Detecting BadBIOS, Evil Maids, Bootkits, and Other Firmware Malware

SeaGL

October 6, 2017

Paul English, PreOS Security
Twitter: @penglish_preos
penglish@preossec.com
https://preossec.com/
personal blog of Lee Fisher (CTO):
https://firmwaresecurity.com/

Agenda

- Technology: some system/peripheral firmware.
- Threats: types of and existing firmware-level malware.
- Tools: open source (and a few freeware) firmware defect/security analysis tools.
- Guidance: Introduce some basic advice for protecting your firmware from attackers, mostly based on NIST SP 800-147 lifecycle guidance.

About

- Me:System Administrator since 1998, added “manager” more recently. Former board member for the League of Professional System Administrators (LOPSA), Board Member for Seattle Privacy Coalition, organizer for Seattle Techno Activism 3rd Mondays
- Employer: PreOS Security is a firmware security startup that focuses on defensive tools for enterprises. I am co-founder and CEO.
- Presentation: discusses existing open source (and a few freeware) firmware diagnostic/security tools, combined with some existing NIST guidance. No new security exploits or vulnerabilities or research.

Scope

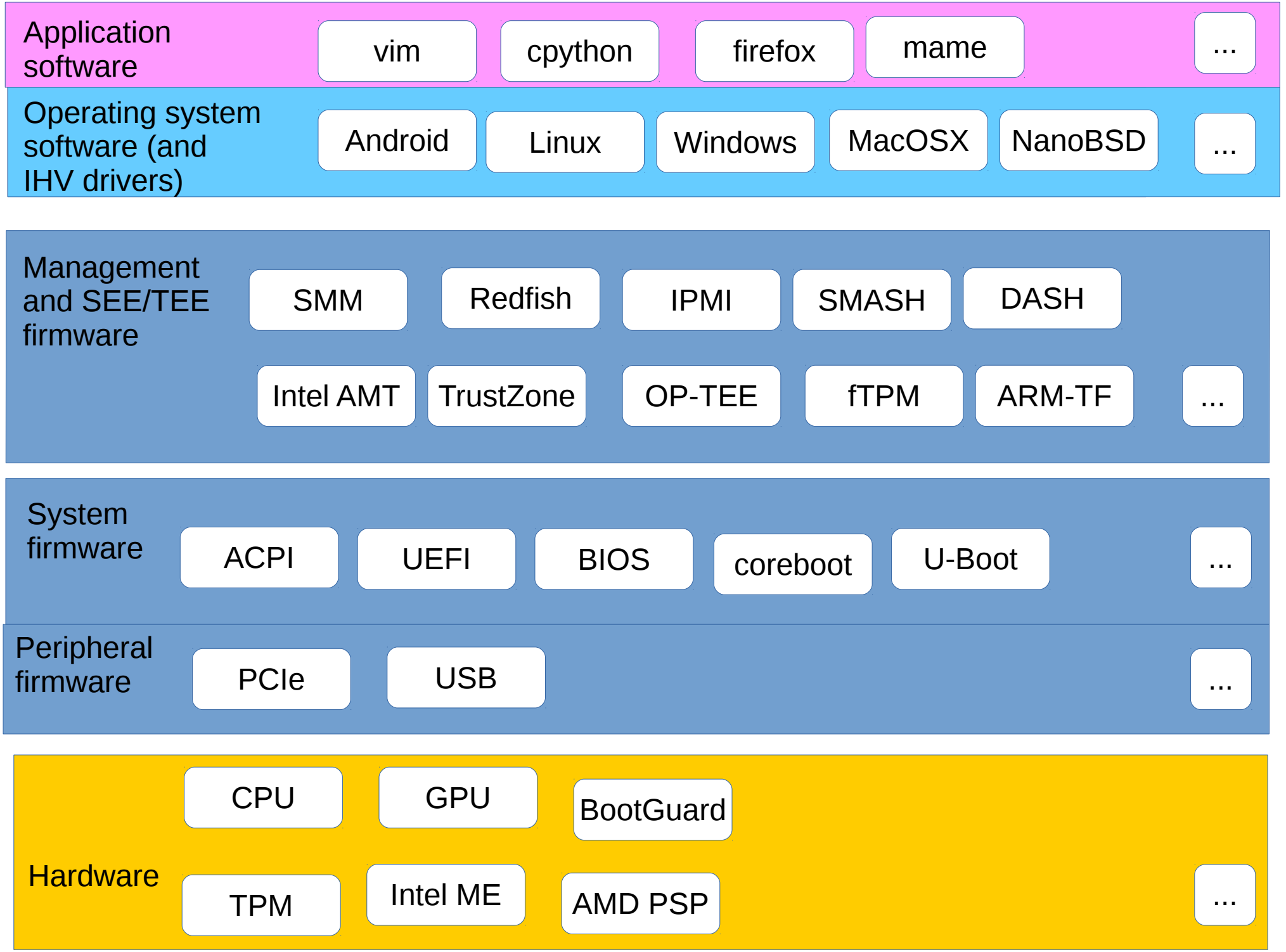
- Using ‘System Firmware’ aka ‘Platform Firmware’ (BIOS, UEFI, ACPI, etc.) definition of firmware, not embedded OS firmware (eLinux, Windows IoT, Android, QNX, etc.).
- Focusing on Intel x64 UEFI-based systems. Though much applies to x86, AArch32, AArch64 UEFI systems, and some BIOS systems.
- Mostly focusing on UEFI-style systems, not coreboot or U-Boot or other boot loader technologies.

Why are we talking about this?

- Industry (standards bodies, governments):
  - NIST SP 800-(147/147b/155/193) are guidance not policy, few follow it
  - No mandatory policy explicitly requires addressing firmware security
- Vendors (OEMs, ODMs):
  - Most still shipping insecure systems
  - None providing golden image hashes
  - Insufficient changelog documentation
  - Some never update firmware after initial release
- Consumers (you):
  - Most ignorant of the problem
  - Not spending time to learn to solve problem
  - Continuing to purchase insecure systems from vendors

NEXT MODULE

- Tech
- Threats
- Tools
- Guidance


Application software
vim
cpython
firefox
mame
...
Operating system software (and IHV drivers)
Android
Linux
Windows
MacOSX
NanoBSD
...
Management and SEE/TEE firmware
SMM
Redfish
IPMI
SMASH
DASH
Intel AMT
TrustZone
OP-TEE
fTPM
ARM-TF
...
System firmware
ACPI
UEFI
BIOS
coreboot
U-Boot
...
Peripheral firmware
PCIe
USB
...
Hardware
CPU
GPU
BootGuard
TPM
Intel ME
AMD PSP
...

