Windows 10 Kernel Mitigations and Exploitation

Stephen Sims @Steph3nSims
Jaime Geiger @jgeigerm

Speakers

Jaime Geiger
GRIMM
Certified SANS Instructor
@jgeigerm

- Northern Virginia
- Co-Author - SEC760

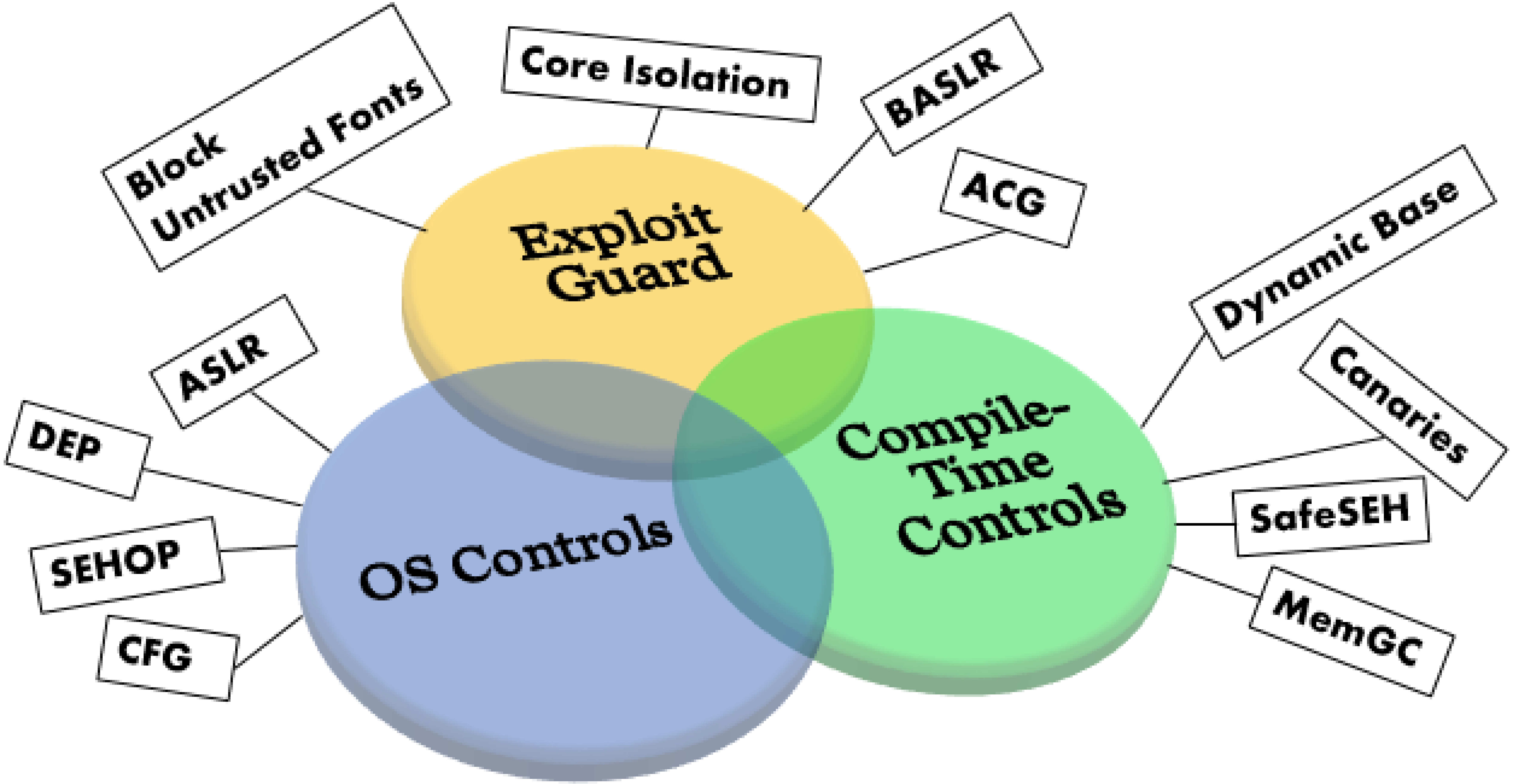
Stephen Sims
SANS Pen Test Curriculum Lead
SANS Fellow
@Steph3nSims

- Berkeley, California
- Author - SEC599, SEC501, SEC660, SEC760

Exploit Mitigations

What are they?

