Windows Exploit Mitigations

Content

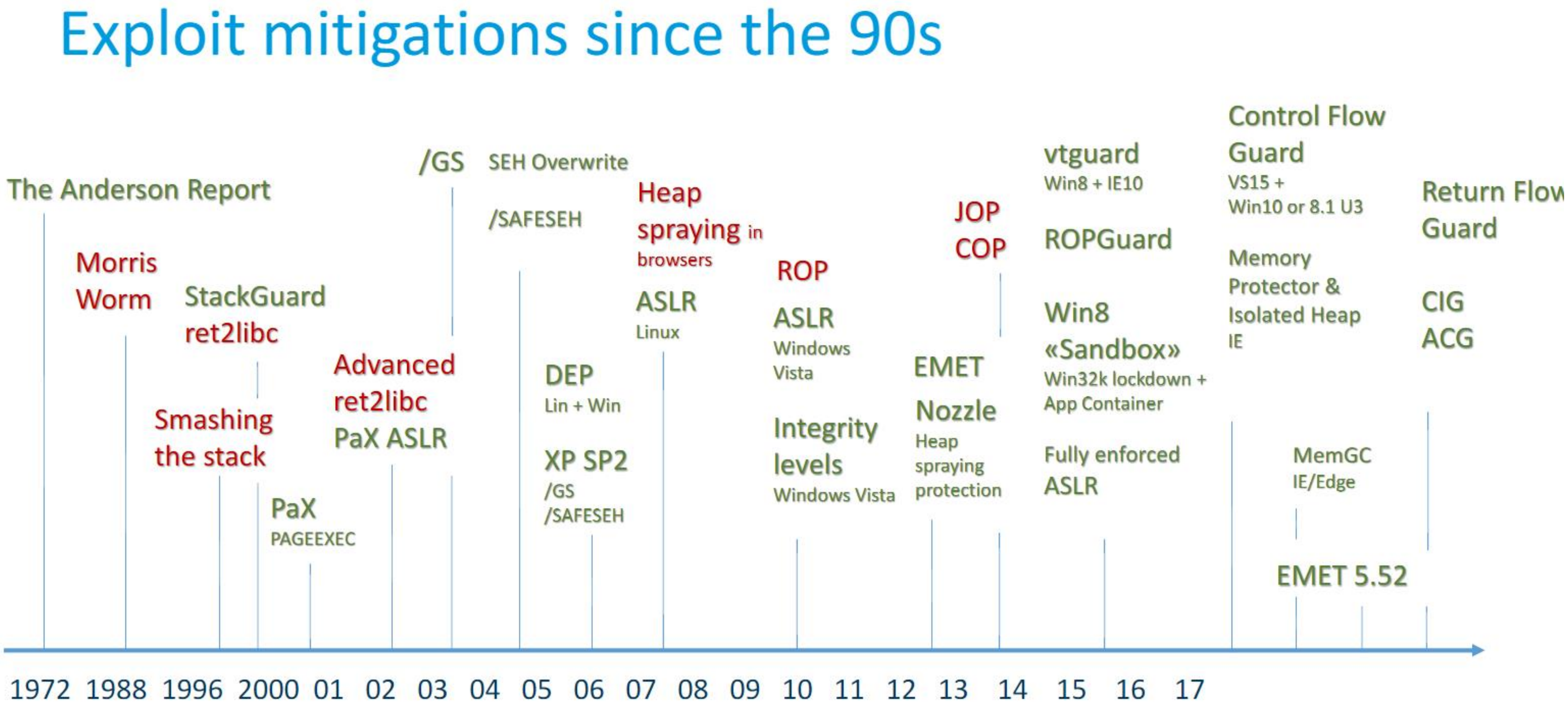
Slides based on excellent presentation:

- “Modern Exploit Mitigations”, Swiss CyberStorm 2017
- “Compilers, Memory Errors and Hardening Techniques”
  - https://www.ethz.ch/content/dam/ethz/special-interest/infk/inst-cs/lst-dam/documents/Education/Classes/Spring2016/2810_Advanced_Compiler_Design/Slides/20160518_advanced_compiler_design.pdf

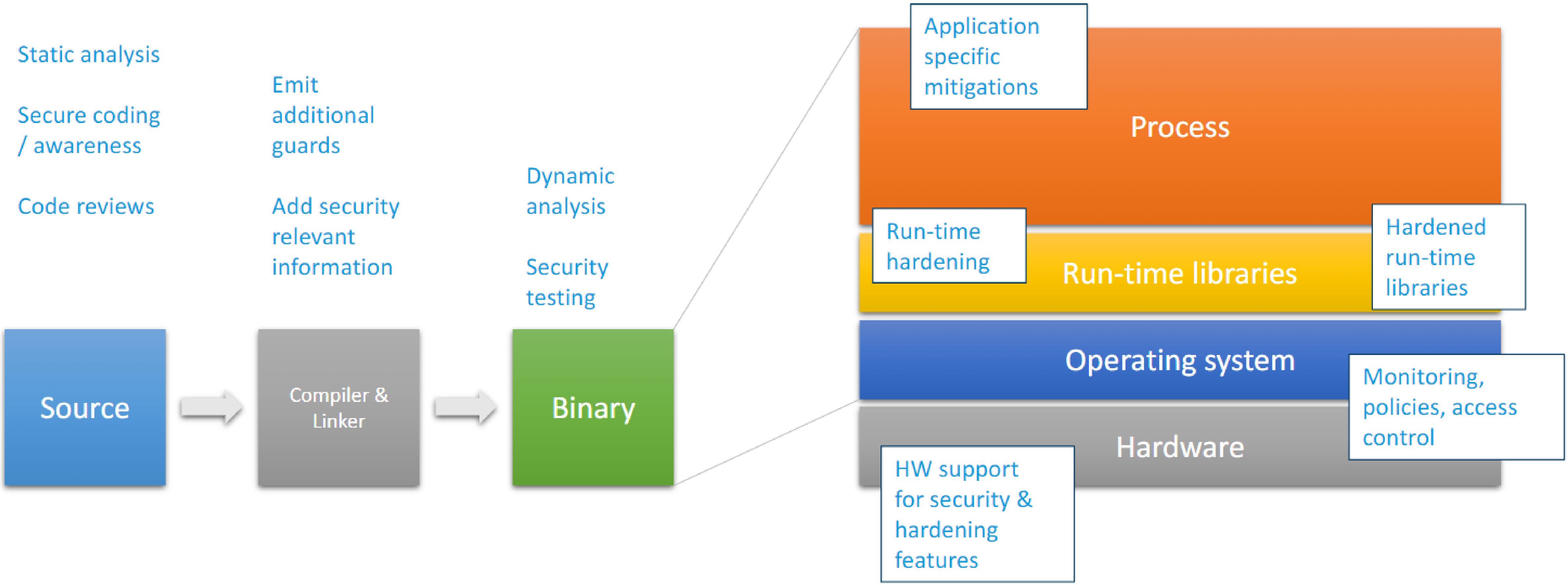
Both by:

