

KERNEL MODE THREATS AND PRACTICAL DEFENSES



Joe Desimone
Gabriel Landau

ENDGAME.



@dez_

Joe Desimone

Senior Malware Researcher

Interests include offensive security research, reverse engineering, threat intelligence, and development of endpoint protections.



@gabriellandau

Gabriel Landau

Principal Software Engineer

Past work includes product & DRM evaluation, malware RE, and offensive security research. Interests include Windows internals, with a focus development of endpoint protections.

AGENDA

PART 1

Evolution of kernel
mode threats and
platform protections

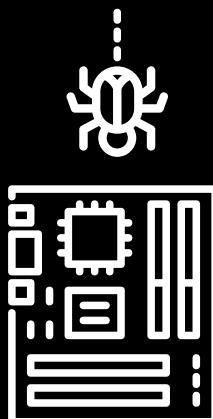
PART 2

Offensive tradecraft
to evade platform
protections

PART 3

Augmenting OS
defenses

WHY THE KERNEL?



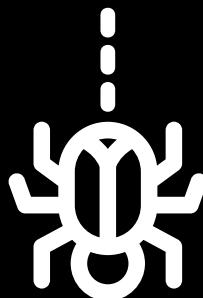
Defense Evasion

- Hide processes, files, registry keys, network activity
- Entrenchment/Persistence
- Lack of visibility into kernel

Privilege Escalation

- Same privilege as security software

FIRST GENERATION KERNEL MALWARE



Rustock (~2007)

- Infect Windows drivers on disk
- Standard rootkit functionality

TDSS/TDL-1 (~2008)

- Reg/File/Process/Network hiding
- Infect driver

ZeroAccess (~2009)

- Overwrite random driver
- Hidden encrypted NTFS volume
- x86 only

NEW OS DEFENSES



PatchGuard

- Detect kernel patching/hooks → BSOD
- First on XP SP3 x64 (~2005)



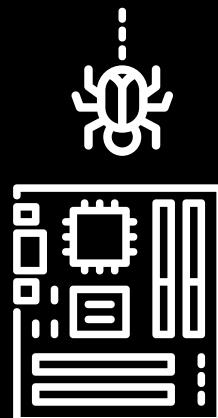
Driver Signature Enforcement (DSE)

- All drivers must be digitally signed
- First on Vista x64 (~2007)

Both defenses became more important as x64 market share grew

BOOTKITS

Replace MBR (or VBR), gain execution **before** OS.



EEye Bootroot (2005)

- First public POC

Mebroot AKA Sinowal (~2008)

- NDIS hook to evade FW

TLD4 (~2010)

- Replace kdcom.dll to gain early execution
- Modify Boot Config Data (BCD) to disable DSE

Xpaj (~2012)

- Set hooks before PG initialization

COUNTERING BOOTKITS



Secure Boot

- UEFI validates integrity of OS bootloader
- First support was Windows 8 (~2012)

Trusted Boot

- Each component of early windows boot process is verified by signature

Intel Boot Guard

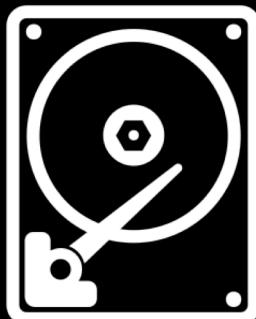
- Root of trust in hardware (CPU)

Intel BIOS Guard

- Secure BIOS Updates

BRING YOUR OWN VULN

Bring a signed, **vulnerable** driver.
Use it to bootstrap into kernel.



Uroburos / Turla

- Exploit VBox driver
- Disable DSE by modifying g_CiEnabled
- Disable PatchGuard via RtlCaptureContext hook

Derusbi

- Exploit Novell driver
- Disable DSE by modifying CiOptions
- Also used stolen certificates

Slingshot

- Exploit Sandra driver
- Hide network traffic

DUQU 2.0



- **Main payload used Oday in win32k.sys for kernel execution (CVE-2015-2360)**
- **Hooked IAT of Kaspersky driver to spoof process information**
 - Allowed user mode component to become trusted ksp process
- **Separate driver used for persistence in DMZ**
 - Used stolen Foxconn certificate
 - Redirect traffic with NDIS filtering

DOUBLEPULSAR

- Volatile kernel mode implant
- Loaded via remote ring0 exploit (ex **ETERNALBLUE**)
- Evade PatchGuard by hooking
srv!SrvTransactionNotImplemented pointer
- Injects DLLs into usermode via APCs

VIRTUALIZATION BASED SECURITY

- VT-L1 - new Secure Kernel, trustlets
- VT-L0 - normal Kernel, drivers
- Extended Page Tables (SLAT/EPT) to guard access
- IOMMU to protect from DMA access
- Hypervisor Code Integrity (HVCI)
 - All kernel code must be signed, (W^X)
- Credential Guard

Attacker



Initial Kernel Malware –
Rustock, TDSS, ZeroAccess

Defender

First Kernel Defenses – Patchguard and
Driver Signature Enforcement (DSE)

Attacker



BootKit Malware – Sinowal,
TDL4, Xpaj

Defender

Countering Bootkits – SecureBoot,
Trusted Boot, Boot Guard, etc.