Exploit Mitigation Controls



Attack Surface – Why do we need the mitigations?

Win7

Windows 10 without Mitigations

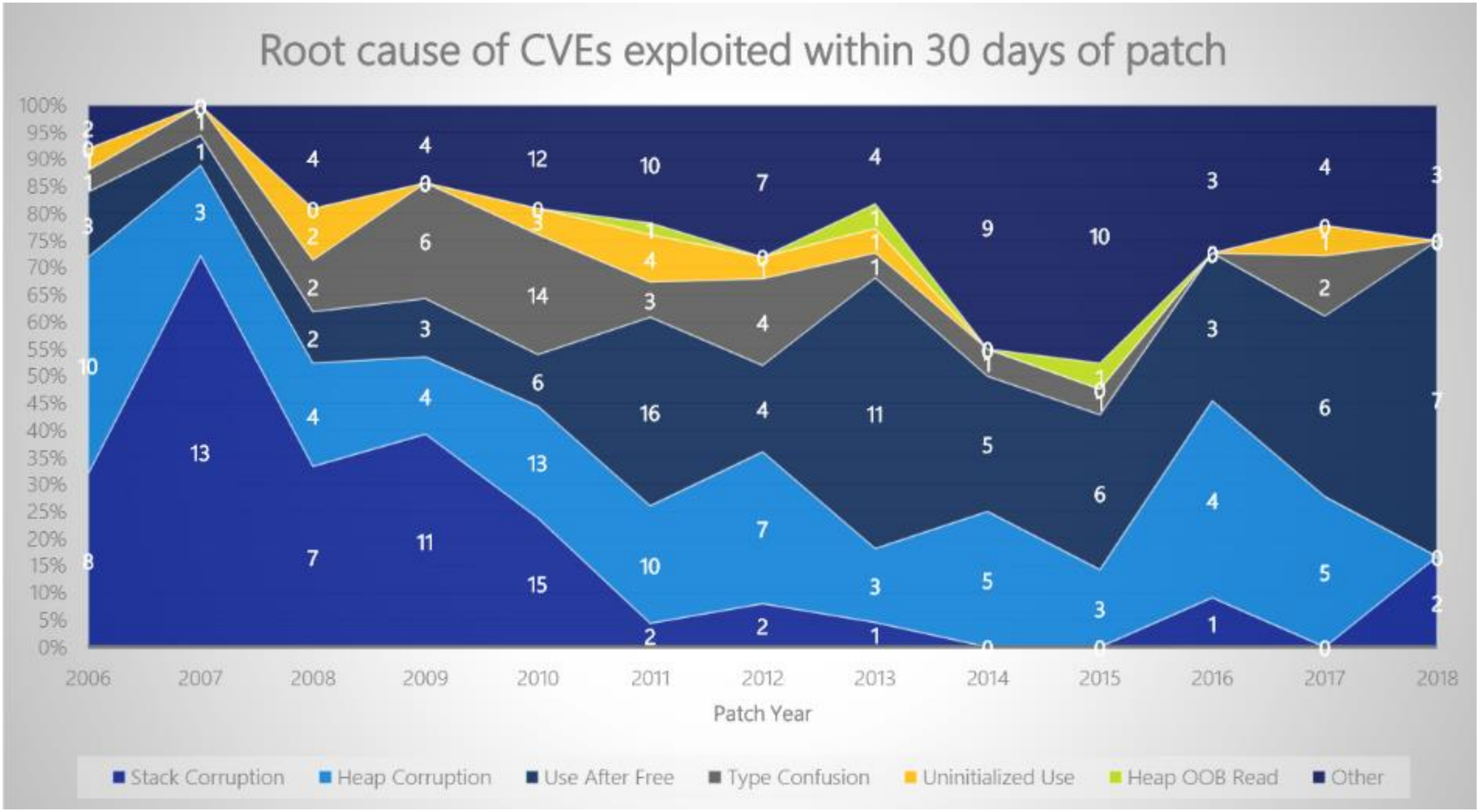
Windows 10 with Mitigations

What Are These Mitigations Targeting?