- xorlab (ETH Spinoff),
- Matthias Ganz & Antonio Berresi

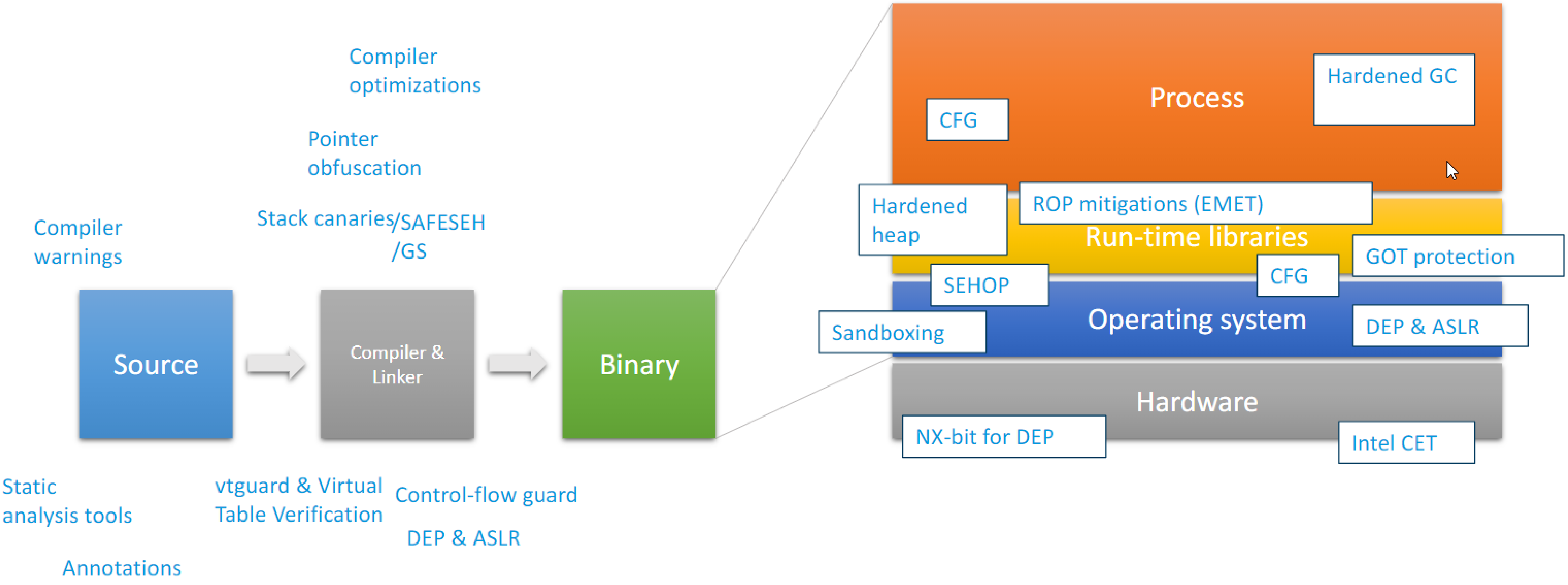
Exploit Mitigations



Hardening value chain



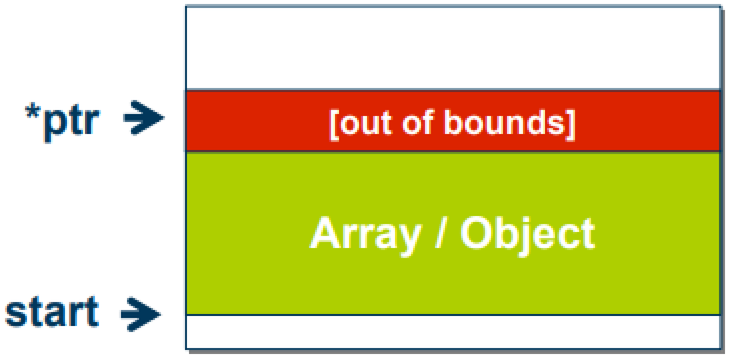
Hardening value chain



Spatial vs. Temporal

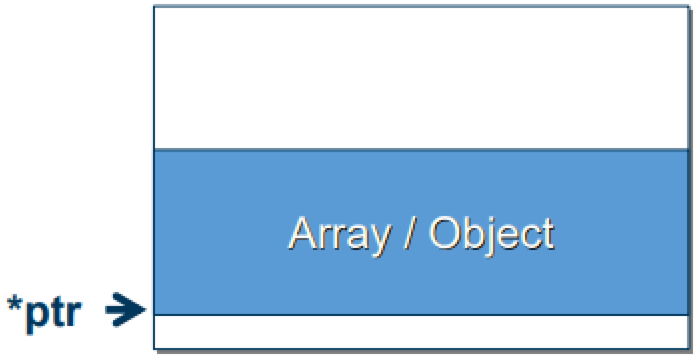
Types of memory errors

Spatial error



- De-reference pointer that is out of bounds

Temporal error



- De-reference pointer to freed memory

Types of bugs

- Out-of-bounds bugs / Buffer overflows
  - On stack or heap
- Dangling pointer / Use-after-free
- Integer bugs, signedness bugs
- Format string bugs
- Uninitialized memory
- NULL pointer dereference

Attack types

- Code corruption attack
- **Control-flow hijack attack**
- Data-only attack
- Information leak

Windows Exploit Mitigations

Some statements:

- “Windows is insecure”
- “Firefox is more secure than IE”

In respect of memory corruptions – Are these statements (still) true?