Rings of Protection

- Early Intel processor security model (MS-DOS-era, aka no security):
  - Ring 3 (outermost): user mode, apps
  - Rings 2-1: often not used.
  - Ring 0 (innermost): kernel mode, drivers/OS kernel
- Simplistic model, 3 (aka user space) and 0 (aka kernel space), ignoring nuances of physical and virtual firmware and silicon levels.
- Invisible Things Lab (ITL) and others have proposed adding new negative rings – below ring 0 – to clarify this (next slide).

Negative (Subzero) Rings

- Ring -1: VM, hypervisor/CPU (Intel VMX, AMD SVM)
- Ring -2: Intel SMBIOS, SMM
- Ring -3: Firmware (BIOS, UEFI, coreboot, U-Boot, ...)
- Ring -4: hardware, silicon exploits
- Firmware (-3) unites all of these rings: it is the gateway to access SMM (-2) and HW (-4), and also works on virtualized systems (-1), and can be accessed (and controlled) by OSes/drivers (0) and apps (3).
- We will be spending most of our time talking about ring -3, focusing on UEFI-flavored firmware…

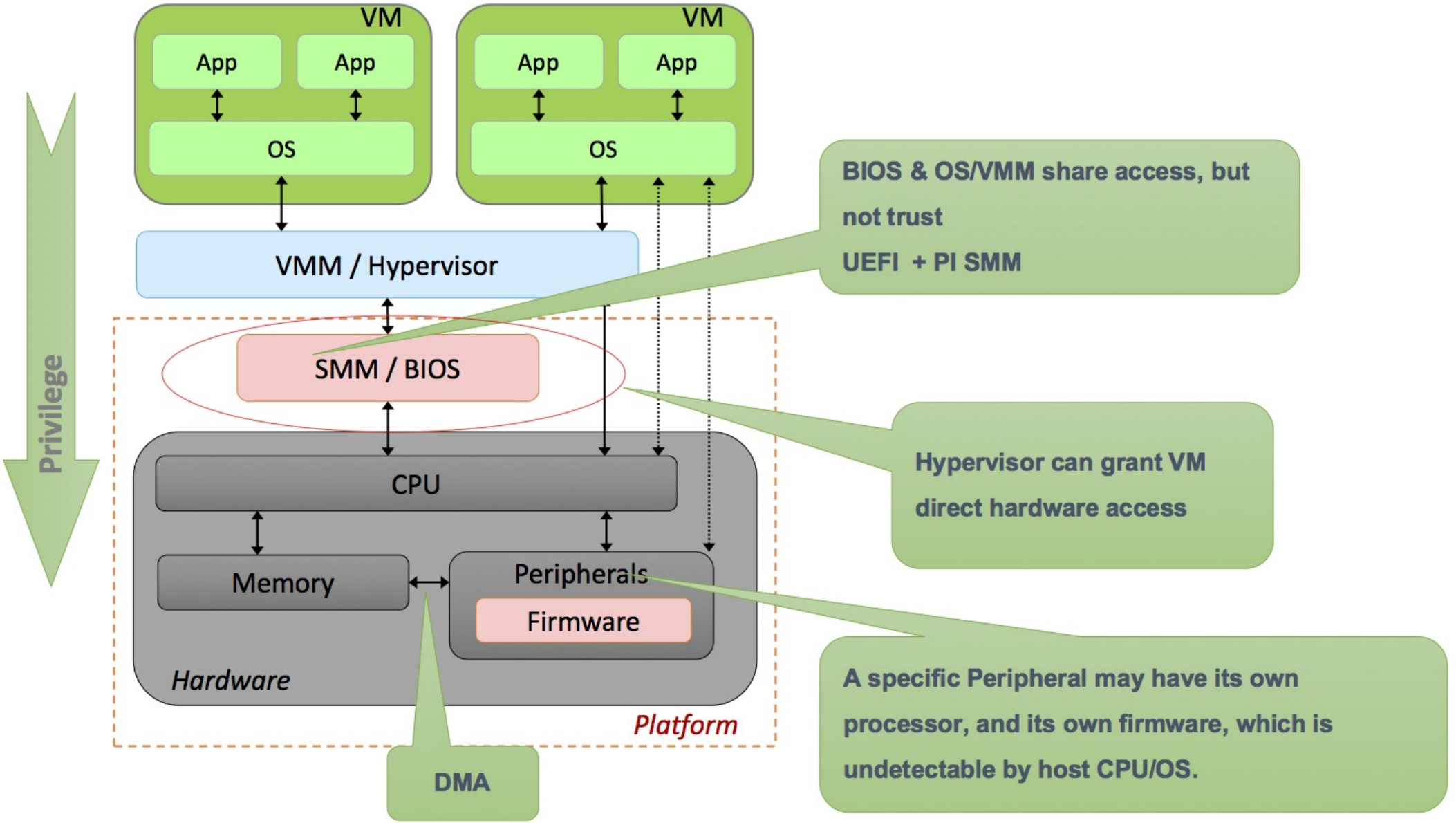
Intel Systems Mangement Mode

- Systems Management Mode (SMM)
- AKA “Ring -2”
- Intel-based technology, mode of CPU other than Real or Protect Mode, where attackers prefer to be operating in. Basically, the first attempts of a TEE/SEE for Intel x86 systems.
- UEFI recently updated the Platform Initialization (PI) spec, abstracts both Intel SMM and ARM TrustZone as “MM”, Management Mode.

Image credit

- Image from next slide comes from the Blackhat-US-2017 presentation: "Firmware is the new Black – Analyzing Past 3 years of BIOS/UEFI Security Vulnerabilities", by Bruce Monroe & Rodrigo Rubira Branco & Vincent Zimmer, Intel Corp. https://github.com/rrbranco/BlackHat2017/