Attacker



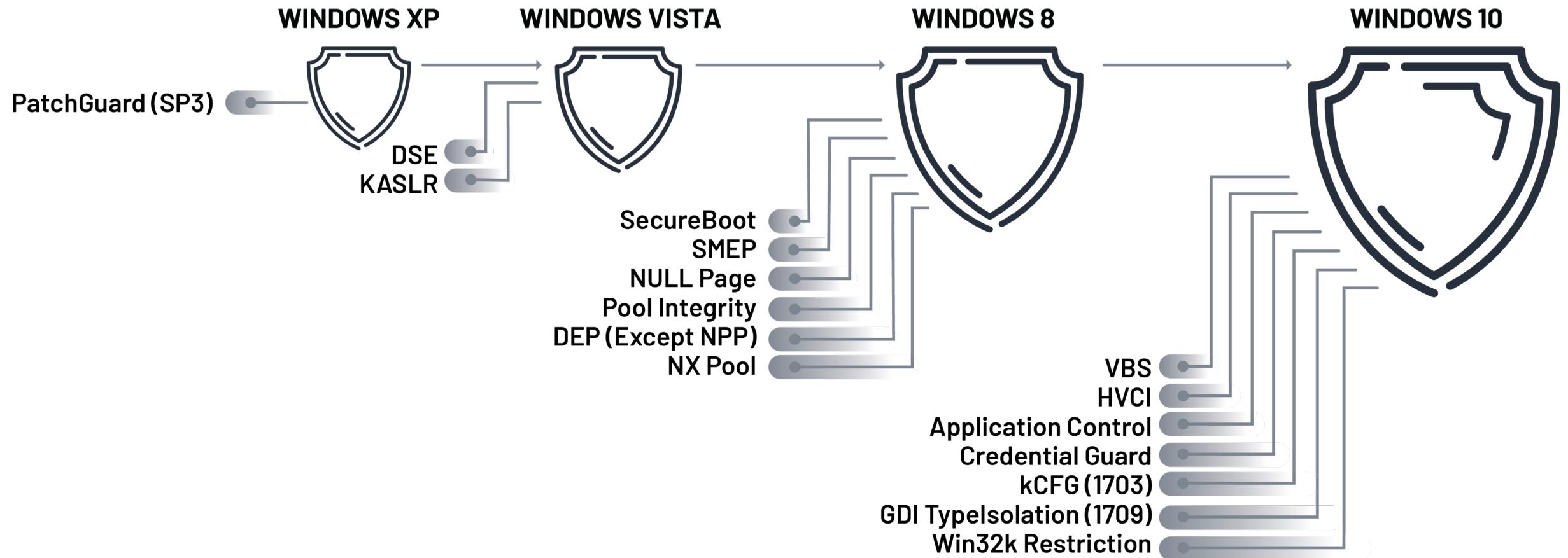
Exploit To Load Driver – Uroburos,
Duqu2, DoublePulsar

Defender

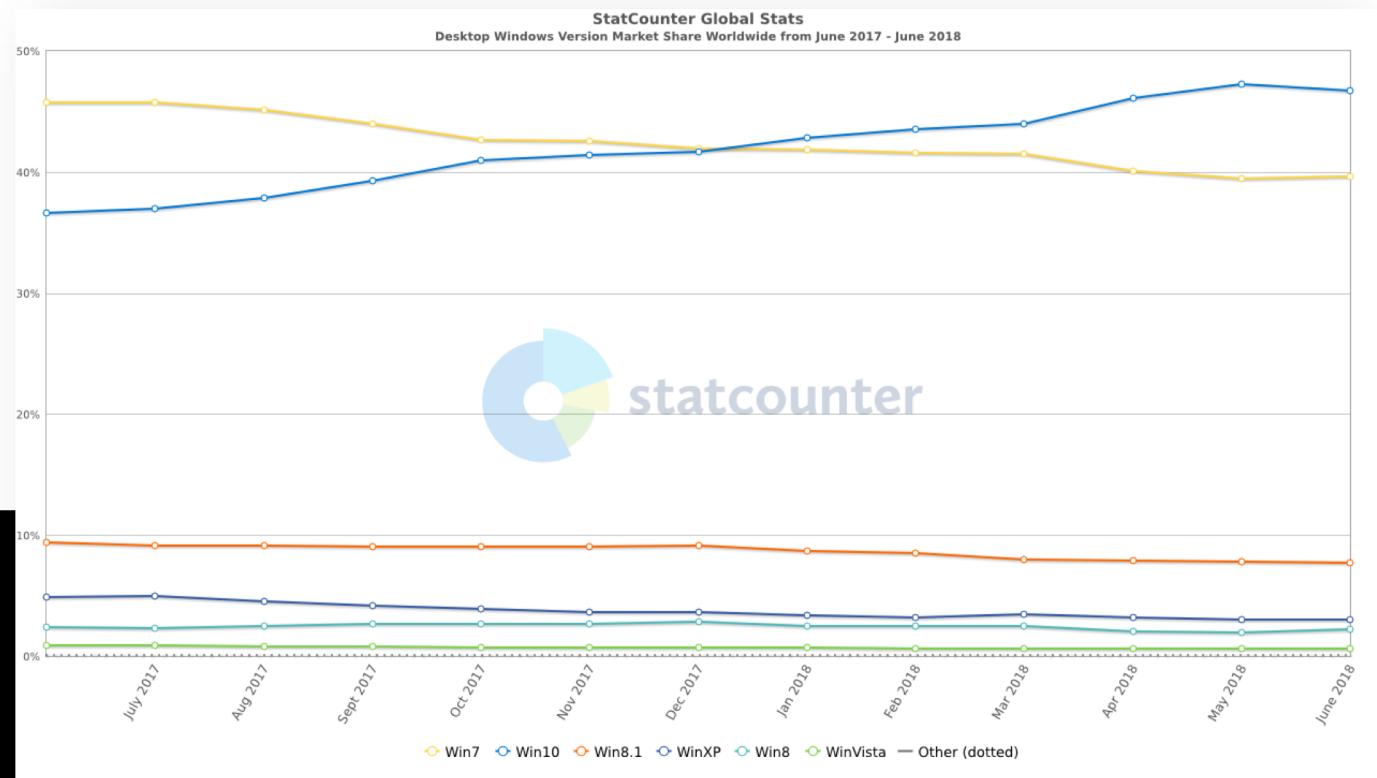


Virtualization Based Security (VBS)