Windows Exploit Mitigation

Stack Canaries

Windows: Stack Canary

Stack Canaries

- Integrated in Visual Studio
- /gs
- Since Visual Studio 2002
- Deployed in: XP SP2

Version

- GS v1 (2002)
- GS v1.1 (2003)
- GS v2 (2005)
- GS v3 (2010)

Windows Exploit

SEH / AntiSEH

Windows: SEH

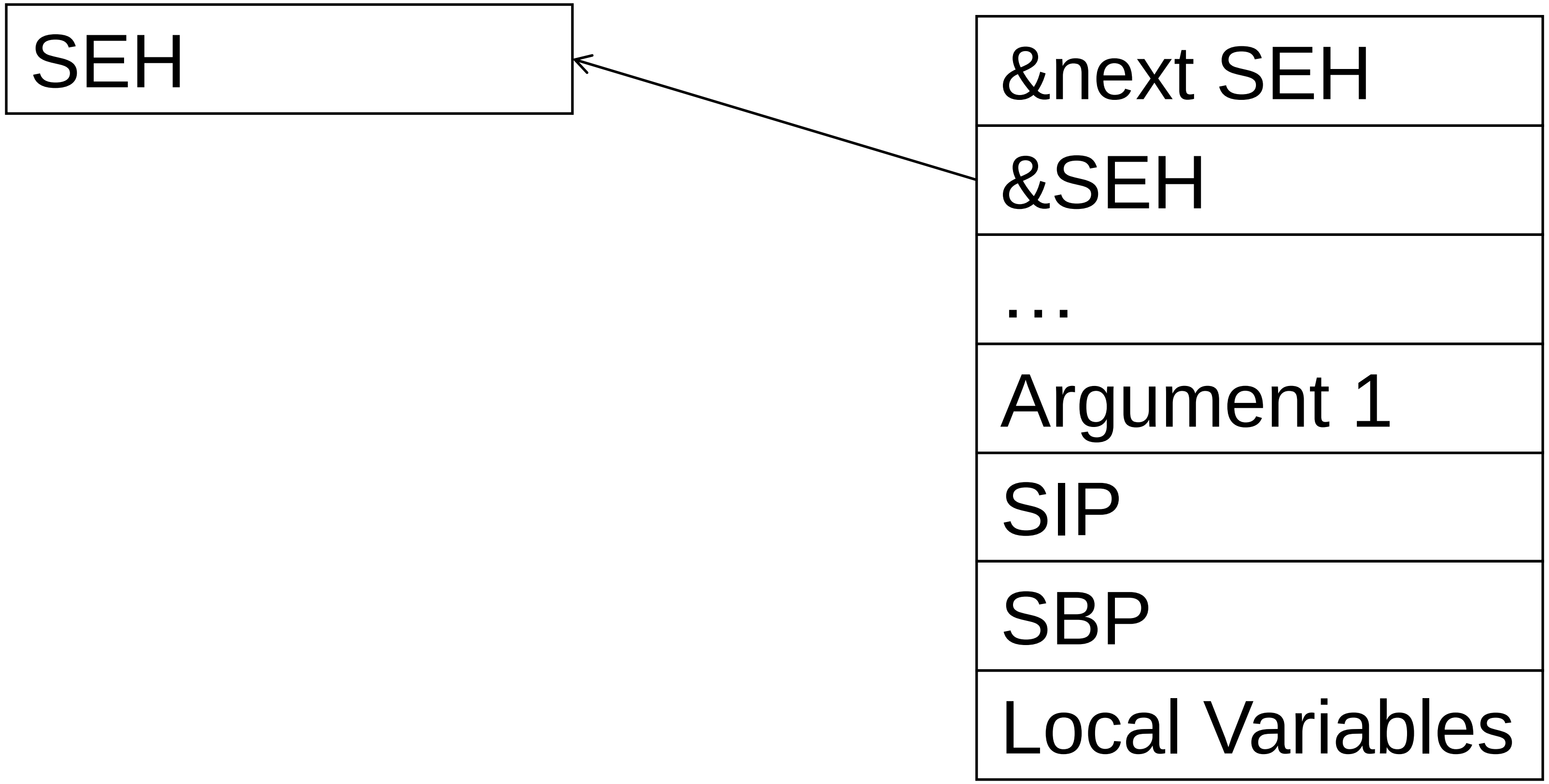
SEH Overwrite

- Structured Exception Handler
- To handle exceptions
- **Located on the stack**

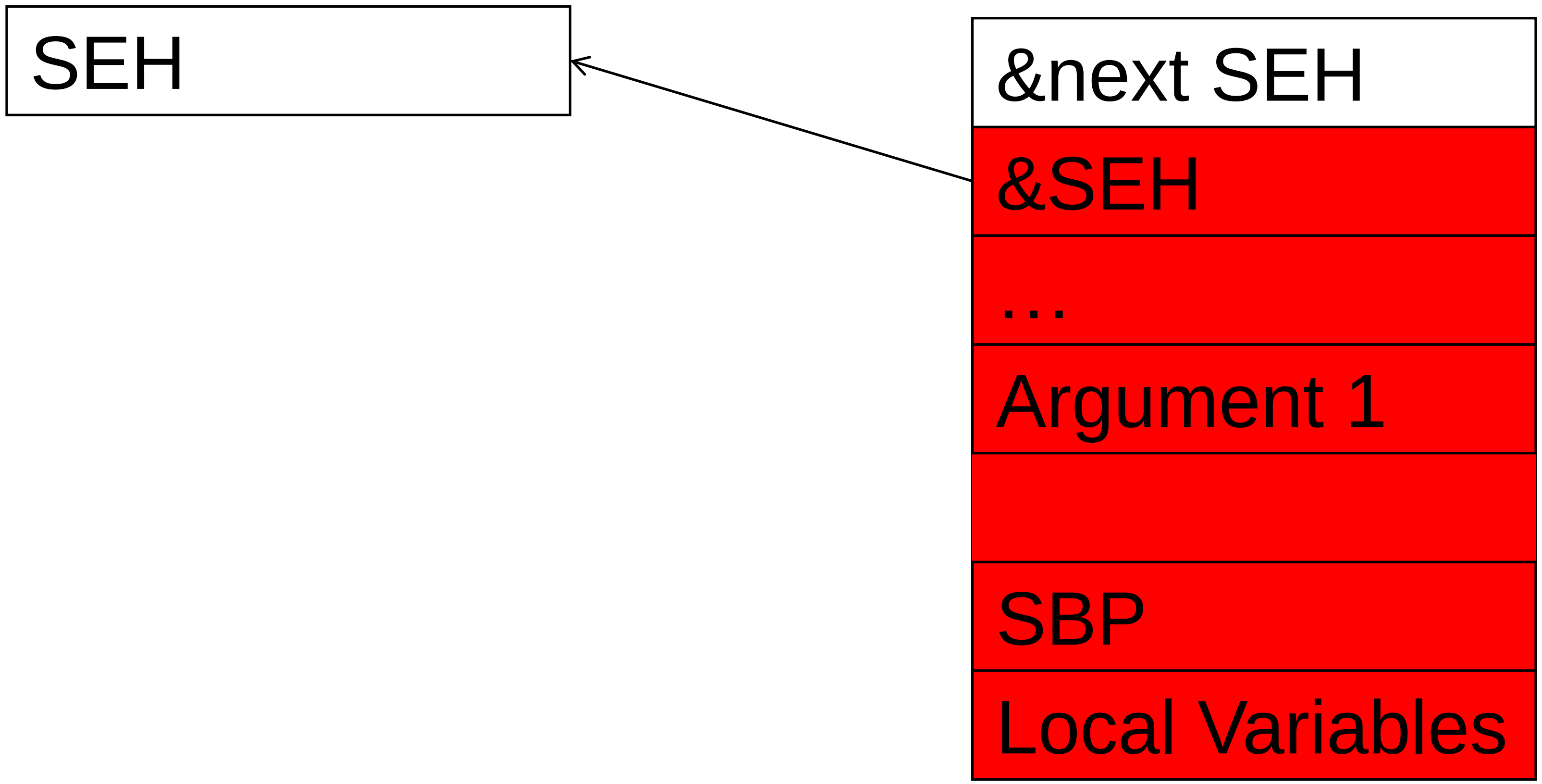
Favorite target for Windows exploits for years