- Common exploitation techniques include:
  - Stack & Heap Overflows
  - Integer Overflows
  - Null Pointer Dereferencing
  - Use After Free & Type Confusion
  - Race Conditions – Double Fetch
  - Logic Bugs

- They are attempting to block a successful attack, or at least make the life of an attacker more difficult

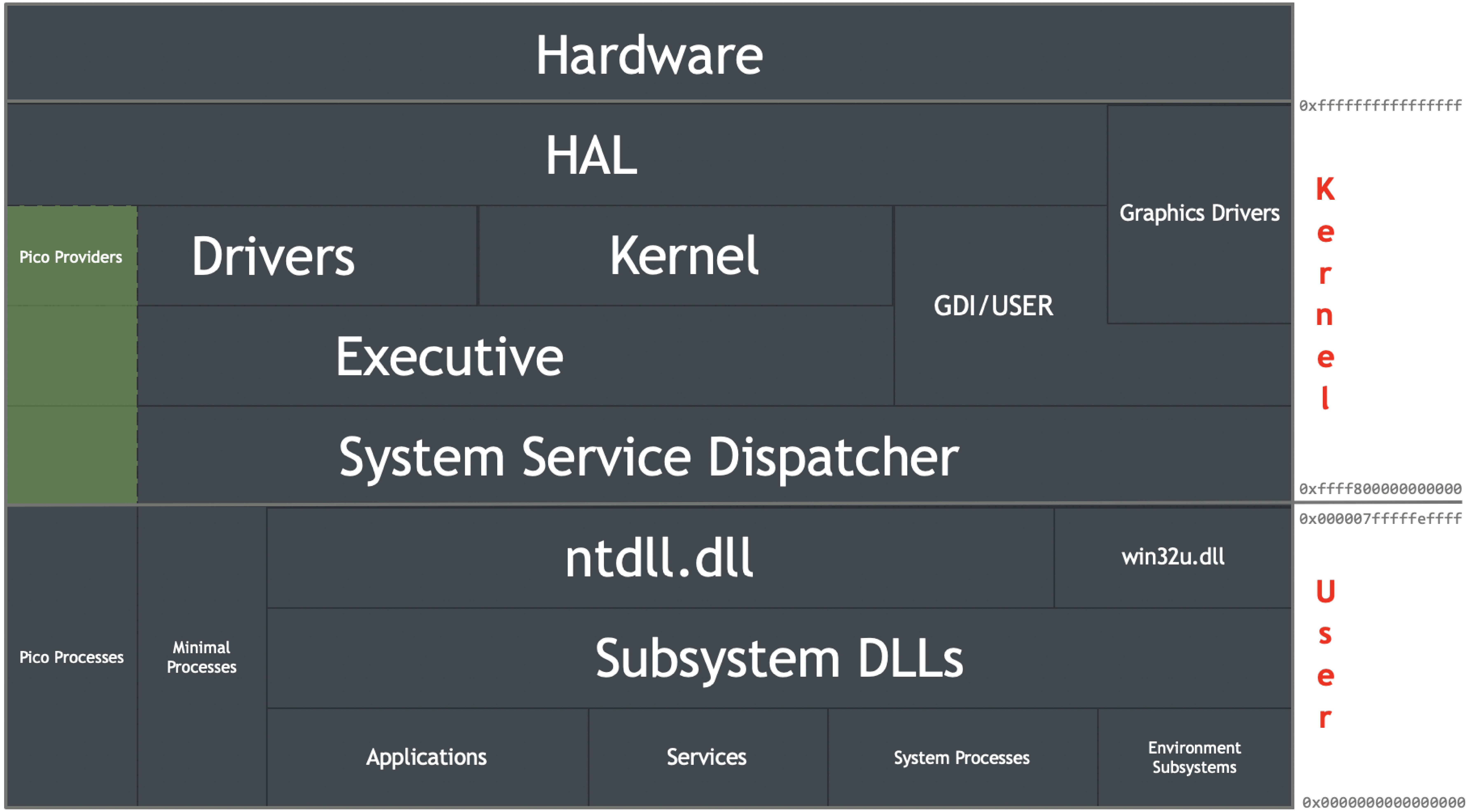
RCE Vulnerability Trend

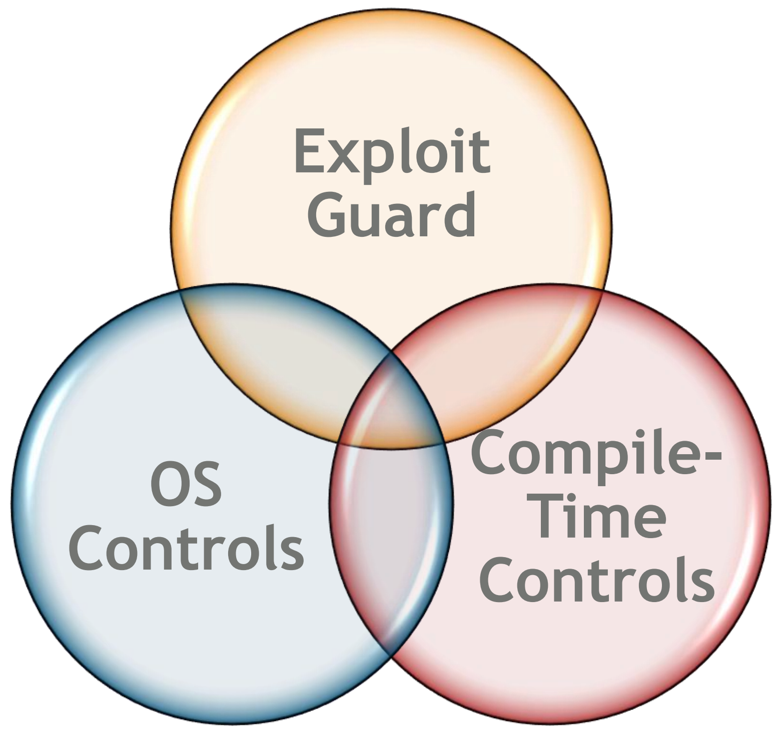


Miller, M. (2019, August 12). Trends and challenges in the vulnerability mitigation landscape. Retrieved April 14, 2020, from