KERNEL MITIGATIONS ACROSS WINDOWS VERSIONS



Adoption Rate



Windows 10 is less than half of worldwide installations. **How many are HVCI?**

AGENDA

PART 1

Evolution of kernel mode threats and platform protections

PART 2

Offensive tradecraft to evade platform protections

PART 3

Augmenting OS defenses

INTERNAL RvB

FILELESS KERNEL MODE IMPLANT

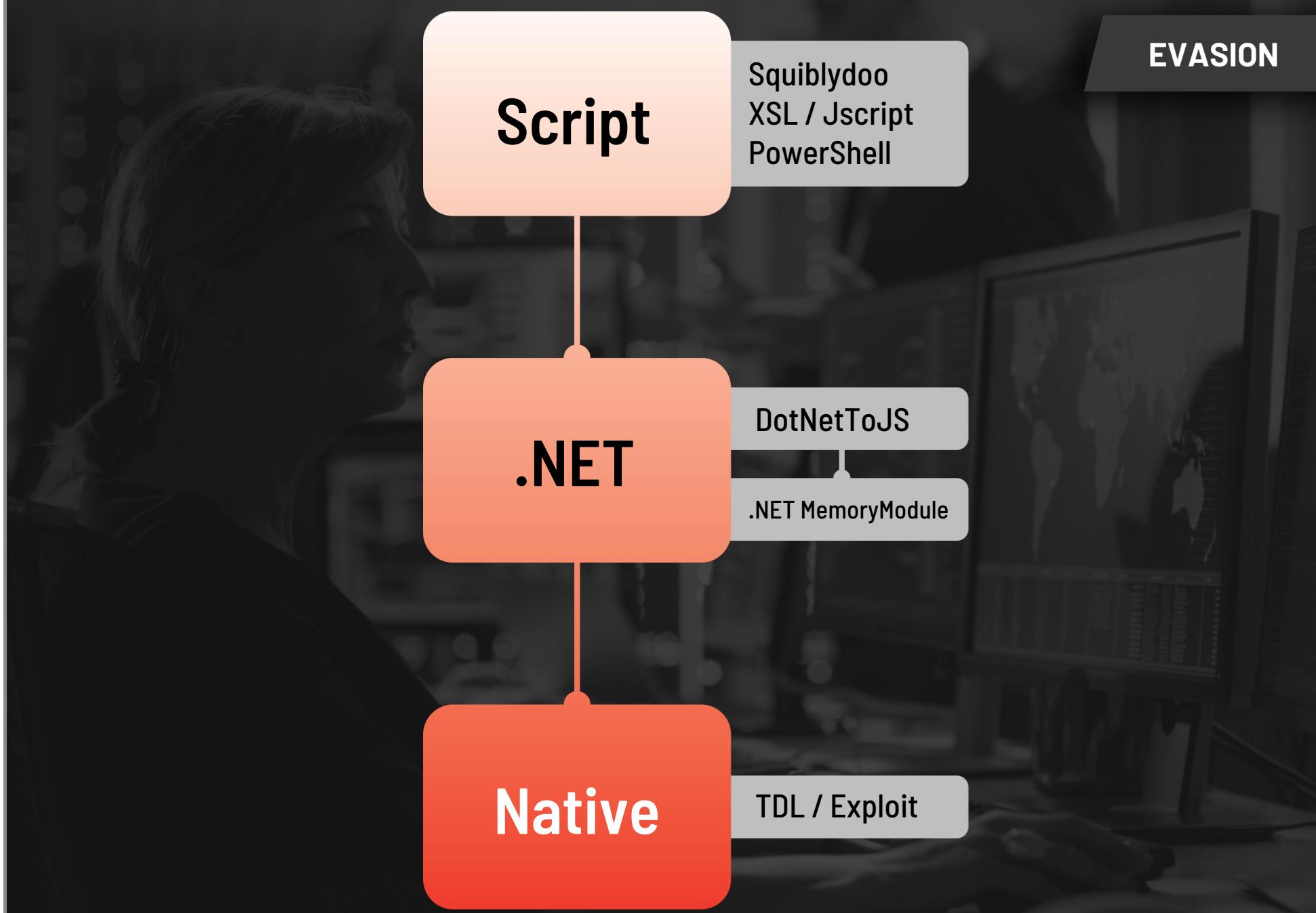
- Evade Blue Team by moving to the kernel
- Kernel dev environment
 - VS 2015, Windows Driver Kit (WDK) 10
 - Virtual KD / Windbg
- Turla Driver Loader (TDL)



IMPLANT DESIGN

- Kernel mode ONLY
- Winsock Kernel (WSK) for C2
- Triggerable (no beacons)
- Basic backdoor functionality