https://blogs.technet.microsoft.com/srd/2009/02/02/preventing-the-exploitation-of-structured-exception-handler-seh-overwrites-with-sehop/

Windows: SEH



Windows: SEH



Windows: SEH

Mitigation: SafeSEH

- VS2003: /SafeSEH
- Whitelist of safe exception handlers

Mitigation: Dynamic SafeSEH

- End of SEH List has a validation frame
- The complete SEH list has to be valid (*next)

Mitigation: SEHOP

- Default active in Windows Server 2008, Vista SP2 (?)
- SEH Overwrite Protection

Windows Exploits

Ret2libc

Windows: Call convention

Call convention:

- “Stdcall” call convention
  - Caller pushes arguments
  - Callee pops arguments (unlike linux!)

Can call Windows library functions

- E.g: VirtualProtect()
- Changes the permission of a memory region
- Can make it executable again (removing DEP)

Windows: ret2libc

VirtualProtect: Set memory protection bits

```
BOOL WINAPI VirtualProtect(
  _In_  LPVOID lpAddress,
  _In_  SIZE_T dwSize,
  _In_  DWORD  flNewProtect,
  _Out_ PDWORD lpflOldProtect
);
```

Windows: ret2libc

Ret2libc chaining:

```
BOOL WINAPI VirtualProtect(
  _In_  LPVOID lpAddress,
  _In_  SIZE_T dwSize,
  _In_  DWORD  flNewProtect,
  _Out_ PDWORD lpflOldProtect
);
```

<shellcode>
lpflOldProtect
flNewProtect
dwSize
lpAddress
&jmp esp
SIP (&<VirtualProtect>)
SFP
isAdmin
firstname

Windows: ret2libc

Ret2libc chaining:

```
BOOL WINAPI VirtualProtect(
  _In_  LPVOID lpAddress,
  _In_  SIZE_T dwSize,
  _In_  DWORD  flNewProtect,
  _Out_ PDWORD lpflOldProtect
);
```

<shellcode>
&writeableAddr
RWX
len(shellcode)
&shellcode
&jmp esp
SIP (&<VirtualProtect>)
SFP
isAdmin
firstname

Windows: ret2libc

Ret2libc chaining:

```
BOOL WINAPI VirtualProtect(
  _In_  LPVOID lpAddress,
  _In_  SIZE_T dwSize,
  _In_  DWORD  flNewProtect,
  _Out_ PDWORD lpflOldProtect
);
```

<shellcode>
&writeableAddr
RWX
len(shellcode)
&shellcode
&jmp esp
SIP (&<VirtualProtect>)
SFP
isAdmin
firstname

Windows: ret2libc

Conclusion:

Possible to chain library calls

Like ROP, just for function calls

Can defeat DEP (or be used for other things)

Windows Exploit Mitigation

ASLR

Windows: ASLR

ASLR in Windows

- Introduced in Windows Vista

Windows 7

- Randomized: Heap and Stack
- Not randomized: VirtualAlloc, MapViewOfFile
- A little randomized: PEBs, TEPBs

Windows 8

- **Opt-in!** (/dynamicbase)
- More things are randomized
  - A little bit more randomized: PEBs, TEPBs
- High Entropy ASLR for 64 bit processes

Windows: ASLR

Windows ASLR problems

- Not all binaries are compiled with relocation
- Windows Vista: Relocation on Boot
  - Brute force able
- "… if the same library is loaded in multiple processes, it will be at the same base address; so any library loaded in the renderer will be loaded at a known address in the browser process."

- Not all libraries are compiled with relocation!
  - Adobe Flash…
  - Adobe PDF…
  - Java…
  - Some Antivirus inject(ed) DLLs
    - On every process
    - On static addresses…