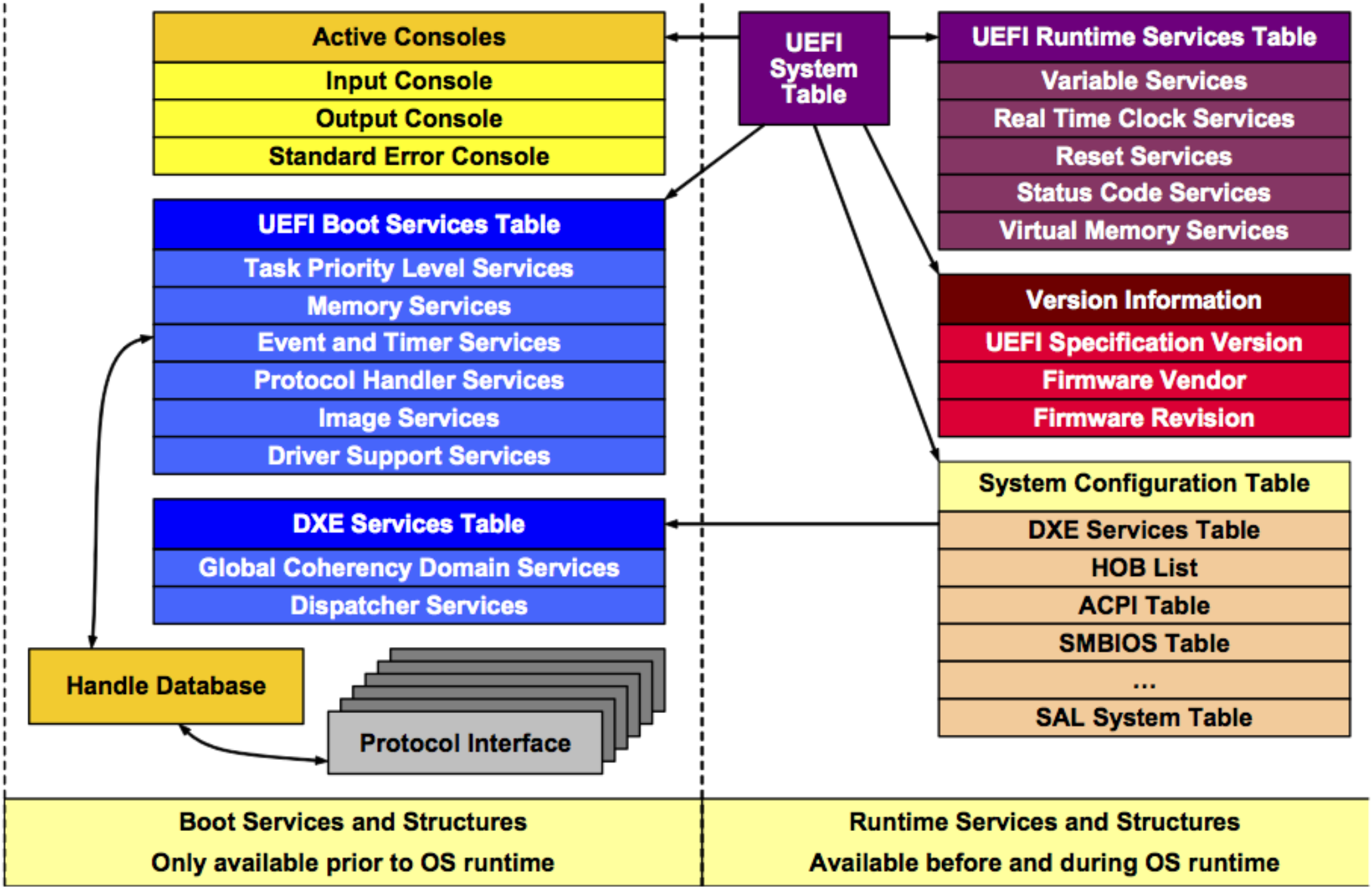
Modern x64 UEFI system arch



UEFI Services

- UEFI Boot Services
  - used by OS loader to transition from FW to OS kernel (or boot manager, a pre-OS UEFI app)
  - Only used during init, they're not available later.
- UEFI RunTime Services
  - Similar to BIOS interrupts like 10h (video) and 13h (disk), UEFI has services that the OS uses.
  - virtual memory, time, variables, console i/o, reset, capsule, memory, event/timer, protocol, image

Image source: UEFI PI spec



Secure Boot

- Secure Boot is an optional build-time feature of UEFI 2.x using multiple sets of keys to try and secure the firmware from malware.
- UEFI Variables Secure Boot Databases:
  - Platform Key (PK)
  - Key Exchange Key Database (KEK)
  - Secure Boot Signature Database (db)
  - Secure Boot Blacklist Signature Database (dbx)
  - Secure Boot Timestamp Signature Database (dbt)
  - Secure Boot Authorized Recovery Signature Database (dbr)

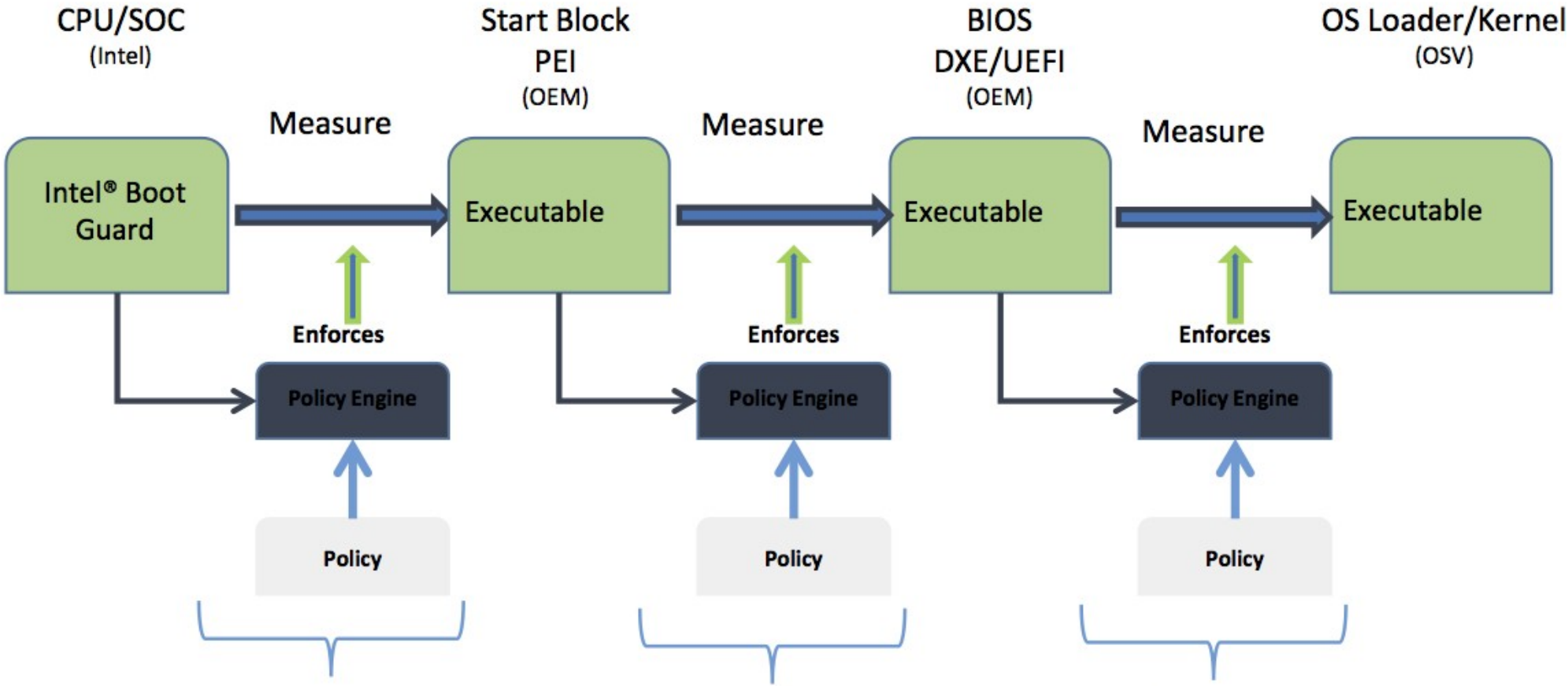
More secure boot flavors

- Besides default (insecure) boot modes of past, more secure boot modes are being used, to help detect attacks:
  - UEFI Secure Boot
  - TCG Measured Boot (uses TPM)
  - Trusted Boot (uses Intel TXT)
- Related technologies:
  - coreboot Verified Boot (Android, ChromeOS)
  - U-Boot Verified Boot