https://github.com/microsoft/MSRC-Security-Research/blob/master/presentations/2019_08_WOOT/WOOT19 - Trends and challenges in vulnerability mitigation.pdf

Kernel Mitigations

Why attack the Kernel?





Kernel Exploit Mitigations Overview

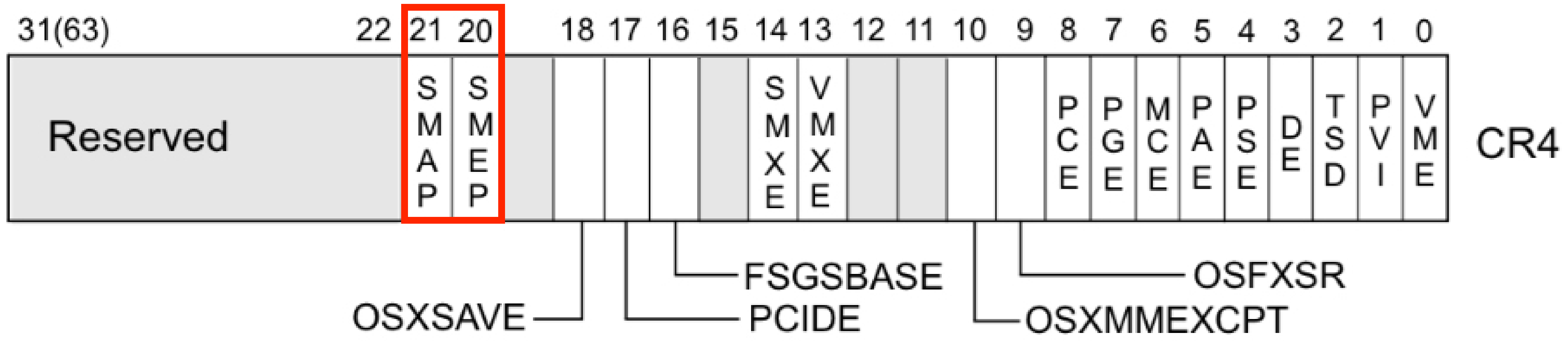
- Kernel Mode Code Signing (KMCS)
- Supervisor Mode Execution/Access Prevention (SMAP/SMEP)
- Kernel Address Space Layout Randomization (KASLR)
- Control Flow Guard (CFG)
- Virtual Based Security (VBS) and Device/Credential Guard
- PatchGuard (KPP)
- Other and Future Mitigations