Windows: ASLR

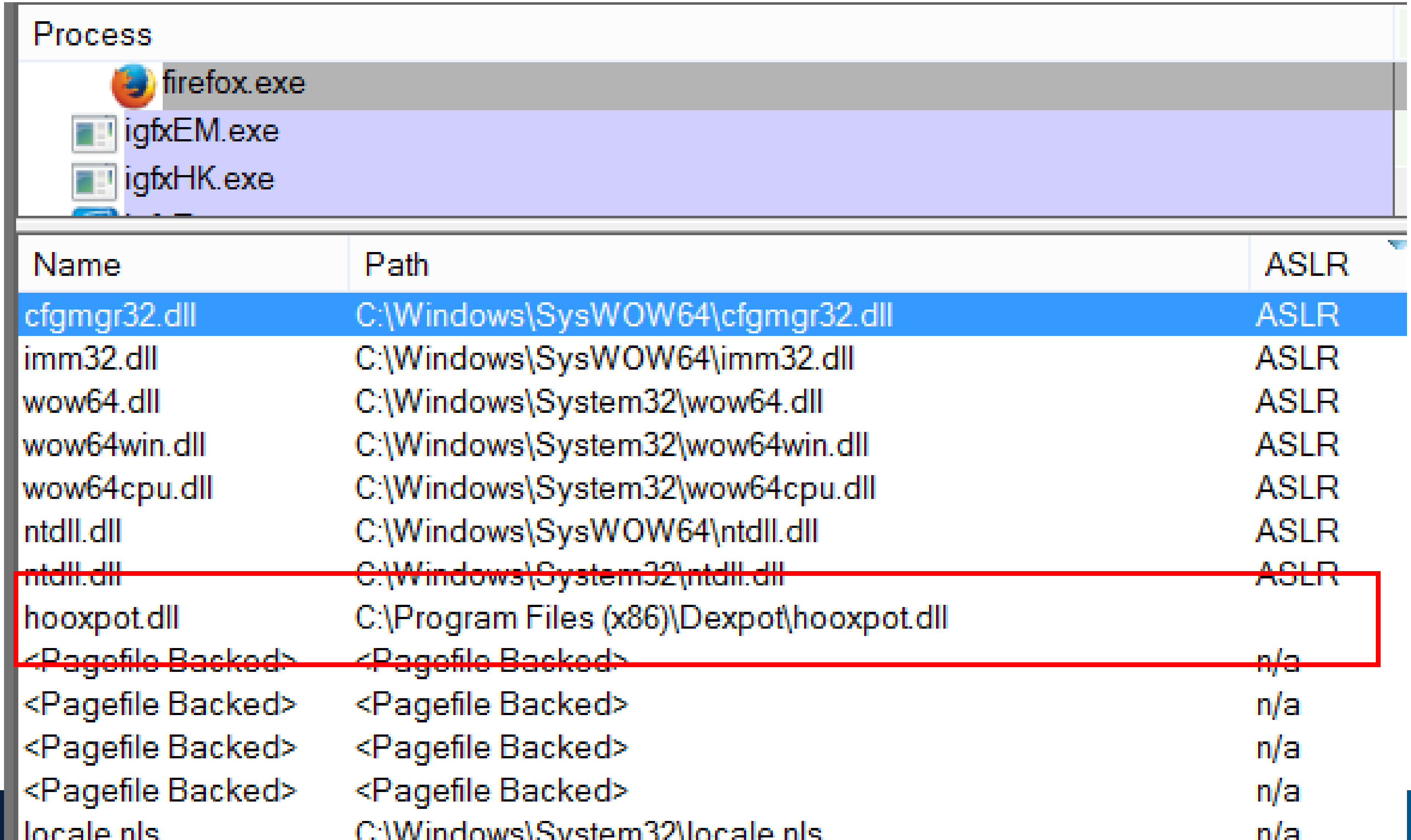
Pidgin DLL ASLR status:

pidgin.exe

Name	Path	ASLR
wow64.dll	C:\Windows\System32\wow64.dll	ASLR
wow64win.dll	C:\Windows\System32\wow64win.dll	ASLR
wow64cpu.dll	C:\Windows\System32\wow64cpu.dll	ASLR
ntdll.dll	C:\Windows\SysWOW64\ntdll.dll	ASLR
ntdll.dll	C:\Windows\System32\ntdll.dll	ASLR
libpng14-14.dll	C:\Program Files (x86)\Pidgin\Gtk\bin\libpng14-14.dll	
libmeanwhile-1.dll	C:\Program Files (x86)\Pidgin\libmeanwhile-1.dll	
hooxpot.dll	C:\Program Files (x86)\Dexpot\hooxpot.dll	
exchndl.dll	C:\Program Files (x86)\Pidgin\exchndl.dll	
libgtk-win32-2.0-0.dll	C:\Program Files (x86)\Pidgin\Gtk\bin\libgtk-win32-2.0-0.dll	
libatk-1.0-0.dll	C:\Program Files (x86)\Pidgin\Gtk\bin\libatk-1.0-0.dll	
libsilcclient-1-1-3.dll	C:\Program Files (x86)\Pidgin\libsilcclient-1-1-3.dll	
libwimp.dll	C:\Program Files (x86)\Pidgin\Gtk\lib\gtk-2.0\2.10.0\engines\li...	
zlib1.dll	C:\Program Files (x86)\Pidgin\Gtk\bin\zlib1.dll	
libgobject-2.0-0.dll	C:\Program Files (x86)\Pidgin\Gtk\bin\libgobject-2.0-0.dll	
libsilc-1-1-2.dll	C:\Program Files (x86)\Pidgin\libsilc-1-1-2.dll	
libfontconfig-1.dll	C:\Program Files (x86)\Pidgin\Gtk\bin\libfontconfig-1.dll	

Windows: ASLR

Dexpot DLL injection



Process
firefox.exe
igfxEM.exe
igfxHK.exe

Name	Path	ASLR
cfgmgr32.dll	C:\Windows\SysWOW64\cfgmgr32.dll	ASLR
imm32.dll	C:\Windows\SysWOW64\imm32.dll	ASLR
wow64.dll	C:\Windows\System32\wow64.dll	ASLR
wow64win.dll	C:\Windows\System32\wow64win.dll	ASLR
wow64cpu.dll	C:\Windows\System32\wow64cpu.dll	ASLR
ntdll.dll	C:\Windows\SysWOW64\ntdll.dll	ASLR
ntdll.dll	C:\Windows\System32\ntdll.dll	ASLR
hooxpot.dll	C:\Program Files (x86)\Dexpot\hooxpot.dll	
<Pagefile Backed>	<Pagefile Backed>	n/a
<Pagefile Backed>	<Pagefile Backed>	n/a
<Pagefile Backed>	<Pagefile Backed>	n/a
<Pagefile Backed>	<Pagefile Backed>	n/a
locale.nls	C:\Windows\System32\locale.nls	n/a

Windows: ASLR

ASLR entropy improvements

Entropy (in bits) by region	Windows 7 32-bit	Windows 7 64-bit	Windows 8 32-bit	Windows 8 64-bit	Windows 8 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

