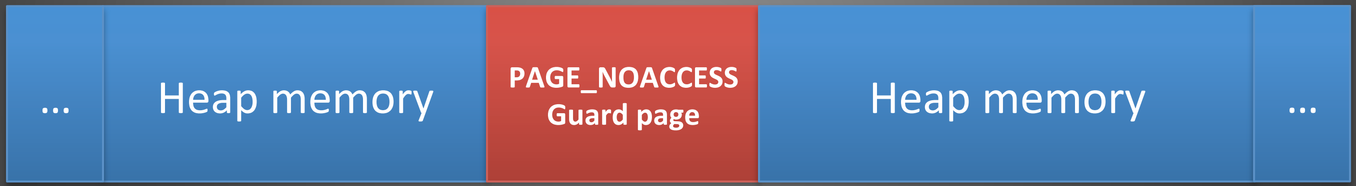
LFH design changes & integrity checks

Change in Windows 8	Impact
LFH is now a bitmap-based allocator	LinkOffset corruption no longer possible [8]
Multiple catch-all EH blocks removed	Exceptions are no longer swallowed
HEAP handle can no longer be freed	Prevents attacks that try to corrupt HEAP handle state [7]
HEAP CommitRoutine encoded with global key	Prevents attacks that enable reliable control of the CommitRoutine pointer [7]
Validation of extended block header	Prevents unintended free of in-use heap blocks [7]
Busy blocks cannot be allocated	Prevents various attacks that reallocate an in-use block [8,11]
Heap encoding is now enabled in kernel mode	Better protection of heap entry headers [19]

Outcome: attacking metadata used by the heap is now even more difficult

Guard pages

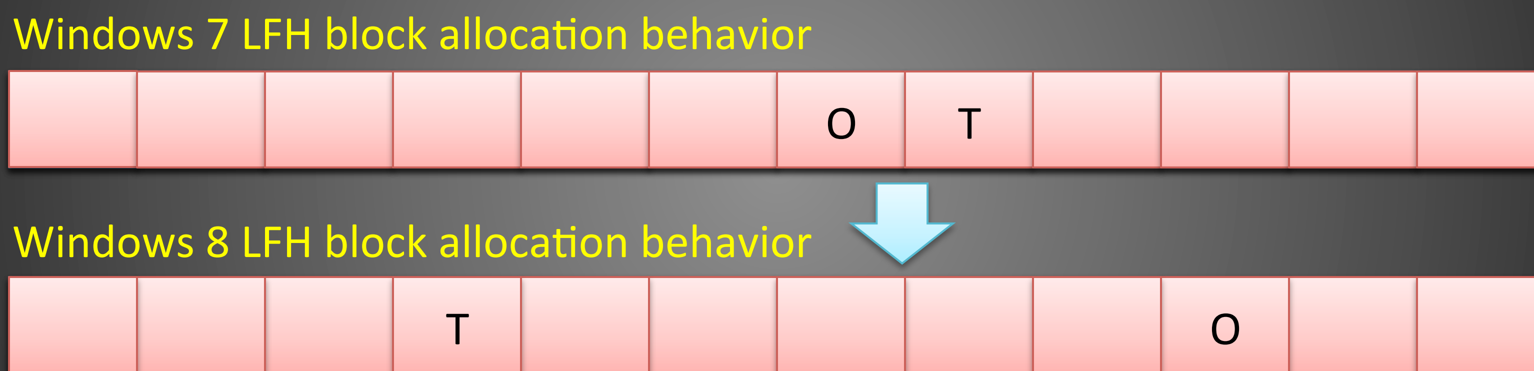
- Guard pages are now used to partition the heap
  - Designed to prevent & localize corruption in some cases
  - Touching a guard page results in an exception



- Insertion points for guard pages are constrained
  - Large allocations
  - Heap segments
  - Max-sized LFH subsegments (probabilistic on 32-bit)

Allocation order randomization

- Allocation order is now nondeterministic (LFH only)
  - Exploits often rely on surgical heap layout manipulation [10]
  - Randomization makes heap normalization unreliable



- Maximizing reliability is more challenging
  - Application-specific and vulnerability-specific
  - May require corrupting more data (increasing instability)
  - May require allocating more data (triggering guard pages)