Kernel Mode Code Signing

- Can be defeated by exploiting signed driver
- Windows Hardware Quality Labs (WHQL)
- Extended Validation (EV)
  - Extra tests, no third-party signing
  - Mode available in Server 2019 to only allow EV-signed drivers
- Once it’s signed, it will load, even if it is expired
- Leaked certificates can help you sign your own code
  - Antivirus may look for drivers signed by these certificates

Supervisor Mode Execution/Access Prevention

- Prevent execution or access of data residing in user mode from kernel mode
- SMEP is fully implemented as of Windows 8
- SMAP is only implemented as of Windows 10 1903
  - Very limited form, would break many legacy drivers
  - EFLAGS AC bit allows switching it on and off when user mode addresses need to be accessed



Kernel ASLR and Address Leak Protection

- Kernel ASLR has been vastly improved over time
  - HEASLR + ForceASLR make a powerful combination
  - 4 bits of entropy (Win Vista 32-bit) -> 22 bits of entropy (Win10 64-bit)
- Windows 10 vastly reduces the number of information leaks that could disclose the base of the kernel or other modules
  - Randomize HAL heap
  - Remove kernel pointer references from TEB (Desktop Heap) and PEB (GdiSharedHandleTable)
- GDI objects have been moved into session pool to reduce likelihood of read/write primitive abuse
- Randomized page tables
- Unable to query driver bases via Psapi and NtQuerySystemInformation from low integrity (Win 8)

Control Flow Guard (CFG)

- Forward control flow protection
- Guards indirect calls via validation of destination
  - Is the function that is about to be called a valid call target?
- Only enabled on Pro and Enterprise versions of Windows 10 when Core Isolation is also enabled
  - guard_dispatch_icall validates call targets via bitmap check

Call what is at cs:off_1401F1C68

```
loc_1403EDB8D:
mov     edx, 0Ch
mov     [rsp+38h+var_18], ecx
lea     r9, [rsp+38h+arg_0]
lea     ecx, [rdx-0Bh]
lea     r8, [rsp+38h+var_18]
call    cs:off_1401F1C68
test    eax, eax
js      short loc_1403EDBBA
```

VS

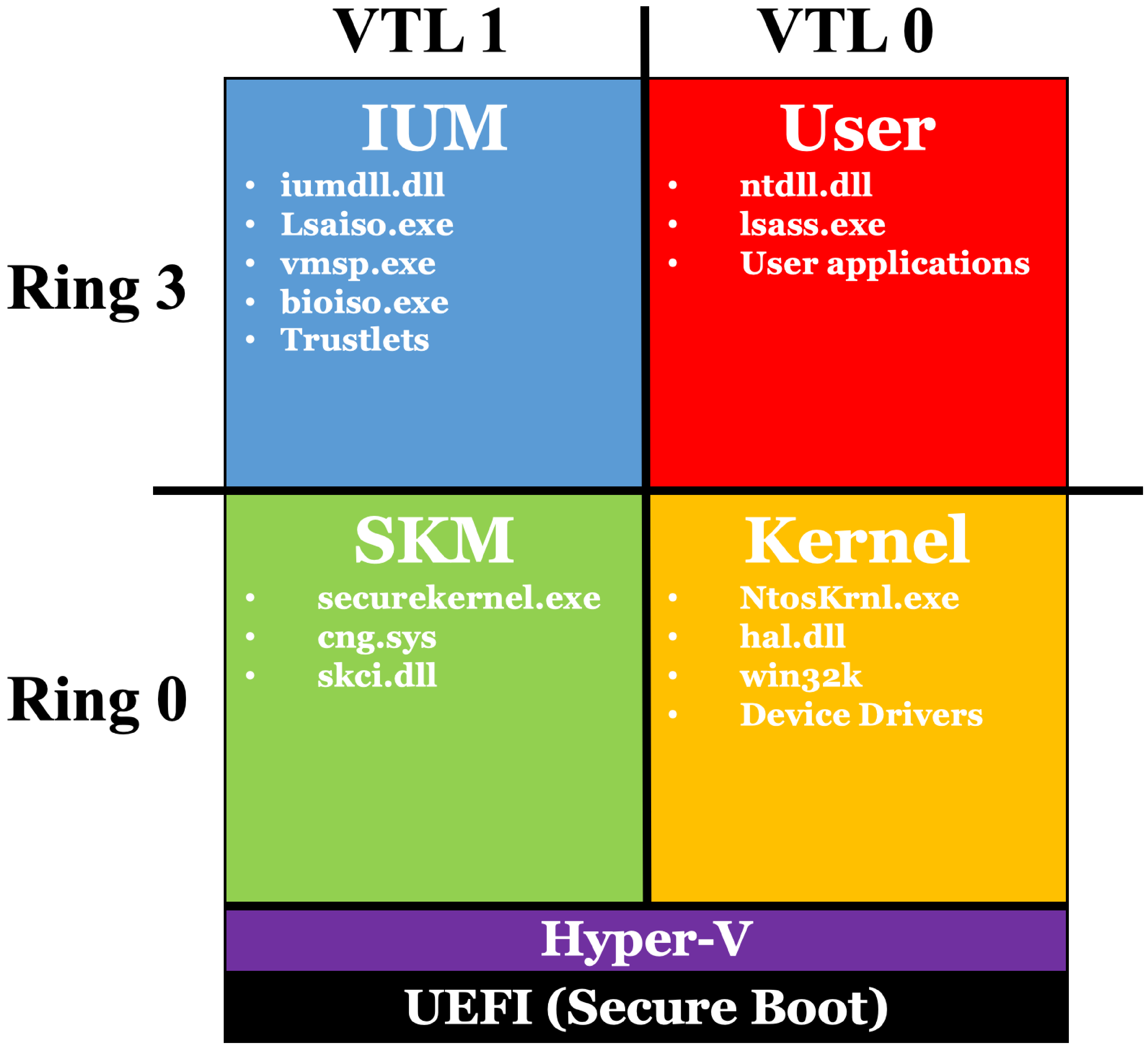
```
mov     rax, cs:off_140428258
lea     r9, [r11+8]
mov     edx, 18h
mov     [rsp+58h+var_28], ecx
lea     r8, [r11-28h]
lea     ecx, [rdx-17h]
call    _guard_dispatch_icall
test    eax, eax
js      short loc_1406F314A
```