Windows Exploit Mitigation

HEAP

Windows: Heap

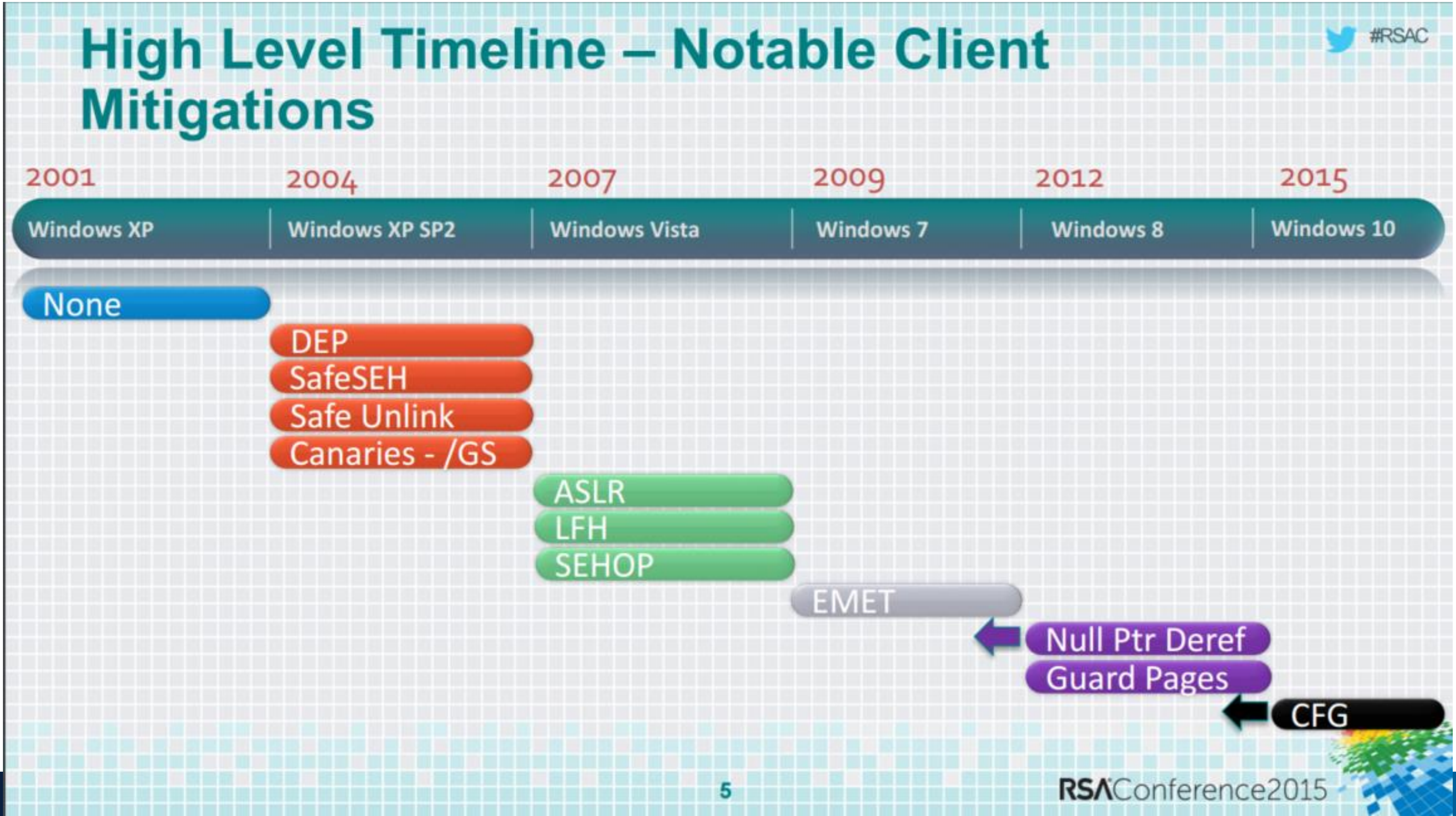
Heap Protections:

- 2004: Safe unlinking
- 2006: Vista heap hardening
- Win8:
  - Additional Heap metadata structure improvements
  - Guard pages
  - Allocation order randomization
    - Makes HEAP massaging more difficult

Windows Exploit Mitigation

History

Windows History



Windows History

http://www.welivesecurity.com/wp-content/uploads/2017/01/Windows-Exploitation-2016-A4.pdf

Mitigation (*SetProcessMitigationPolicy*)	Windows 8.1	Windows 10
DEP (*ProcessDEPPolicy*)	X	X
ASLR (*ProcessASLRPolicy*)	X	X
Dynamic code prohibited (*ProcessDynamicCodePolicy*)	X	X
Strict handle checks (*ProcessStrictHandleChecksPolicy*)	X	X
Win32k system calls disabled (*ProcessSystemCallDisablePolicy*)	X	X
Extension points disabled (*ProcessExtensionPointDisablePolicy*)	X	X
Control Flow Guard enabled (*ProcessControlFlowGuardPolicy*)	X	X
Signatures restricted (*ProcessSignaturePolicy*)		X
Non-system fonts disabled (*ProcessFontDisablePolicy*)		X
Loading of remote and low IL images disabled (*ProcessImageLoadPolicy*)		X

Table 6. List of mitigations that are available for applications to use to improve their own security.

Windows History

Bill Gates' "Trustworthy Compting Memo" from 2012

Aka "Stop the fuck you are doing right now, get 6 months of education on how to do things securely"

Security: *The data our software and services store on behalf of our customers should be protected from harm and used or modified only in appropriate ways. Security models should be easy for developers to understand and build into their applications.*
https://news.microsoft.com/2012/01/11/memo-from-bill-gates/

The move was reportedly prompted by the fact that they *"...had been under fire from some of its larger customers–government agencies, financial companies and others–about the security problems in Windows, issues that were being brought front and center by a series of self-replicating worms and embarrassing attacks."* such as Code Red, Nimda and Klez.

Windows History

Virus:

- Self replicating
- File based
- Requires some user interaction

Worm:

- Self replicating
- Network based
- Requires no user interaction

Trojan:

- Fake some good functionality
- But perform evil actions

Backdoor:

- Bypass authentication/authorization

Malware!

Windows History

SDL: Security Development Lifecycle

What is the Security Development Lifecycle ?

The Security Development Lifecycle (SDL) is a software development process that helps developers build more secure software and address security compliance requirements while reducing development cost

Training | Requirements | Design | Implementation | Verification | Release | Response

Click to select a phase

Training Phase

SDL Practice #1: Core Security Training

This practice is a prerequisite for implementing the SDL. Foundational concepts for building better software include secure design, threat modeling, secure coding, security testing, and best practices surrounding privacy.

Windows History

SDL: Security Development Lifecycle

1. TRAINING → 2. REQUIREMENTS → 3. DESIGN → 4. IMPLEMENTATION → 5. VERIFICATION → 6. RELEASE → 7. RESPONSE

Design Phase

SDL Practice #5: Establish Design Requirements

Considering security and privacy concerns early helps minimize the risk of schedule disruptions and reduce a project's expense.