Image credit

- Image from next slide comes from the Blackhat-US-2017 presentation: "Firmware is the new Black – Analyzing Past 3 years of BIOS/UEFI Security Vulnerabilities", by Bruce Monroe & Rodrigo Rubira Branco & Vincent Zimmer, Intel Corp. https://github.com/rrbranco/BlackHat2017/

Intel/UEFI boot sequence



Intel® Device Protection Technology with Boot Guard

http://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/4th-gen-core-family-mobile-brief.pdf

OEM PI Verification Using PI Signed Firmware Volumes

Vol 3, section 3.2.1.1
of PI 1.3 Specification

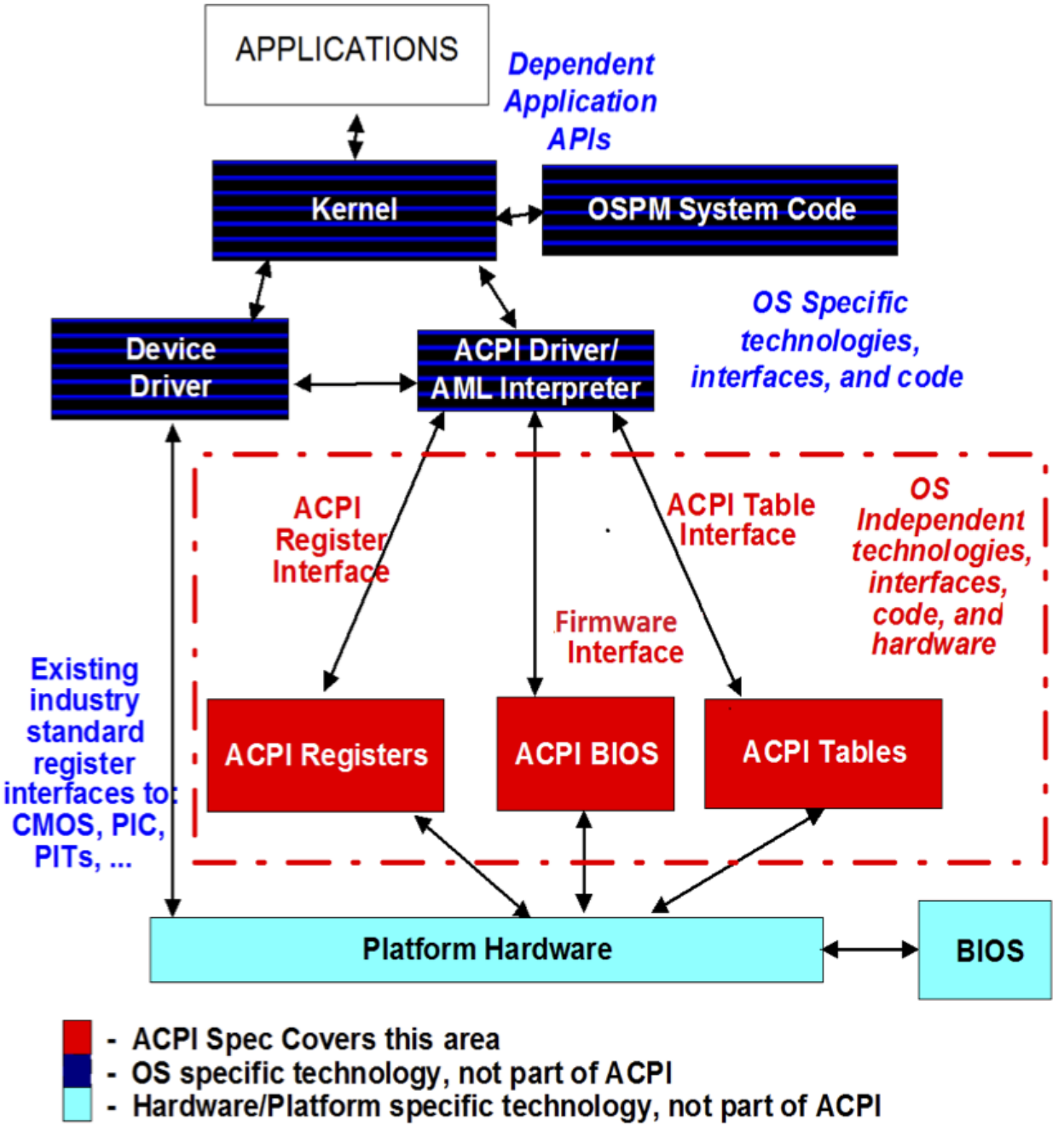
OEM UEFI 2.7 Secure Boot

Chapter 27.2 of
The UEFI 2.4
Specification

ACPI

- Advanced Configuration and Power Interface
- Successor to ISAPnP, APM and MP.
- ACPI used by both BIOS and UEFI. Initially only on Intel, now also on ARM systems.
- Dozens of ACPI ‘tables’ are defined.
- ACPI has an AML (ACPI Machine Language) and bytecode which the firmware and OS has to run.
- The UEFI Forum owns the ACPI specifications.

Image source: ACPI spec



PCIe

- PCIe is one of the main peripheral busses these days.
- In addition to main system firmware ROM, each PCIe device on the system can have option/expansion ROMs on their flash, often BIOS-based code as well as UEFI-based drivers, that enhances the platform firmware to support the PCIe device.
- NVMe and Thunderbolt are PCIe-based.
- See CHIPSEC_util's PCI command
- Recent Apple Mac EFI firmware has new boot key to press to disable loading of PCIe device option ROMs.

Other: USB, Hard disks, Monitors?

- “BadUSB” https://github.com/brandonlw/Psychson
- Hard disks (and SSDs): on-board controller. Sprites: http://spritesmods.com/?art=hddhack
- Monitors! - MonitorDarkly https://github.com/redballoonshenanigans/monitordarkly

NEXT MODULE

- Tech
- Threats
- Tools
- Guidance

Why Threaten Firmware?