Check if what is at cs:off_140428258 is a valid call target before calling

Virtual Based Security (VBS)

- Hyper-V backed security mechanism
  - Trust split across Virtual Machines into Virtual Trust Levels (VTLs)
  - Runs a secure OS in VTL1, normal Windows OS in VTL0
    - Only Microsoft-signed code can run in VTL1 if boot process is secure
- Secure Kernel Mode (SKM) – VTL1/Ring 0
  - Stripped-down kernel
- Isolated User Mode (IUM) – VTL1/Ring 3
  - System calls still pass through a version of NTDLL into SKM
  - Runs normal, but specially signed exes called “trustlets”

VBS Diagram



VTL 1
VTL 0
Ring 3
Ring 0
IUM
iumdll.dll
Lsaiso.exe
vmsp.exe
bioiso.exe
Trustlets
User
ntdll.dll
lsass.exe
User applications
SKM
securekernel.exe
cng.sys
skci.dll
Kernel
NtosKrnl.exe
hal.dll
win32k
Device Drivers
Hyper-V
UEFI (Secure Boot)

Device Guard and Credential Guard

- Device Guard
  - Ensure that only known-good code can run via Hypervisor Code Integrity (HVCI)
- Credential Guard
  - Lock LSA secrets away inside of Isolated User Mode
  - Includes NTLM hashes and Kerberos Tickets (TGT)
  - Lsass.exe (VTL0/Ring 3) communicates with Lsaiso.exe (VTL1/Ring 0) via secure channel (ALPC)

Hypervisor Code Integrity (HVCI)

- SKM module (SKCI.DLL)
- Checks if a page can become executable from a policy (CCI)
- Kernel Mode Code Integrity (KMCI): "Strong code guarantees"
  - Kernel pages can only become executable with proper signing
  - "Software SMEP"
- User Mode Code Integrity (UMCI): "Hard code guarantees"
  - User mode pages can only become executable with proper signing
- MSR, control register, and DMA filtering
  - Legacy drivers that request executable memory will think they have it, but the hypervisor will prevent it

PatchGuard (KPP)

- Kernel Patch Protection (KPP) a.k.a. PatchGuard protects the kernel from modifications of critical structures and registers
  - Only runs on 64-bit systems, 32-bit does not have KMCS
- Obfuscated code that hooks into many different kernel mechanisms to check for modifications periodically and randomly
- Hooks DPCs, APCs, some kernel functions, and much more
- Relies on rtdsc instruction for randomness
- Checks IDT, SSDT, HAL dispatch table, 100+ Nt- functions, MSRs
- Can be defeated with a bootkit, hard to defeat at runtime
- Does not run if a debugger is attached at boot