IMPLANT LOADER

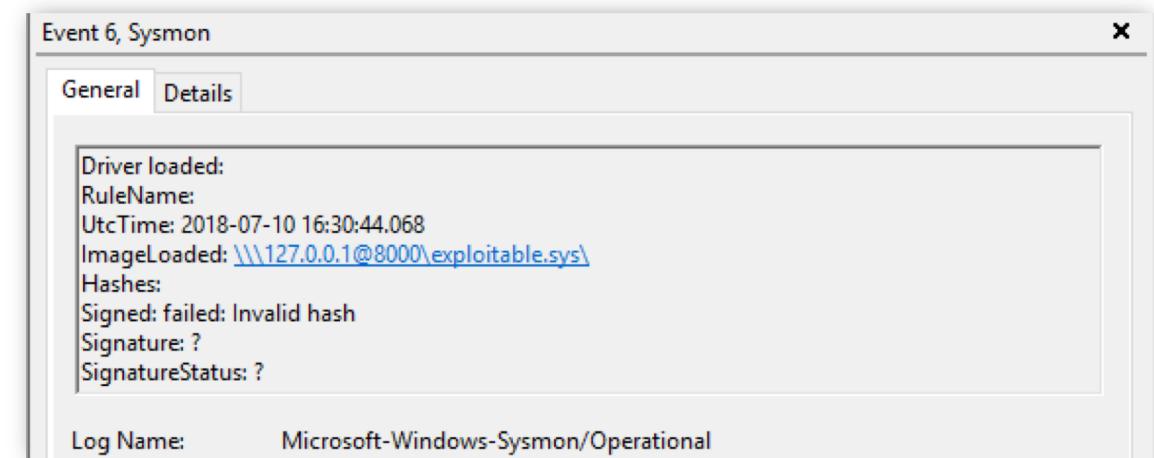


EVASION

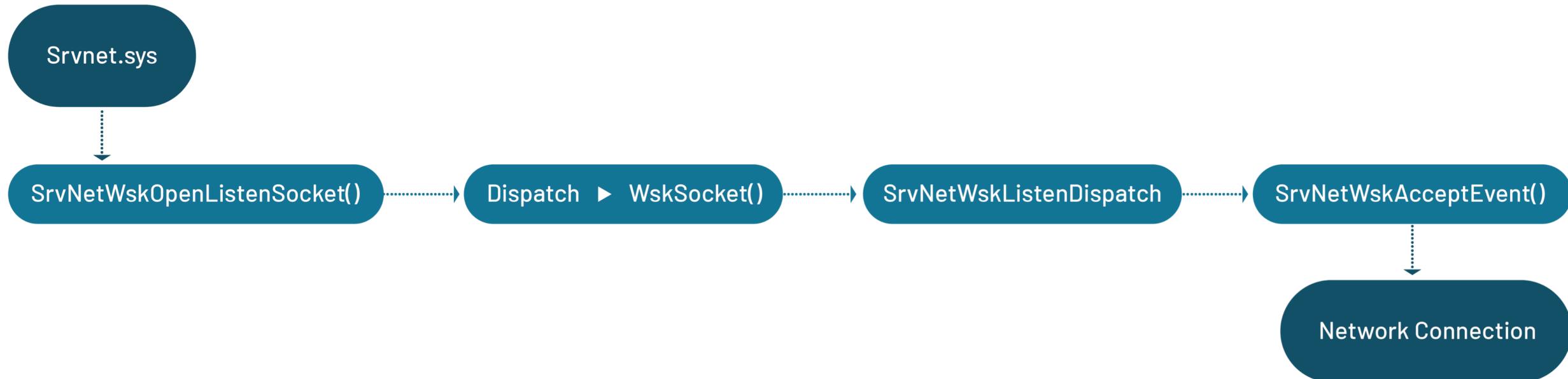
STAYING DISKLESS

NETWORK DRIVER LOAD

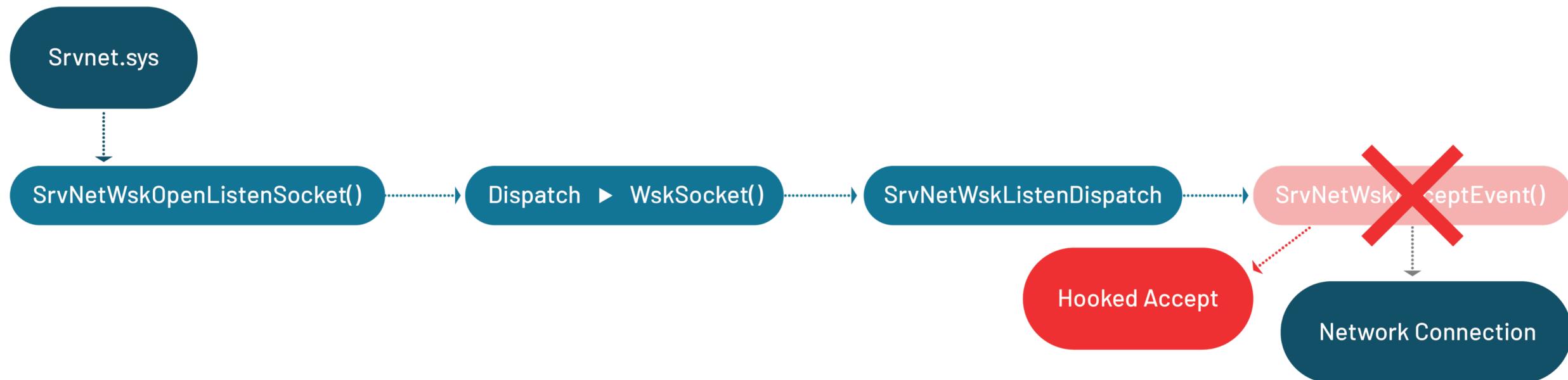
- Avoid dropping driver to disk by loading it with WebDAV:
 - ZwLoadDriver(\Device\WebDavRedirector\;\\127.0.0.1@8000\exploitable.sys)
- Sysmon path conversion bug