- Stealth
  - As current tools & practices do not address firmware, compromises can remain undetected
- Persistence
  - Even if detected, issues can be more difficult to fix at the firmware level. Nobody wants to throw away hardware.
- Full Access
  - “Ring 0” or even “Negative Ring” access operate below most protection. Some protection relies on firmware features!

Evil Maid attack

- The “Evil Maid” attack is perhaps the best-known firmware attack.
- Results from physical access to hardware, game over.
- Evil Maids attack unattended systems, like the laptop left at a hotel while out at dinner, handed to TSA, insecure server room,...
- Attacks vary, from plugging in rogue PCIe/USB/Thunderbolt peripheral, attaching SPI/JTAG device, to complex de-chipping.
- Never leave unsecure computers unattended.

Supply Chain Verification

- New hardware is bad enough
  - Vendors do not provide golden image hashes for verification
  - Most vendors do not provide sufficient documentation for firmware features, or for updates, to distinguish changes in ROM that are vendor updates -vs- attacker malware.
  - Shipped hardware may be be intercepted by government agencies, or criminals in shipping companies; interdictions of shipped hardware, infected with malware.
- Grey market hardware is much worse
  - Basically the same problem as interdictions. Do you know how to factory-reset the firmware of all the used hardware you've acquired?
- Industry has no mechanism to verify silicon for threats.

Physical and network security

- Data centers and 'server rooms' should have good physical security.
- Desktops left at work are often accessible by evil maids in night shift cleaning staff.
- Mobile devices have roaming network security issues.
- Mobile devices left in hotels subject to evil maid attacks.
- New airline travel (“Travel 2.0”) restrictions enables evil maid attack for phones/laptops/etc of more and more travelers.
- Firmware management OOB network management must be isolated, encrypted, and authenticated (Intel AMT, Redfish, IPMI, iLO, …). Previously Ethernet-centric, now WiFi-based OOB solutions make this even harder to isolate.

Firmware Updates and OS attacks

- With UEFI, firmware updates are more standardized than with BIOS, and are now more easily called by user-mode applications. Recent OS platform integration has firmware updates included in OS updates (eg: Windows Update).
- If vendor has not locked down their firmware update mechanism to signed code and signed payloads, attacker can pivot off an OS root/admin exploit to embed bootkit into firmware.
- There are lots of OS/app-level security tools and rules out today: do any watch for (and selectively disallow) firmware-updates?

Existing UEFI malware

- The CHIPSEC project maintains a blacklist for UEFI malware, currently has 3 entries:
  - Hacking Team’s Vector-EDK
  - @cr4sh’s ThinkPwn
  - CIA Mac EFI malware
- A few security researchers have some POCs on Github.
- Old leaked goverment malware:
  - NSA (BIOS, not UEFI): DeityBounce, BananaBallot, JetPlow
  - CIA (Mac EFI): DarkMatter, DerStark, QuarkMatter, …

UEFI Secure Boot keys

- UEFI uses PKI for Secure Boot and code signing. UEFI does not have CRL/OSCP for checking for bad keys.
- The UEFI Forum maintains a list of bad keys used by UEFI Secure Boot. There are a few dozen entries in current DBX.
- Does your vendor provide tools to update the DBX? :-)

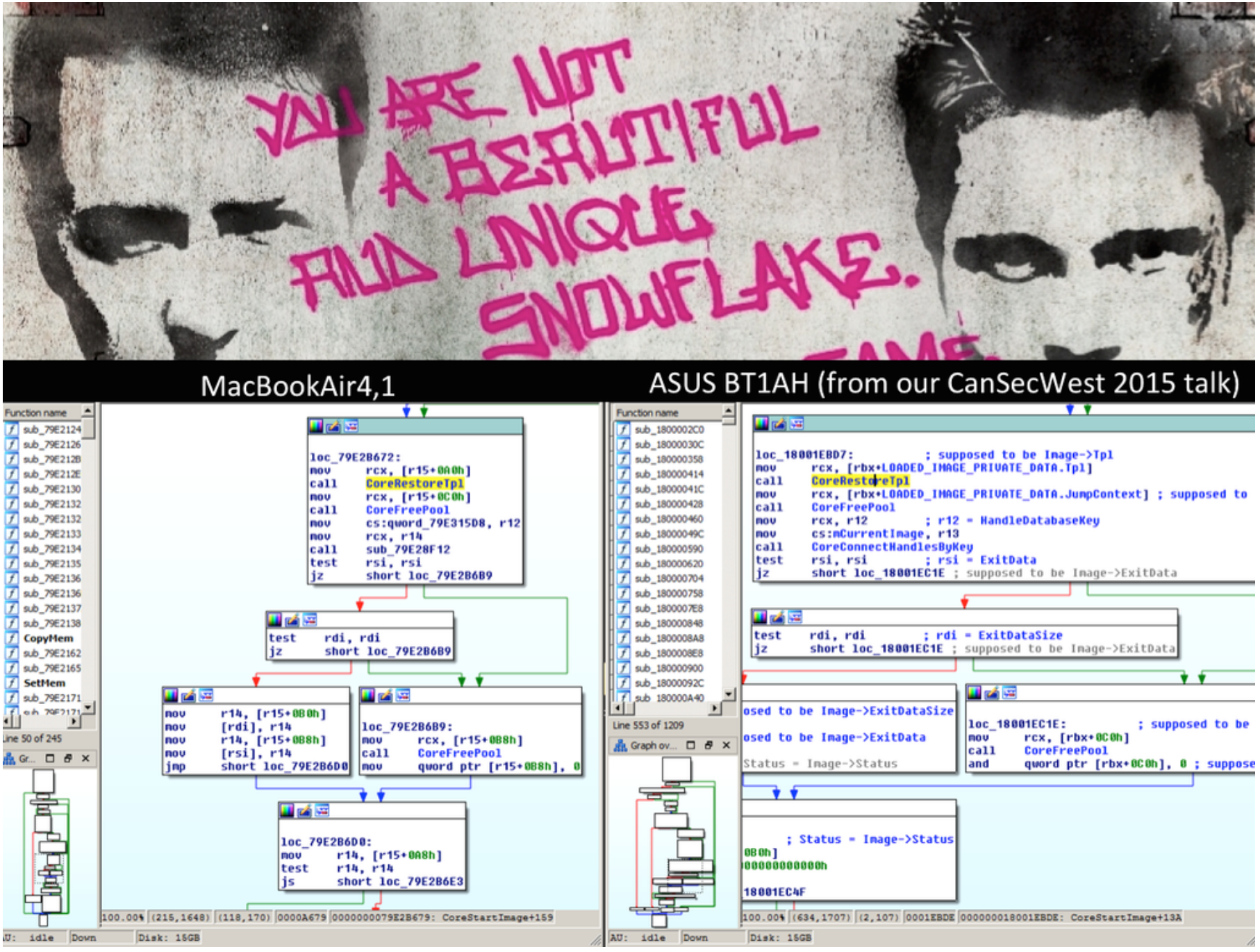
Hardware attacks

- Hardware attacks:
  - USB, Hak5 USB Rubber Ducky, etc.
  - PCIe, PCILeech-based, DMA/other attacks by ‘rogue hardware’ (inexpensive COTS dev board)
  - Thunderbolt
  - Rowhammer, memory attacks

Open Source == shared vulnerabilites

- OEM vendors share a common core UEFI codebase (tianocore.org).
- Next slide, image source:
- https://twitter.com/XenoKovah/status/623483244890189824
- “Dear firmware makers: ALL UEFI-RELATED FIRMWARE LOOKS THE SAME TO ATTACKERS! YOUR SECURITY THROUGH OBSCURITY IS GONE!”