SDL Practice #6: Attack Surface Analysis/Reduction

Reducing the opportunities for attackers to exploit a potential weak spot or vulnerability requires thoroughly analyzing overall attack surface and includes disabling or restricting access to system services, applying the principle of least privilege, and employing layered defenses wherever possible.

SDL Practice #7: Use Threat Modeling

Applying a structured approach to threat scenarios during design helps a team more effectively and less expensively identify security vulnerabilities, determine risks from those threats, and establish appropriate mitigations.

Windows History

SDL: Security Development Lifecycle

1. TRAINING › 2. REQUIREMENTS › 3. DESIGN › 4. IMPLEMENTATION › 5. VERIFICATION › 6. RELEASE › 7. RESPONSE

Verification Phase

SDL Practice #11: Perform Dynamic Analysis

Performing run-time verification checks software functionality using tools that monitor application behavior for memory corruption, user privilege issues, and other critical security problems.

SDL Practice #12: Fuzz Testing

Inducing program failure by deliberately introducing malformed or random data to an application helps reveal potential security issues prior to release while requiring modest resource investment.

SDL Practice #13: Attack Surface Review

Reviewing attack surface measurement upon code completion helps ensure that any design or implementation changes to an application or system have been taken into account, and that any new attack vectors created as a result of the changes have been reviewed and mitigated including threat models.

Windows History

Windows XP SP2

- First big step in anti-exploiting
- Compiled with /GS /SAFESEH
- DEP

Windows Vista

- ASLR

Windows History

Windows 8

/GS:

- Better heuristics
- VS now performs bounds checks on array

ASLR:

- Force ASLR on all DLLs of a process (Force ASLR option)

Windows History

Windows 10

- Control Flow Guard (CFG)
  - Anti ROP
  - Needs help from compiler (Visual studio)
  - Pretty damn awesome
  - IE11 @Win8 Update 3
  - Edge
- EDGE: MemGC
  - Use-After-Free exploit mitigation
- Improved Kernel ASLR
- EPM (Enhanced Protected Mode, Sandbox for IE)

Windows History

Control Flow Integrity (CFI)

`/guard:cf`

- Control Flow Guard

> - First, the compiler identifies all indirect branches in a program
> - Next, it determines which branches must be protected. For instance, indirect branches that have a statically identifiable target don't need CFI checks.
> - Finally, the compiler inserts lightweight checks at potentially vulnerable branches to ensure the branch target is a valid destination.

https://blog.trailofbits.com/2016/12/27/lets-talk-about-cfi-microsoft-edition/

Windows History

Example: Windows 10 IE11 + EPM + EMET exploit;

- Find UAF
- Heap massage
- Overwrite arraybuffer length for write-what-where
- Re-enable God-Mode (Compiler fail…)
- Without ROP (because of CFI)
- Execute ActiveX
- -> Still in EPM Sandbox
  - Create local web server via ActiveX
  - Netbios DNS spoof/bruteforce to fake hostname so website is in trusted zone
  - Perform above exploit again in 32bit
  - Full RCE

Windows 10

Windows 10 Protections

Process / Mitigation	Services	Smss	Csrss	Winlogon	Lsass	Explorer
DEP	X	X	X	X	X	X
HEASLR, force relocate	X	X	X	X	X	ASLR
Dynamic code prohibited	X	X	X			
Strict handle checks	X	X	X	X	X	
Win32k system calls disabled						
Extension points disabled	X					
Control Flow Guard enabled	X	X	X	X	X	X
Signatures restricted	X (MS only)	X (MS only)	X (MS only)			
Non-system fonts disabled						
Loading of remote and low IL images disabled						

Table 7. Mitigations that are applied by default for important processes (Windows 10).

http://www.welivesecurity.com/wp-content/uploads/2017/01/Windows-Exploitation-2016-A4.pdf

Windows 10 Protections

Hypervisor based security

- DeviceGuard, Credential Guard, Hypervisor Code Integrity (HVCI)
- Use separate VM's for sensitive tasks