STEALTHY NETWORK COMMS



STEALTHY NETWORK COMMS



EVASION

DEMO

EVADING VBS / HVCI

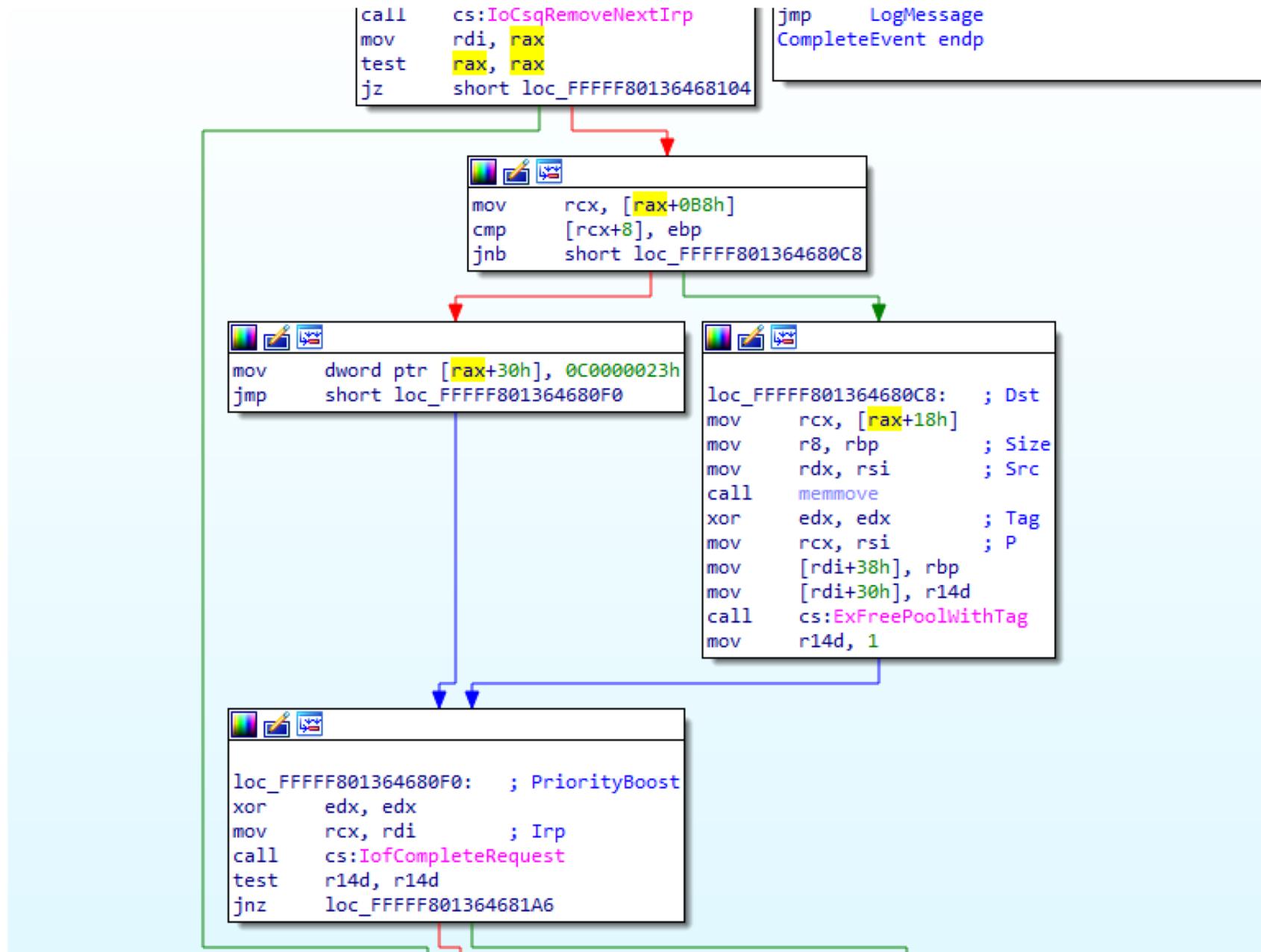
- Virtual box driver vulnerability is a no-go, won't even load under HVCI
- Step one was to find a vulnerable driver which could load under HVCI. www.greyhathacker.net has tons of POCs (Parvez Anwar)
- BYOV – Bring your own vulnerability. Choose wisely!
 - Static 1-byte write vs
 - Write *what where

EVADING VBS / HVCI

DATA DRIVEN ATTACKS

- Modification of kernel memory can significantly compromise the integrity of the system
- IAT patching
 - Even if IAT was protected by VBS, there are many other targets
- Disable EDR kernel-user communications
- Disable security focused kernel ETW providers
 - Microsoft-Windows-Threat-Intelligence
- Elevate Privileges – Token or Handle
- Data / Data Corruption





 xrefs to EtwThreatIntProvRegHandle

Direction	Type	Address	Text
	r	EtwTiLogAllocExecVm+19D	mov rcx, cs:EtwThreatIntProvRegHandle
	r	EtwTiLogAllocExecVm+34	mov rcx, cs:EtwThreatIntProvRegHandle
	r	EtwTiLogAllocExecVm+5C	mov r14, cs:EtwThreatIntProvRegHandle
	r	EtwTiLogMapExecView+19D	mov rcx, cs:EtwThreatIntProvRegHandle
	r	EtwTiLogMapExecView+34	mov rcx, cs:EtwThreatIntProvRegHandle
	r	EtwTiLogMapExecView+5C	mov r14, cs:EtwThreatIntProvRegHandle
	r	EtwTiLogProtectExecVm+19D	mov rcx, cs:EtwThreatIntProvRegHandle
	r	EtwTiLogProtectExecVm+34	mov rcx, cs:EtwThreatIntProvRegHandle
	r	EtwTiLogProtectExecVm+5C	mov r14, cs:EtwThreatIntProvRegHandle
	r	EtwTiLogQueueApcThread+2BC	mov rcx, cs:EtwThreatIntProvRegHandle
	r	EtwTiLogQueueApcThread+57	mov rcx, cs:EtwThreatIntProvRegHandle
	r	EtwTiLogQueueApcThread+8A	mov r15, cs:EtwThreatIntProvRegHandle
	r	EtwTiLogQueueApcThread:loc_1405403F6	mov rsi, cs:EtwThreatIntProvRegHandle
	r	EtwTiLogReadWriteVm+199	mov rcx, cs:EtwThreatIntProvRegHandle
	r	EtwTiLogReadWriteVm+4D	mov rbx, cs:EtwThreatIntProvRegHandle
	r	EtwTiLogSetContextThread+1AA	mov rcx, cs:EtwThreatIntProvRegHandle
	r	EtwTiLogSetContextThread+221	mov rdi, cs:EtwThreatIntProvRegHandle
	r	EtwTiLogSetContextThread+59	mov rcx, cs:EtwThreatIntProvRegHandle
	r	EtwTiLogSetContextThread+8C	mov rdi, cs:EtwThreatIntProvRegHandle
	r	EtwTiLogSetContextThread:loc_1405BD...	mov rcx, cs:EtwThreatIntProvRegHandle
	r	EtwpInitialize+380	lea r9, EtwThreatIntProvRegHandle

Return address protection with hardware

We have worked with Intel on designing a hardware-assisted solution for return address protection

Initial attempt to implement stack protection in software failed

REDTEAM designed software shadow stack (RFG) did not survive internal offensive research

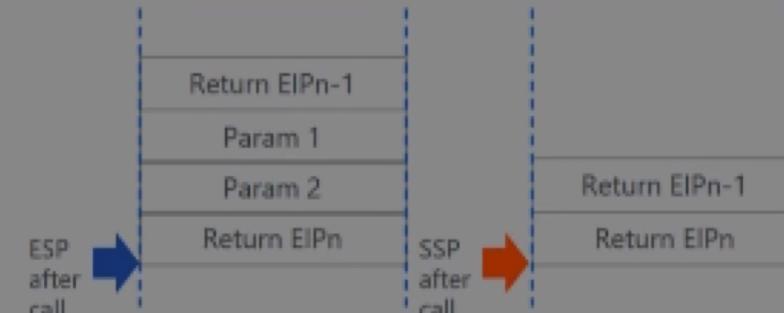
Control-flow Enforcement Technology (CET)