Image source:
https://twitter.com/XenoKovah/status/623483244890189824



Firmware Bug Classes

- 1) Inconsistent power-transition checks
- 2) Race Condition
- 3) Trusting input
- 4) Measurement failures
- 5) Platform capability not properly configured
- 6) Security of meaningful assets exposed to untrusted entities
- 7) Hardware Misbehavior

- Classes defined in the Blackhat-US-2017 presentation: "Firmware is the new Black – Analyzing Past 3 years of BIOS/UEFI Security Vulnerabilities", by Bruce Monroe & Rodrigo Rubira Branco & Vincent Zimmer, Intel Corp. https://github.com/rrbranco/BlackHat2017/

NEXT MODULE

- Threats
- Tech
- Tools
- Guidance

Tool Scope

- These tools, as with presentation scope, are mostly focusing on UEFI and ACPI.
- Tools not covered: Intel ME, Intel AMT, IPMI, Redfish, USB, microcode, etc. Tool coverage for these varies!

  Two kinds of tool usages for firmware:

  - Live: testing against a live running system
  - Offline: testing a rom.bin or other file (NVRAM UEFI variables, ACPI tables, etc.), earlier generated by live tools.

Core Live Tools by OS

- Linux: CHIPSEC, acpidump, FWTS, FlashROM, Google Pawn, ls(hw,pci,usb), ...
- macOS: CHIPSEC, acpidump, Apple eficheck[1], DarwinDumper (FlashROM), ...
- Windows: CHIPSEC, acpidump, WinFlashROM, RWEverything[1], ...
- UEFI Shell: CHIPSEC, acpidump (2), multiple shell commands, RU.efi[1], many built-in shell commands, ...
- FreeBSD: FlashROM, acpidump

[1] closed-source freeware, other tools are open source.

Core Offline Tools

- CHIPSEC
- ACPICA (ACPI trade group) tools (acpidump, etc.)
- FWTS
- UEFI Firmware Parser
- UEFITool
- UEFIDump

CHIPSEC

- https://github.com/chipsec/chipsec
- The McAfee (formerly Intel) CHIPSEC Project is a framework for analyzing the security of Intel x86 and x64 systems including hardware, system firmware (BIOS/UEFI), and platform components.
- CHIPSEC does both online analysis of live systems – bare metal and multiple virtualized targets – as well as offline analysis of system images. It includes a security test harness with multiple security modules. It can be run on Windows, Linux, Mac OS X and UEFI shell.
- CHIPSEC_main is a set of security tests, roughly one module per public vulnerability. CHIPSEC_util is a collection of tools – including fuzzers – to explore a system, eg. to dump rom.bin via SPI. Main and Util share a common “HAL” driver, a native OS kernel driver, for accessing various low level interfaces, and forensic capabilities. The Python-based tool also includes a library that other tools can use.
- Attendees: grab a CHIPSEC quick reference sheet before you leave!

ACPIdump

- https://www.acpica.org/source
- The ACPI Component Architecture (ACPICA) project provides an operating system (OS)-independent reference implementation of ACPI. The complexity of the ACPI specification leads to a lengthy and difficult implementation in operating system software. The primary purpose of ACPICA is to simplify ACPI implementations for OSVs by providing major portions of an ACPI implementation in OS-independent ACPI modules that can be integrated into any OS.
- ACPICA includes a handful of ACPI tools, ACPIDump, ACPI Extract, etc.

FWTS (FirmWare Test Suite)

- https://wiki.ubuntu.com/FirmwareTestSuite/Reference
- Firmware Test Suite (FWTS) comprises of over fifty tests that are designed to exercise and test different aspects of a machine's firmware.The tools read UEFI, BIOS, ACPI, IPMI and other platform firmware. FWTS was created by Canonical to help test systems, and works on Ubuntu, and other Linux systems but not BSD or Windows. FWTS has a Linux kernel driver to test UEFI Runtime Services. FWTS has both a command line and a CURSES UI.
- FWTS also has a liveboot Linux distribution called FWTS-live which can be used to run the tests, using the CURSES UI.

FlashROM

- https://www.flashrom.org/Flashrom
- https://github.com/pinczakko/winflashrom
- flashrom is an open source utility for identifying, reading, writing, verifying and erasing flash chips. It is designed to flash BIOS/EFI/coreboot/firmware/optionROM images on mainboards, network/graphics/storage controller cards, and various other programmer devices. It supports parallel, LPC, FWH and SPI flash interfaces and various chip packages. It works on most Unix-like systems, and there is a Windows port.

Apple eficheck

- https://apple.com/
- Recent versions of Apple macOS High Sierra have a new utility called ‘eficheck’. It can dump your system rom into a file (eg, a rom.bin) as well as perform some security analysis, including performing weekly scans of the rom.
- If there’s a problem with firmware, eficheck can opt-into uploading the image to apple.com for them to analyze.

UEFITool (and UEFIDump)

- https://github.com/LongSoft/UEFITool
- UEFITool is a powerful cross-platform C++/Qt program for parsing, extracting and modifying UEFI firmware images. It supports parsing of full BIOS images starting with the flash descriptor or any binary files containing UEFI volumes.
- UEFITool is a Qt GUI tool, but the project also includes a few Qt-free C++ command line tools, UEFIDump, UEFIExtract, and UEFIPatch. The main parsing engine and most of the command line tools are not Qt-dependent. (UEFITools' 'UEFIDump' is like a non-GUI version of UEFITool, and is different from FWTS's 'uefidump'.)
- For an example of using UEFITool, look at Intel Security's Advanced Threat Research team's blog post with analysis of the Hacking Team's UEFI malware, they use CHIPSEC and UEFITool to analyze it.