DEP, ASLR, SMEP/PXN, NULL dereference protection, and pool integrity checks

WINDOWS KERNEL

Retrospective: Windows Kernel

- Kernel vulnerabilities have been less targeted
  - Relatively few remote kernel exploits exist
  - User mode exploitation is better researched
- Attack focus is shifting more toward the kernel
  - Interest in sandbox escapes is increasing
  - Local kernel exploitation techniques well-understood
  - New kernel pool attacks have been proposed [13]
  - Sophisticated remote kernel exploits exist [14,21]

DEP is broadly enabled in the Windows 8 kernel

Many kernel memory regions were unnecessarily executable in Windows 7 and prior

	x86 (PAE)		x64		ARM
	Win7	Win8	Win7	Win8	Win8
Paged pool	X	X	NX	NX	NX
Non-paged pool	X	X	X	X	X
Non-paged pool (NX)	N/A	NX	N/A	NX	NX
Session pool	X	X	NX	NX	NX
Image data sections	X	X	NX	NX	NX
Kernel stacks	NX	NX	NX	NX	NX
Idle/DPC/Initial stacks	X	NX	X	NX	NX
Page table pages	X	NX	X	NX	NX
PFN database	X	NX	X	NX	NX
System cache	X	NX	X	NX	NX
Shared user data	X	NX	X	NX	NX
HAL heap	X	NX	X	NX	NX

X = executable   NX = non-executable

Windows 8 introduces NonPagedPoolNx for non-executable non-paged pool allocations (default on ARM)

NX HAL heap and NonPagedPoolNx break the assumptions of exploits for MS09-050

Kernel ASLR improvements

- Kernel ASLR was first added in Server 2008 RTM
  - 4 bits of entropy for drivers, 5 bits for NTOS/HAL
  - Driver entropy was improved in Windows 7
- Entropy has been further improved in Windows 8
  - Biasing of kernel segment base
  - NTOS/HAL receive 22 bits (64-bit) and 12 bits (32-bit)
  - Various boot regions also randomized (P0 idle stack)

Support for SMEP/PXN

- New processor security feature
  - Prevents supervisor from executing code in user pages
  - Most exploits for local kernel EOPs rely on this today
  - Requires Intel Ivy Bridge or ARM with PXN support
- SMEP/PXN + DEP make exploitation more difficult
  - Strong mitigation for some issues (CVE-2010-2743 from Stuxnet)
  - Attackers need to leverage code in kernel images [15]

NULL dereference protection

- Kernel NULL dereferences are a common issue
  - Examples include MS08-025, MS08-061, MS09-001
- Local exploitation is generally straightforward
  - NULL is part of the user mode address space
  - Kernel currently allows user processes to map NULL page
- Windows 8 prohibits mapping of the first 64K
  - All kernel NULL dereferences become a DoS (not EoP)
  - NTVDM has been disabled by default as well
  - Enabling NTVDM will disable NULL dereference protection

Kernel pool integrity checks

- The kernel pool allocator is similar to the heap
  - Implementation is very different, though
- New integrity checks block various attacks [13]
  - Process quota pointer encoding
  - Lookaside, delay free, and pool page cookies
  - PoolIndex bounds check
  - Additional safe unlinking checks

Other improvements

- Safe unlinking has been enabled globally
  - Previously only used in the heap and kernel pool
  - Now applies to all usage of LIST_ENTRY, closing known gaps [16]
  - New "FastFail" mechanism enables rapid & safe process termination
- Improved entropy for GS and ASLR
  - Use of PRNG seeded by TPM/RDRAND/other sources
  - Hardcoded GS initialization is overridden by the OS
  - Addresses weaknesses described in attack research [17,18]
- Object manager hardened against reference count overflows
- Resolved kernel information disclosure via certain system calls [22]

ARM, Inbox, Windows 8 style apps, IE 10, the new Office, and other applications

DEFAULT SETTINGS

ARM default settings

All applicable mitigations are enabled on ARM

DEP	On
ASLR (images)	On
ASLR (force relocate)	N/A (all images are ASLR)
ASLR (bottom-up)	On
ASLR (top-down)	On
ASLR (high entropy)	N/A (not 64-bit)
SEHOP	N/A (not needed)
Heap termination	On
Kernel NULL dereference	On
Kernel SMEP	On