Indirect branch tracking via ENDBRANCH

Return address protection via a shadow stack

Hardware-assists for helping to mitigate control-flow hijacking & ROP

Robust against our threat model



Call pushes return address on both stacks

Ret/ret_imm
pops return address from both stack
Exception if the return addresses don't match

No parameters passing on shadow stack

Hardware-assists for helping to mitigate control-flow hijacking & ROP

Robust against our threat model



EVADING VBS / HVCI

CODE RE-USE

- Create a "surrogate" thread, put it to sleep
- Find location of stack [_ETHREAD]
- Build rop chain
- Hook stack -> overwrite NtWaitForSingleObject pointer with pivot gadget
- Resume thread (ReleaseMutex)



EVADING VBS / HVCI

CODE RE-USE

- Dynamically build chain based on number of arguments in target function
- We have 10 gadgets for full N-Argument function call
- Pivot (pop rsp, ret)
- Restore the 16 bytes overwritten on the stack
- Setup arguments in registers/stack
- Call target function
- Save return value (rax)
- Restore R14 (mutex object)
- Unpivot

EXAMPLE - WriteProcessMemory

```
NTSTATUS WPM(DWORD_PTR targetProcess, DWORD_PTR destAddress, void * pBuf, SIZE_T Size)
{
    SIZE_T Result;
    DWORD_PTR srcProcess = CallFunction("PsGetCurrentProcess");

    LONG ntStatus = CallFunction("MmCopyVirtualMemory", srcProcess,
        (DWORD_PTR)pBuf, targetProcess, destAddress, Size, KernelMode, (DWORD_PTR)&Result);

    return ntStatus;
}
```

EVASION

DEMO

AGENDA

PART 1

Evolution of kernel mode threats and platform protections

PART 2

Offensive tradecraft to evade platform protections

PART 3

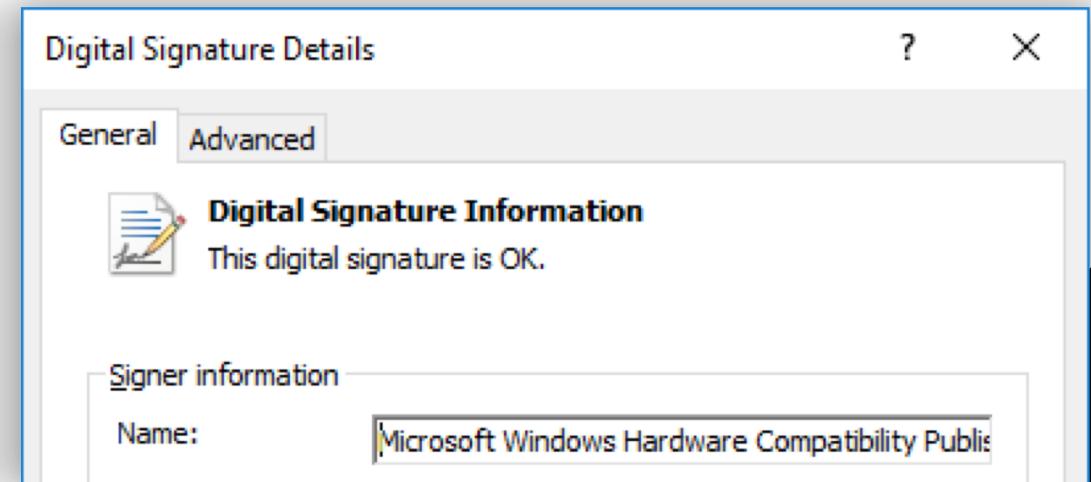
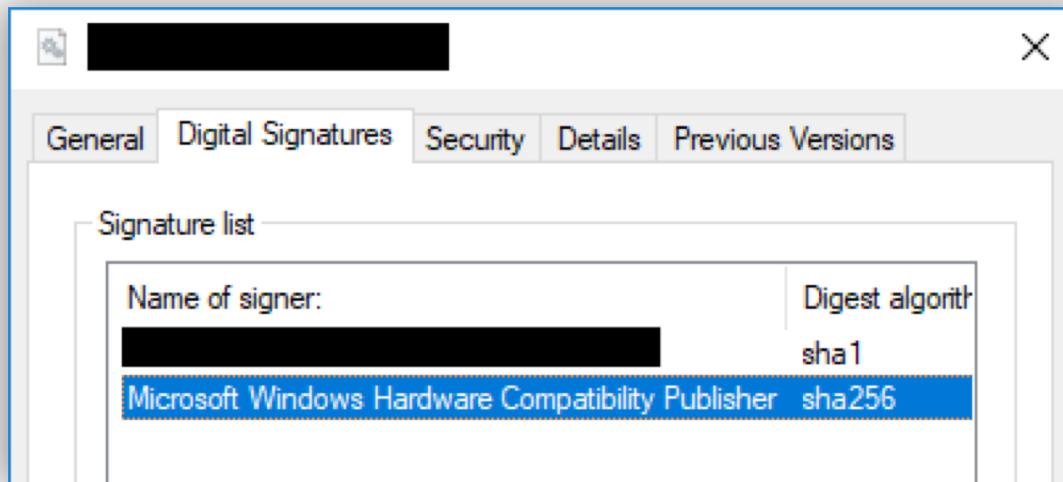
Augmenting OS defenses

DRIVER EVENTING

- You should very carefully monitor driver load events
- Look for low prevalence drivers
- Identify known-exploited drivers (blacklist)
- Free Instrumentation:
 - Sysmon Event ID 6: Driver loaded
 - Windows Defender Application Control (DG) Audit Mode