UEFI Firmware Parser

- https://github.com/theopolis/uefi-firmware-parser
- UEFI Firmware Parser is a Python module and set of scripts for parsing, extracting, and recreating UEFI firmware volumes.
- This includes parsing modules for BIOS, OptionROM, Intel ME and other formats too. It supports: UEFI Firmware Volumes, Capsules, FileSystems, Files, Sections parsing, Intel PCH Flash Descriptors, Intel ME modules parsing (ME, TXE, etc), Dell PFS (HDR) updates parsing, Tiano/EFI, and native LZMA (7z) [de]compression, Complete UEFI Firmware volume object hierarchy display, Firmware descriptor [re]generation using the parsed input volumes, and Firmware File Section injection.

DarwinDumper

- https://bitbucket.org/blackosx/darwindumper
- DarwinDumper is collection of open source tools to dump Apple Mac OS X system information to aid troubleshooting. It dumps ACPI tables, DMI, EFI memory, EFI variables, SMC keys, system BIOS, etc. It has a privacy mode which omits some serial numbers and machine-unique data from the resulting report.
- Tools include: bdmesg, cmosDumperForOsx, dmidecode, dumpACPI, edid-decode, fdisk440, FirmwareMemoryMap, ***flashrom***, getcodecid, genconfig, getdump, gfxutil, iasl, ioregwv, lzma, nvram, oclinfo, lspci, RadeonDump, radeon_bios_decode, smbios-reader, SMC_util, VoodooHDA.kext, x86info.

EFIgy

- https://efigy.io/
- Duo Security just released EFIgy yesterday!
- EFIgy checks the EFI firmware of Apple Mac systems.

UEFI Shell-based tools

- The UEFI Shell has nearly a hundred commands, many powerful diagnostics for low-level firmware/hardware information. No other OS in your way, either.
  - Fairly easy to build a USB thumb drive to boot a system into the UEFI Shell, instead of it’s default OS.
- Sample of some of built-in commands:
  - Bcfg, Dblk, Devices, DevTree, DH, Disconnect, DMem, DmpStore, DP, Drivers, DrvCfg, DrvDiag, EfiCompress, EfiDecompress, GetMTC, IfConfig, Load, LoadPCIROM, Map, MemMap, MM, OpenInfo, Parse, PCI, Reconnect, Reset, SMBIOSView, ..
- Samples of some external third-party commands:
  - CPython 2.7x, CHIPSEC, acpidump, FPMurphy’s UEFI Utilities, UefiToolPkg, Vim, Intel EFI Disk Utilities, ...

Read and Write Everything

- http://rweverything.com/
- RW, aka RWEverything (Read and Write Everything) is a GUI Windows-based firmware utility that enables access to almost all the computer hardware, including PCI (PCI Express), PCI Index/Data, Memory, Memory Index/Data, I/O Space, I/O Index/Data, Super I/O, Clock Generator, DIMM SPD, SMBus Device, CPU MSR Registers, ATA/ATAPI Identify Data, Disk Read Write, ACPI Tables Dump (include AML decode), Embedded Controller, USB Information, SMBIOS Structures, PCI Option ROMs, MP Configuration Table, E820, EDID and Remote Access.
- It ships with Win32 or Win64 binaries, and is freeware, not open source.

Read Universal utility

- http://ruexe.blogspot.tw/
- The Read Universal utility is a multi-function tool for BIOS debugging. It includes tools that provides direct access to almost all resources like memory, IO space, PCI, SMBIOS data, UEFI variables and so on.
- The tool is freeware -- not open source - and is written by a UEFI Engineer at AMI. It ships as ru.exe and ru.efi binaries.

Linux UEFI Validation (LUV)

- https://01.org/linux-uefi-validation
- Aka: LUV, LUVos, luvOS, LUV-live, ‘the LUV shack’.
- LUV is a UEFI test-centric Linux distribution from Intel that helps test UEFI implementation issues for Linux. It bundles multiple external tests (FWTS, CHIPSEC, BITS, etc.), runs them all in batch-mode, saves results for later review. LUV is based on Yocto Linux, and works on Intel x86 and x64 systems.
- LUV provides integrated testing, such as the interaction between the bootloader, Linux kernel and firmware, that require cooperation across multiple runtime phases. Enables LUV to test UEFI capsule updates work correctly across a reboot.
- LUV-live is a LUV-based liveboot distribution. It boots via a thumb drive or via PXE, and runs in batch mode and gathers up test results.

Linux UEFI Validation (LUV)

- https://01.org/linux-uefi-validation
- Aka: LUV, LUVos, luvOS, LUV-live, ‘the LUV shack’.
- LUV is a UEFI test-centric Linux distribution from Intel that helps test UEFI implementation issues for Linux. It bundles multiple external tests (FWTS, CHIPSEC, BITS, etc.), runs them all in batch-mode, saves results for later review. LUV is based on Yocto Linux, and works on Intel x86 and x64 systems.
- LUV provides integrated testing, such as the interaction between the bootloader, Linux kernel and firmware, that require cooperation across multiple runtime phases. Enables LUV to test UEFI capsule updates work correctly across a reboot.
- LUV-live is a LUV-based liveboot distribution. It boots via a thumb drive or via PXE, and runs in batch mode and gathers up test results.

Vendor-Specific Tools

- Many OEMs/IBVs add diagnostic tools.
  - OEMs provides tools for business-class systems, and rely on IBV for tools for ODM-based consumer-class systems.
- Examples:
  - Multiple (HP, Dell, Cisco, etc.) have UEFI diagnostic tools on their boot menus.
  - Microsoft’s Surface has optional enterprise-only firmware, with security features unavailable in consumer firmware.
  - Lenovo’s TPM reset CD and UEFI diagnostic CD.
  - Apple Store’s Geniuses have access to a tool that can reset your firmware password.

PreOS vaporware announcement

- PreOS Security is creating a new open source firmware tool for SysAdmins, SRE, DFIR, Blue Teams, and advanced users.
- It gathers data about firmware (UEFI, ACPI etc.) via invoking multiple firmware tools, starting with CHIPSEC and FWTS, more tools planned.
- It targets Linux on Intel x64 UEFI systems. More arch/OS targets planned, x86, AArch64, Windows, macOS, more Linux distros.
- For announcement of availability, firmware security tips, sign up at:

https://preossec.com/newsletter

NEXT MODULE

- Threats
- Tech
- Tools
- Guidance

Guidance

- Next few slides have some advice taken from existing sources
- 1) CHIPSEC team interviewed for an article in DarkReading.com, with 5 basic tips for firmware security.
- 2) NIST has 3 documents with ‘secure BIOS’ advice, for enterprises, including a secure BIOS platform lifecycle.
- I’ve added some of the existing tool suggestions for the various phases of the NIST secure BIOS platform lifecycle. Don’t blame NIST for those suggestions! :-)

DarkReading's 5 tips

- Credit to: http://www.darkreading.com/iot/5-tips-for-protecting-firmware-from-attacks/d/d-id/1325604

  1) Know that the threat is real.