ARM PE images must opt-in to DEP and ASLR

Kernel will fail to load images that do not

Lack of application compatibility concerns enables us to be more aggressive

Application default settings

All applicable mitigations are enabled for Windows 8 UI style apps

Default settings for Windows 8 client	32 bit (x86)				64 bit (x64)			
	Win8 UI apps	Inbox	IE10	Default	Win8 UI apps	Inbox	IE10	Default
DEP	On	On	On	OptIn	On	On	On	OptIn
ASLR (images)	On	On	On	OptIn	On	On	On	OptIn
ASLR (force relocate)	On	OptIn	On	OptIn	On	OptIn	On	OptIn
ASLR (bottom-up)	On	On	On	OptIn	On	On	On	OptIn
ASLR (top-down)	On	On	On	OptIn	On	On	On	OptIn
ASLR (high entropy)	N/A	N/A	N/A	N/A	On	On	On	OptIn
SEHOP	On	On	On	OptIn	N/A	N/A	N/A	N/A
Heap termination	On	On	On	OptIn	On	On	On	On

Internet Explorer 10 and the new Office also enable all applicable mitigations

Enabling opt-in mitigations

Opt-in methods

- "MitigationOptions" Image File Execution Option (IFEO)
- Process creation attribute (via UpdateProcThreadAttribute)
- SetProcessMitigationPolicy API
- Linker flag

Opt-in mitigation	IFEO	Proc Attr	API	Linker flag
Bottom-up randomization	Yes	Yes	No	/DYNAMICBASE (on EXE)
Top-down randomization	No	No	No	/DYNAMICBASE (on EXE)
Bottom-up randomization (high entropy)	Yes	Yes	No	/HIGHENTROPYVA (on EXE)
ASLR	No	No	No	/DYNAMICBASE
Force ASLR	Yes	Yes	Yes	None
DEP	Yes	Yes	Yes	/NXCOMPAT (on EXE)
SEHOP	Yes	Yes	No	None*
Heap termination	Yes	Yes	Yes	None*

* EXEs with a subsystem version >= 6.2 will automatically enable these mitigations

Expectations for exploits on Windows 8

- Writing exploits for Windows 8 will be very costly
  - Some vulnerability classes are now entirely mitigated
  - Many attack techniques are now broken or unreliable

- Attackers will likely focus their attention on
  - Desktop apps that do not enable all applicable mitigations
  - Desktop apps running on previous versions of Windows
  - Refining methods of disclosing address space information
  - Researching new exploitation techniques [20]

- We will continue to evolve our mitigation technologies

Call to action

- Upgrade to Windows 8 and IE 10 ☺
  - 64-bit is best from a mitigations perspective
  - Enable "Enhanced Protected Mode" for IE 10
- Software vendors
  - Build your applications with Visual Studio 2012 [31]
  - Enable new opt-in mitigations
- Driver writers
  - Port your drivers to use NonPagedPoolNx

Questions?

Contact us at switech@microsoft.com

Were you fascinated by the topics discussed in this presentation?

We are hiring.