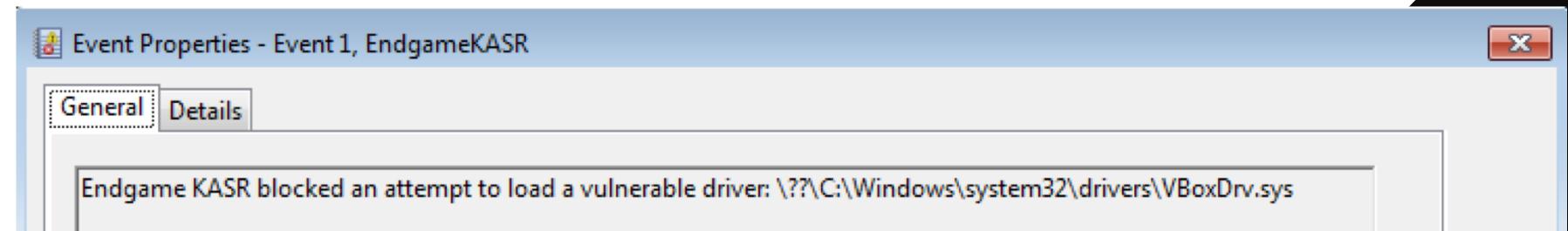
REDUCING ATTACK SURFACE

- Limit kernel drivers to WHQL



REDUCING ATTACK SURFACE

- Blacklisting Exploitable Drivers
 - Default in Win10 RS5 HVCI
- Endgame Kernel Attack Surface Reduction (KASR)
 - No configuration required
 - Free. Available at www.endgame.com



KERNEL MEMORY COLLECTION

Problem

Full memory acquisition and offline analysis does not scale

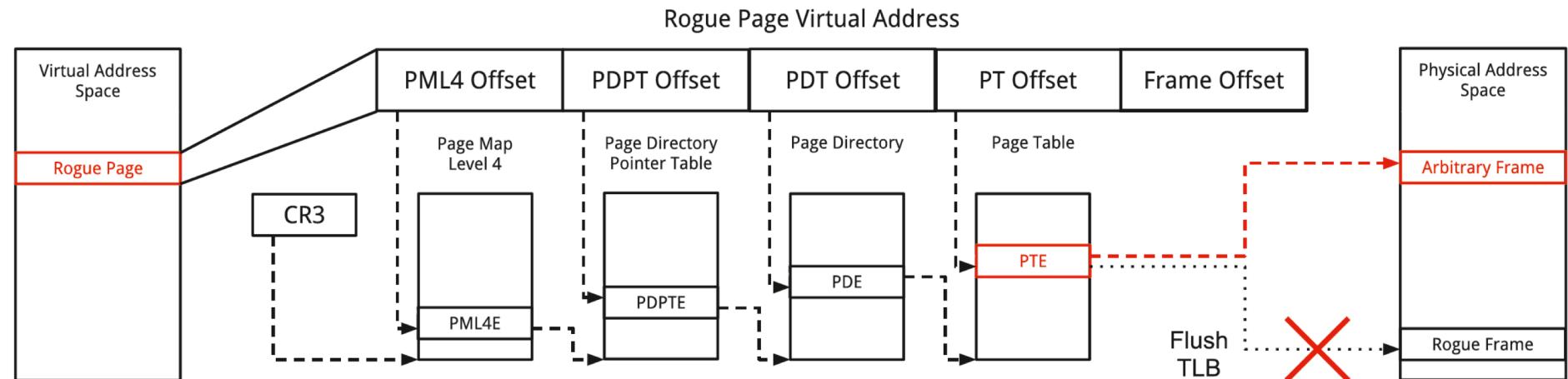
Solution

Leverage the same techniques to read kernel memory at the endpoint and perform analysis on the endpoint

- \Device\PhysicalMemory
- MmMapIoSpace
- MmMapMemoryDumpMdl
- Page Table Entry (PTE)
Remapping

KERNEL MEMORY COLLECTION

- Page Table Entry(PTE) remapping
 - Invented by Stütten and Cohen
 - Most performant in testing
 - More resilient to tampering from rootkits (No API to hook)
 - Based on CPU architecture, so it is cross-OS-compatible
 - Modify the PTE of a controlled chunk of memory to point to where you want to read



FUNCTION POINTER INTEGRITY SCAN

- Generically detect function pointer hook
- Locate function pointers by walking relocation tables
- Endgame Marta
 - Free. Available at www.endgame.com

Consider a hit if...

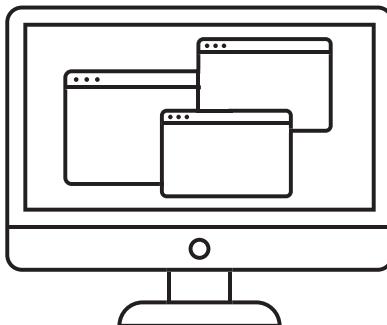
Originally pointed to
+X section in on-
disk copy of driver

Does not point to a
loaded driver in
memory



Points to
executable memory

REALTIME DETECTION OF UNBACKED KERNEL CODE



Performance Monitoring Unit (PMU)

- Built into most CPUs in the last decade
- Programmable to count specific events
- Can generate an interrupt when a certain number of events occur

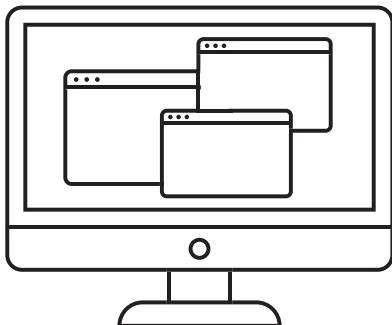
Detect unbacked kernel shellcode / unbacked drivers

- Program PMU to fire interrupts on some event likely to occur inside unbacked driver.
- When interrupt fires, validate IP belongs to a driver

DEMO

DEFENSES

REALTIME DETECTION OF UNBACKED KERNEL CODE



Weaknesses

- FP from unbacked drivers (including PatchGuard)
- Vulnerable to PMU disablement via MSR
- Detection driver is susceptible to data attacks

HYPERVERISOR APPROACHES

- MemoryMon
 - Leverage EPT to detect unbacked kernel code execution
- AllMemPro
 - Mitigates data attacks by isolating driver data with hypervisor
 - Based on same platform as MemoryMon
- SecVisor
 - CMU research from 2007
 - Early implementation of hypervisor enforced code integrity