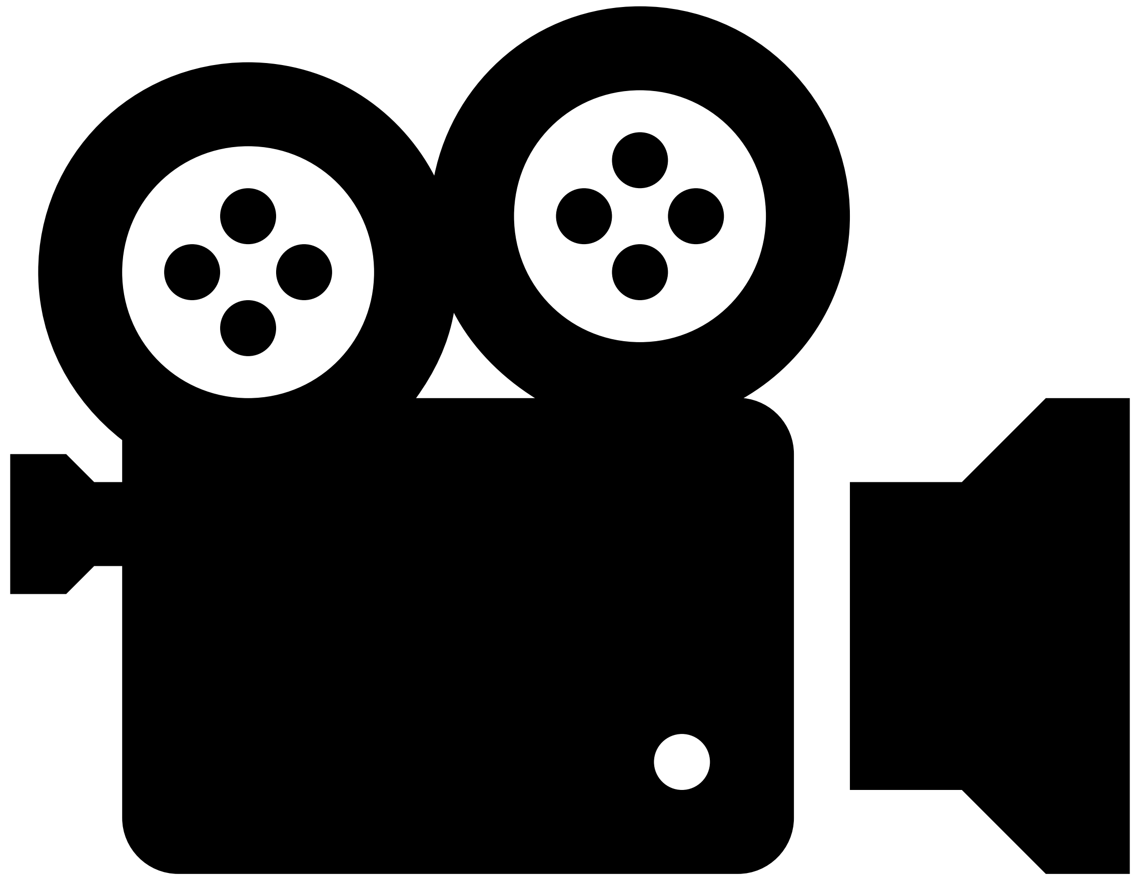
Other Mitigations

- Speculative Execution mitigations
  - Kernel Page Table Isolation (KPTI) / Kernel Virtual Address Shadow (KVAS)
  - Retpoline
- Segment heap
- Null page mapping
- Guard pages
- Range checks
- Stack cookies

Future Mitigations

- Xtreme Flow Guard (XFG) / Enhanced Control Flow Guard
  - Functions have prototype information validated before call
- Kernel Data Protection
  - Lock pages in VSM to prevent modification of important data structures (such as the code integrity bit in the kernel image)
- Control-flow Enforcement Technology (CET)
  - Hardware shadow stacks
  - Implemented in Windows 10 20H1
- System Guard Secure Launch
  - Oversee and limit impact of firmware vulnerabilities in System Management Mode (SMM)

Demo – Control Flow Guard



Thanks!

@Steph3nSims
@jgeigerm