http://www.microsoft-careers.com/go/Trustworthy-Computing-Jobs/194701/

Microsoft®

Be what's next.™

References

1. Enhanced GS in Visual Studio 2010. Tim Burrell. March, 2009. http://blogs.technet.com/b/srd/archive/2009/03/20/enhanced-gs-in-visual-studio-2010.aspx
2. Adobe CoolType SING Table “uniqueName” Stack Buffer Overflow Exploit. Metasploit. Sep, 2010. http://dev.metasploit.com/redmine/projects/framework/repository/revisions/master/entry/modules/exploits/windows/browser/adobe_cooltype_sing.rb
3. Interpreter Exploitation: Pointer Inference and JIT Spraying. Dionysus Blazakis. Black Hat Federal, 2010. http://www.semantiscope.com/research/BHDC2010/BHDC-2010-Paper.pdf
4. Defeat Windows 7 Browser Memory Protection. XiaBo Chen, Jun Xie. XCon 2010.
5. Memory Retrieval Vulnerabilities. Derek Soeder. 2006. http://www.eeye.com/eEyeDigitalSecurity/media/ResearchPapers/eeyeMRV-Oct2006.pdf
6. Win32 ASLR round 2. Justin Ferguson, March, 2010. http://nietmenja.blogspot.com/2010/03/win32-aslr-round-2.html
7. Attacking the Vista Heap. Ben Hawkes. Nov, 2008. http://www.lateralsecurity.com/downloads/hawkes_ruxcon-nov-2008.pdf
8. Understanding the Low Fragmentation Heap. Chris Valasek. July, 2010. http://illmatics.com/Understanding_the_LFH_Slides.pdf.
9. Modern Heap Exploitation using the Low Fragmentation Heap. Chris Valasek. 2011.
10. Heap Feng Shui in JavaScript. Alex Sotirov. 2007. http://www.phreedom.org/research/heap-feng-shui/heap-feng-shui.html
11. Preventing the exploitation of user mode heap corruption vulnerabilities. Matt Miller. 2009. http://blogs.technet.com/b/srd/archive/2009/08/04/preventing-the-exploitation-of-user-mode-heap-corruption-vulnerabilities.aspx
12. Ghost in the Windows 7 Allocator. Steven Seeley, 2012. http://conference.hitb.org/hitbsecconf2012ams/materials/D2T2%20-%20Steven%20Seeley%20-%20Ghost%20In%20the%20Windows%207%20Allocator.pdf

References

13. Kernel Pool Exploitation on Windows 7. Tarjei Mandt. Black Hat DC, 2011. https://media.blackhat.com/bh-dc-11/Mandt/BlackHat_DC_2011_Mandt_kernelpool-wp.pdf.
14. 351 packets from trampoline. Piotr Bania. October, 2009. http://blog.piotrbania.com/2009/10/351-packets-from-trampoline.html.
15. SMEP: What is it, and how to beat it on Windows. Mateusz Jurczyk, Gynvael Coldwind. June, 2011. http://j00ru.vexillium.org/?p=783.
16. VEH. Ben Hawkes. 2011. http://sota.gen.nz/veh/.
17. Windows Kernel-mode GS Cookies and 1 bit of entropy. Mateusz Jurczyk, Gynvael Coldwind. January, 2011. http://j00ru.vexillium.org/?p=690.
18. Reducing the effective entropy of GS cookies. Skape. May, 2007. http://www.uninformed.org/?v=7&a=2&t=sumry.
19. Kernel Attacks through User-Mode Callbacks. Tarjei Mandt. Black Hat USA, 2011. http://mista.nu/research/mandt-win32k-slides.pdf.
20. Windows 8 Heap Internals. Chris Valasek and Tarjei Mandt. Black Hat USA, 2012.
21. More information on MS11-087. Chengyun Chu, Jonathan Ness. December, 2011. http://blogs.technet.com/b/srd/archive/2011/12/13/more-information-on-ms11-087.aspx.
22. Subtle information disclosure in win32k.sys syscall return values. Mateusz Jurczyk. May, 2011. http://j00ru.vexillium.org/?p=762.

References

26. On the effectiveness of DEP and ASLR. Microsoft. December, 2010. http://blogs.technet.com/b/srd/archive/2010/12/08/on-the-effectiveness-of-dep-and-aslr.aspx.
27. The Exploit Intelligence Project v2. Dan Guido. December, 2011. http://vimeo.com/31548167.
28. Microsoft Security Intelligence Report (SIR): Volume 12. Microsoft. April, 2012. http://www.microsoft.com/en-us/download/details.aspx?id=29569.
29. Microsoft Security Intelligence Report (SIR): Volume 11. Microsoft. October, 2011. http://www.microsoft.com/en-us/download/details.aspx?id=27605.
30. Mitigating Software Vulnerabilities. Microsoft. July, 2011. http://www.microsoft.com/en-us/download/details.aspx?id=26788.
31. Compiler Security Enhancements in Visual Studio 11. Tim Burrell. December, 2011. http://blogs.msdn.com/b/sdl/archive/2011/12/02/security.aspx.