HARDENING THE WINDOWS KERNEL AGAINST ROP

Return address protection with hardware

We have worked with Intel on designing a hardware-assisted solution for return address protection

Initial attempt to implement stack protection in software failed

REDTEAM designed software shadow stack (RFG) did not survive internal offensive research

Control-flow Enforcement Technology (CET)

Indirect branch tracking via ENDBRANCH
Return address protection via a shadow stack

Hardware-assists for helping to mitigate control-flow hijacking & ROP

Robust against our threat model

The diagram illustrates the stack usage on a near CALL. It shows two stacks: the Real Stack (ESP) and the Shadow Stack (SSP). The ESP stack contains Return EIPn-1, Param 1, Param 2, and Return EIPn. The SSP stack contains Return EIPn-1, Return EIPn, and Param 1. An arrow points from the ESP stack to the SSP stack after the call, labeled "SSP after call".

Call pushes return address on both stacks

Ret/ret_imm
pops return address from both stack
Exception if the return addresses don't match

No parameters passing on shadow stack

Hardware-assists for helping to mitigate control-flow hijacking & ROP

Robust against our threat model

HARDENING THE WINDOWS KERNEL AGAINST ROP

- CFG provides forward-edge CFI, but there is no reverse-edge CFI
 - RFG was cancelled
 - Intel CET requires future hardware

IDEA

Provide reverse-edge kernel CFI using the PMU

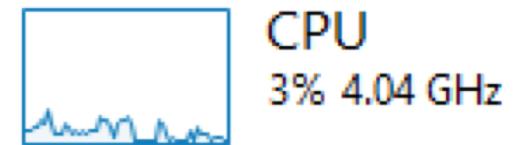
HARDENING THE WINDOWS KERNEL AGAINST ROP

ROP tends to generate lots of mispredictions

- Scan drivers to identify call/return sites
- Configure LBR to record CPL0 near returns
- Configure PMU fire PMI when BR_MISP_RETIRE counter overflows
- Validate every new LBR entry is call-preceded

Performance

- Protection tunable from deterministic to probabilistic
- JetStream benchmark on Intel Skylake 6700K: ~1.1% overhead



DEMO

DEFENSES

HARDENING THE WINDOWS KERNEL AGAINST ROP

- Limitations
 - Vulnerable to malicious drivers. Hypervisor protection needed.
 - wrmsr IA32_PERF_GLOBAL_CTRL, 0
 - Data attack against policy bitmap
 - Requires available PMU and LBR
 - No VMware/Hyper-V support
 - Incompatible with obfuscated drivers
 - Windows ships with two: clipsp.sys and peauth.sys
- Something similar exists on Linux

CONCLUSION

- Windows platform security has gotten much better in the last decade. However, **kernel mode threats are still a big concern**
- Use Windows 10 with SecureBoot and HVCI
- Require EV/WHQL drivers via code integrity policy
- Monitor and hunt on driver load events

QUESTIONS?

REFERENCES

- <https://www.virusbulletin.com/virusbulletin/2008/08/yet-another-rustock-analysis>
- <https://securelist.com/rustock-and-all-that/36217/>
- <https://securelist.com/tdss/36314/>
- https://people.eecs.berkeley.edu/~pearce/papers/zeroaccess_tr_2013.pdf
- <https://www.symantec.com/content/dam/symantec/docs/security-center/white-papers/trojan-zeroaccess-infection-analysis-12-en.pdf>
- <https://nakedsecurity.sophos.com/zeroaccess/>
- <https://msdn.microsoft.com/en-us/library/bb530195.aspx>
- <https://securelist.com/tdss-tdl-4/36339/>
- https://www.coseinc.com/en/index.php?rt=download&act=publication&file=TDL4_Carrier_to_Glupteba.pdf
- <https://resources.infosecinstitute.com/uefi-and-tpm/>
- http://www.uefi.org/sites/default/files/resources/UEFI_Secure_Boot_in_Modern_Computer_Security_Solutions_2013.pdf
- <https://www.intel.com/content/dam/www/public/us/en/documents/white-papers/security-technologies-4th-gen-core-retail-paper.pdf>
- https://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/waterbug-attack-group.pdf
- <https://www.sekoia.fr/blog/windows-driver-signing-bypass-by-derusbi/>

REFERENCES

- https://s3-eu-west-1.amazonaws.com/khub-media/wp-content/uploads/sites/43/2018/03/09133534/The-Slingshot-APT_report_ENG_final.pdf
- https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2018/03/07205202/The_Mystery_of_Duqu_2_0_a_sophisticated_cyberespionage_actor_returns.pdf
- <https://securelist.com/the-duqu-2-0-persistence-module/70641/>
- <https://zerosum0x0.blogspot.com/2017/04/doublepulsar-initial-smb-backdoor-ring.html>
- <http://www.alex-ionescu.com/blackhat2015.pdf>
- <http://gs.statcounter.com/windows-version-market-share/desktop/worldwide/>
- <https://github.com/hfiref0x/TDL>
- <http://www.greyhathacker.net>
- <https://posts.specterops.io/threat-detection-using-windows-defender-application-control-device-guard-in-audit-mode-602b48cd1c11>
- https://www.dfrws.org/sites/default/files/session-files/pres-anti-forensic_resilient_memory_acquisition.pdf
- <https://www.cs.cmu.edu/~arvinds/pubs/secvisor.pdf>
- <https://github.com/tandasat/MemoryMon>
- <https://igorkorkin.blogspot.com/2018/03/hypervisor-based-active-data-protection.html>
- https://infocon.org/cons/SyScan/SyScan%202013%20Singapore/SyScan%202013%20Singapore%20presentations/SyScan2013_DAY1_SPEAKER05_Georg_Wlcherski_Taming_ROP_ON_SANDY_BRIDGE_syscan.pdf
- <https://recon.cx/2015/slides/recon2015-05-peter-hlavaty-jihui-lu-This-Time-Font-hunt-you-down-in-4-bytes.pdf>