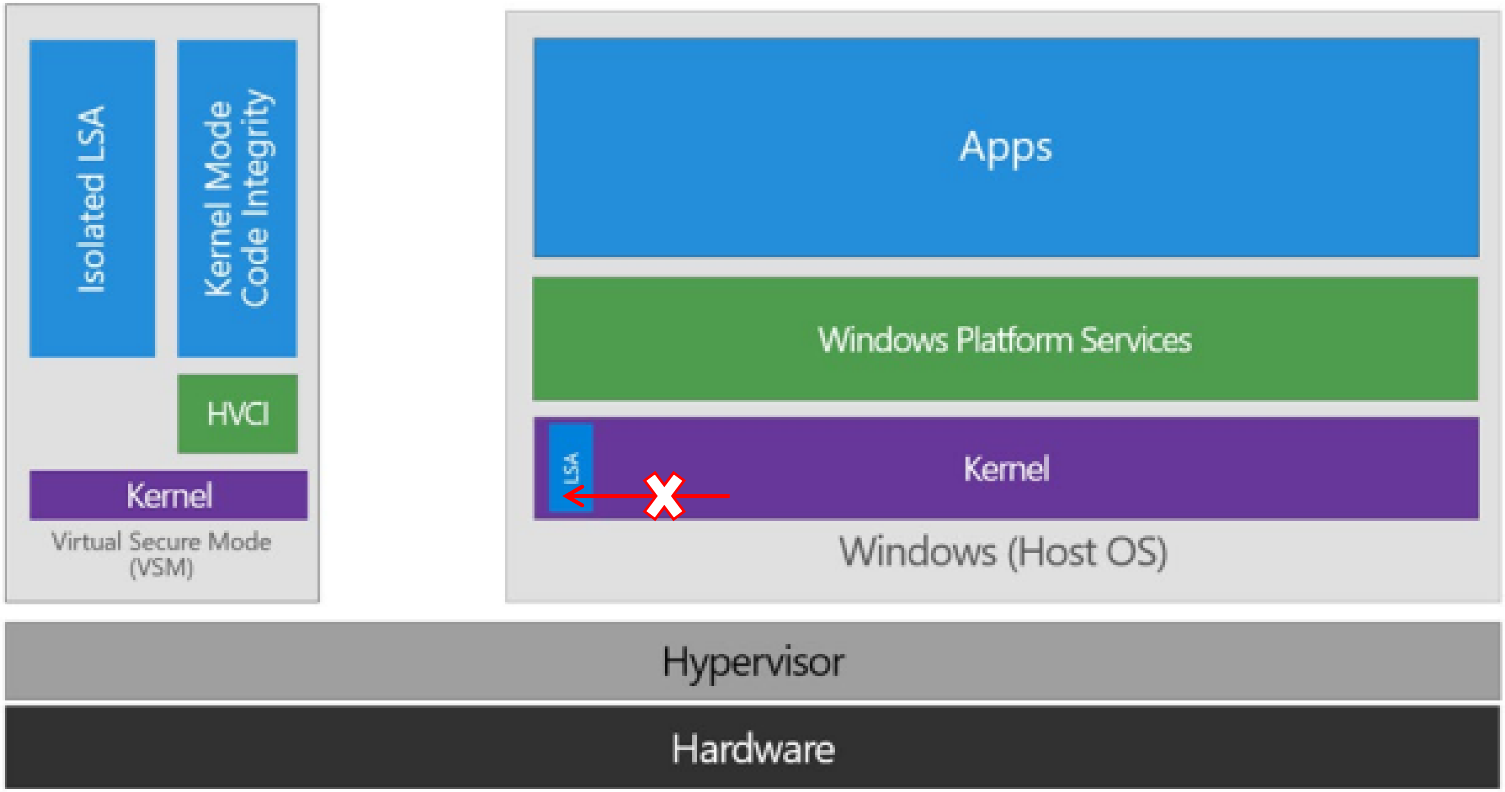


Figure 6. Hyper-V architecture with VSM as it is described by the Microsoft security guys.

Windows 10 Protections

Hypervisor based security

- Windows Defender Application Guard

However, when an employee browses to a site that is not recognized or trusted by the network administrator, Application Guard steps in to isolate the potential threat. As shown in the mode outlined in red above, Application Guard creates a new instance of Windows at the hardware layer, with an entirely separate copy of the kernel and the minimum Windows Platform Services required to run Microsoft Edge. The underlying hardware enforces that this separate copy of Windows has no access to the user’s normal operating environment.

https://blogs.windows.com/msedgedev/2016/09/27/application-guard-microsoft-edge/#SI3kumwvwgYoTPiL.97

Windows 10 Protections

Hypervisor bas

- Windows D

However, whe … rusted by

the network … otential

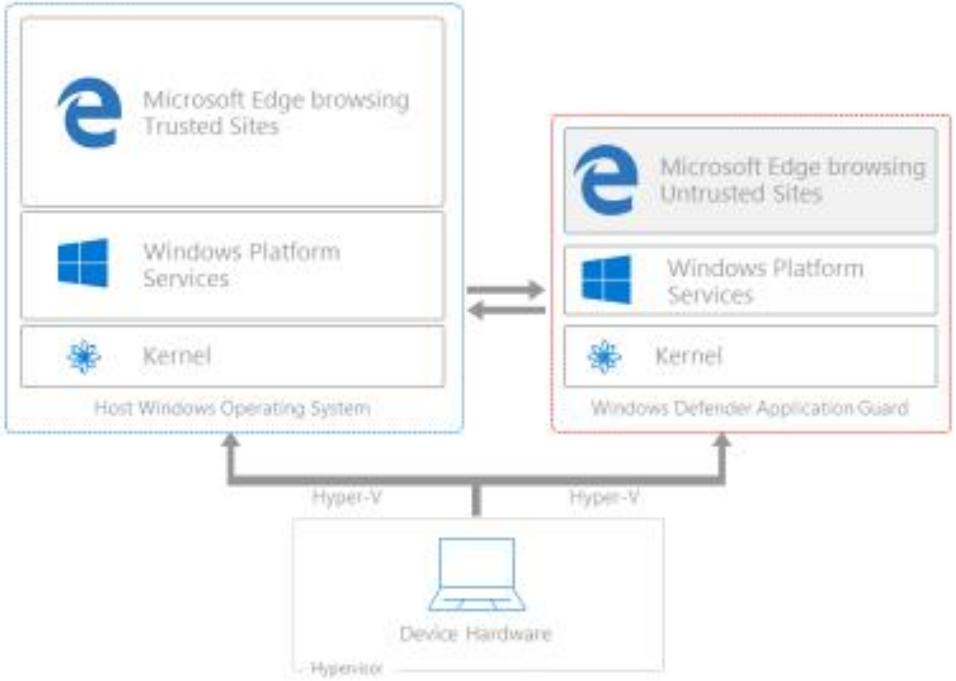
threat. As s … reates a

new instance … e copy of

the kernel a … icrosoft

Edge. The un … ows has no

access to th



https://blogs.windows.com/msedgedev/2016/09/27/application-guard-microsoft-edge/#SI3kumwvwgYoTPiL.97

Windows Exploit Mitigations Conclusion

Windows: Conclusion

Its not 2001 anymore…

- We don’t need to reboot Windows to change IP address anymore
- We don’t have IE6 anymore (IE7 was a partial rewrite after the Bill Gates Memo)

- Current Windows versions have anti exploiting techniques, which:
  - Are superiour to Linux one’s
  - Enabled by default
  - But still not complete

Windows: Conclusion

Main problems:

- Backwards compatibility / technical depth
  - Parts of UI in Kernelspace
  - Pass the hash / Kerberos…
- 3rd party programs
  - Adobe (Flash, PDF Reader)
  - Oracle (Java)
  - Cisco (Webex)
  - HP (Data “Protector”)
- Monocolture (everybody has the same Windows version)
- Unsavy users
- Worse: Unsavy administrators

References

References

References:

https://media.blackhat.com/bh-us-12/Briefings/M_Miller/BH_US_12_Miller_Exploit_Mitigation_Slides.pdf
https://www.rsaconference.com/writable/presentations/file_upload/exp-r01_patching-exploits-with-duct-tape-bypassing-mitigations-and-backward-steps.pdf

References

http://www.welivesecurity.com/wp-content/uploads/2017/01/Windows-Exploitation-2016-A4.pdf
http://www.welivesecurity.com/wp-content/uploads/2016/01/Windows_Exploitation_in_2015.pdf
http://www.welivesecurity.com/wp-content/uploads/2015/01/Windows-Exploitation-in-2014.pdf
http://www.welivesecurity.com/2014/02/11/windows-exploitation-in-2013/
http://slides.com/revskills/fzbrowsers#/
https://github.com/Microsoft/MSRC-Security-Research/blob/master/presentations/2009_03_CanSecWest/CSW09_Burrell_Miller_Evolution_Of_Microsofts_Mitigations.pdf