  2) Practice security basics: Least privilege, Physical access controls, Disable unnecessary services, firewalls, AntiVirus, Incident Response plans and drills, etc.

Dark Reading (cont'd)

3) Benchmark for vulnerabilities: write protection, secure boot, blacklist, cert revocation

4) Make a firmware golden image, check periodically, particularly after an incident

1) Also do this for test results (eg: CHIPSEC for UEFI)
2) When tests are updated, re-run

5) Think broadly about firmware: presence in many aspects of a system, many types of hardware, automated vs. manual updates.

NIST secure BIOS guidance

- 2011: SP 800-147: BIOS Protection Guidelines
  - Protects systems from unauthorized BIOS modification by defining a secure, non-bypassable authenticated update mechanism.
- 2011: SP 800-155: BIOS Integrity Measurement Guidelines (Draft)
  - Outlines a framework for a secure BIOS integrity measurement and reporting chain for client systems, to detect unauthorized modification of System BIOS and configuration using secure measurement and reporting mechanisms.
- 2014: SP 800-147B: BIOS Protection Guidelines for Servers
  - Extends SP800-147 system model, from simple PC to more sophisticated servers with BMC and OOB update mechanisms.
- 2017: SP 800-193: Platform Firmware Resiliency Guidelines (Draft)
  - Provides technical guidelines and recommendations supporting resiliency of platform firmware and data against potentially destructive attacks.

NIST SP 800-147 Platform Lifecycle

- Guidance is both for device vendors, as well as enterprise sysadmins – but YOU are the (only) sysadmin for your own personal system(s) usually.
- 5 States of firmware lifecycle, covering acquisition to disposition:
  - Provisioning
  - Platform Deployment
  - Operations and Maintenance
  - Recovery
  - Disposition

Pre-Provisioning: Before Purchase

- Vendor research
  - RFP, request CHIPSEC logs
  - Don’t buy the system if it fails CHIPSEC’s security tests
  - Goal is to avoid purchasing insecure systems.
- Set company purchase policy about what HW/FW features must be in new models.
  - Require new systems to have features listed in NIST 147/155/etc to provide firmware security
  - Eg, All new Intel UEFI systems must pass all current relevant CHIPSEC security tests.
  - Eg, UEFI instead of BIOS, must have TPM v2, etc.

Provisioning: Pre/+Deployment

- Test CHIPSEC, confirm logs did pass, if not return the system!
- Follow your policy:
  - Using/disabling silicon security tech (VT, TPM, TXT, TZ, TEE, Intel-ME, …)
  - Using/disabling ports/hardware (WiFi, USB, Thunderbolt, BT, Ethernet, …)
  - Smartcard authentication for boot/network access Secure/Trusted/Measured/Verified Boot
  - Clarify system boot order
  - WakeOn<N> features
  - No unencrypted storage (Self-Encrypting Drives, SEDs, NVMe + TPM)
  - Firmware/BIOS password
  - Selecting UEFI over CSM/LegacyBoot
  - Limit ‘pre-OS’ software installed (UEFI drivers, services, apps, UEFI shell scripts)
  - Check ESP, look for startup.nsh (unlike autoexec.bat, there can be N of these, in any directory, if in UEFI Shell's %PATH%)
  - Look for *.efi, look on OEM/IHV flash for OpROMs and UEFI drivers
  - Control/disable net access (IPMI, PXE, HTTP Boot, SMASH and DASH, Redfish, …)
- Enable Secure Boot!
  - Enable signed code: All vendor & 3rd party code should be signed!
  - Some sites may wish to use their own keys, instead of default UEFI CA
  - Ensure DBX file is up to date
- Maintain “golden” image for each platform, all firmwares and state
  - Compare golden images, ACPI tables, UEFI variables, compare image with vendor-supplied image or hash, save archive
  - Create and maintain configuration baseline
  - Maintain a copy of the RTU, if applicable
  - Register endpoint identity and BIOS integrity information in system inventory

Operations: Maintenance & Monitoring

- Update! OS standardized and NIST SP 800-147 authenticated if possible eg:
  - Windows via Windows Update (from Win 7 onward)
  - Linux via fwupd
  - Mac via App Store updates
  - Vendor specific (eg: Eve's USB ethernet adapter firmware update v.0.4 → v0.5)
- Measure!after firmware updates – dump images, ACPI tables, UEFI variables.
- Periodic/Regular firmware scans: (eg: re-run CHIPSEC, dump ACPI tables, UEFI vars)
  - After updates
  - When new tests are available
- Network – isolate, authenticate, encrypt and monitor all firmware network traffic
  - UEFI HTTP[S] Boot, IPsec, iSCSI, PXE, AMT, DASH, SMASH, IPMI, iLO, etc.

Recovery

- After a security incident...or suspected incident including BUT NOT LIMITED TO:
  - Loss of physical control of machine
  - Known Evil Maid Attack
  - OS level compromise
- Take full validation steps:
  - regenerate image
  - rerun tests
  - compare image, ACPI tables, UEFI variables & test results with originals, looking for signs of infection. eg,SPI protections were enabled, now disabled
  - Use forensic tools eg: chipsec_util.py to parse SPI image

Disposition

- Reset BIOS/UEFI configuration to defaults
- Remove passwords and organization-specific cryptographic keys
- Remove organization-specific customizations
- Reset any TPM secrets
- Sanitize media
- UEFI has user data, User ID and Smartcard drivers, for PXE remote boot and IPsec use (CHAP auth used somewhere).
- Be clear on what user credentials are stored in your firmware, before you recycle it!
- Ask your vendor how to purge PII from their firmware product, for safe disposition.
  - Eg, Lenovo has a CD that resets the TPM data.

Summary

- Understand the firmware security problem, start with NIST SP 800-147’s lifecycle.
- Learn to use the tools to help you protect your firmware, start with CHIPSEC.
- PreOS Security has an upcoming ebook that covers this topic.
- PreOS Security has an upcoming firmware test tool to help you diagnose your firmware.

Credits

- Thanks to my business partner & CTO of PreOS Security, Lee Fisher for developing this talk.
- Thanks to many people on the firmware-security list on Twitter, the edk2-devel mailing list, from the CHIPSEC and FWTS projects for answering questions. Thanks to many other security researchers for their tools and firmware research.

Questions?

- Questions?
- Comments?
- Thanks for attending!
- Free e-book, quarterly newsletter, software announcements:

  https://preossec.com/newsletter

- Paul English
  penglish@preossec.com

  @penglish_preos

  https://preossec.com
  https://firmwaresecurity.com (personal blog of Lee Fisher, CTO of PreOS Security)