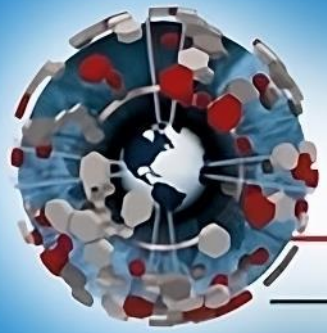


Keynote:
The story of UEFI (and its security mitigations)

H2H 2023 Sao Paulo

Vincent Zimmer

#WhoAmI

Vincent Zimmer is a senior principal engineer at Intel. He has been engaged w/ firmware for over 30 years and presently leads the UEFI Security sub team in the UEFI Forum, EDKII infosec and is Intel's Technical Committee representative in the Trusted Computing Group.
Vincent lives in the Seattle area.

Vincent Zimmer
Intel

Agenda

- Background on UEFI
- Security Features
- Mitigations
- Open resources
- Futures

Old Days

19XX

Pioneer

CP/M

BIOS
(machine specific CP/M)

8080/Z80

1974 Basic I/O (Sub) System
by Gary Kildall in CP/M

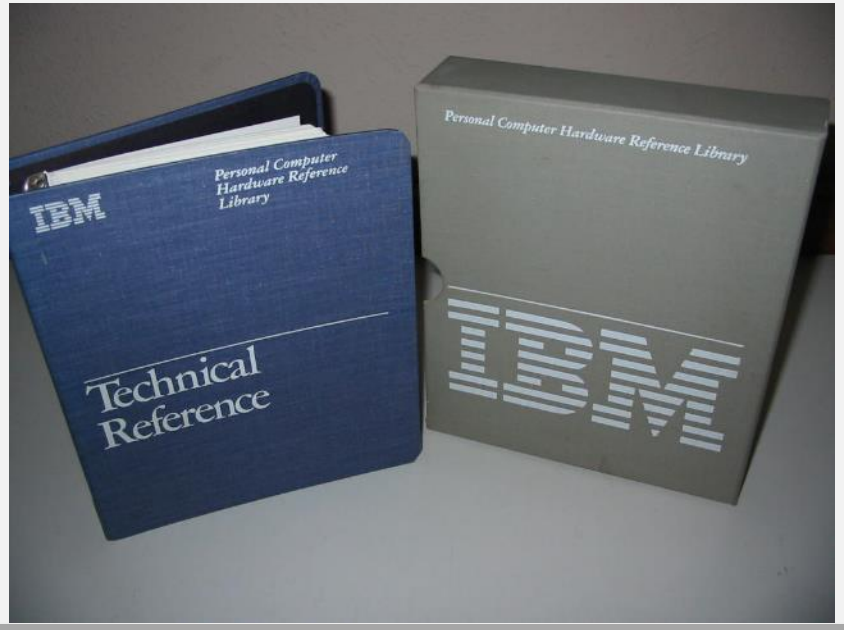
PC/AT BIOS

DOS

BIOS
(de facto standard)

8088

1981 IBM PC



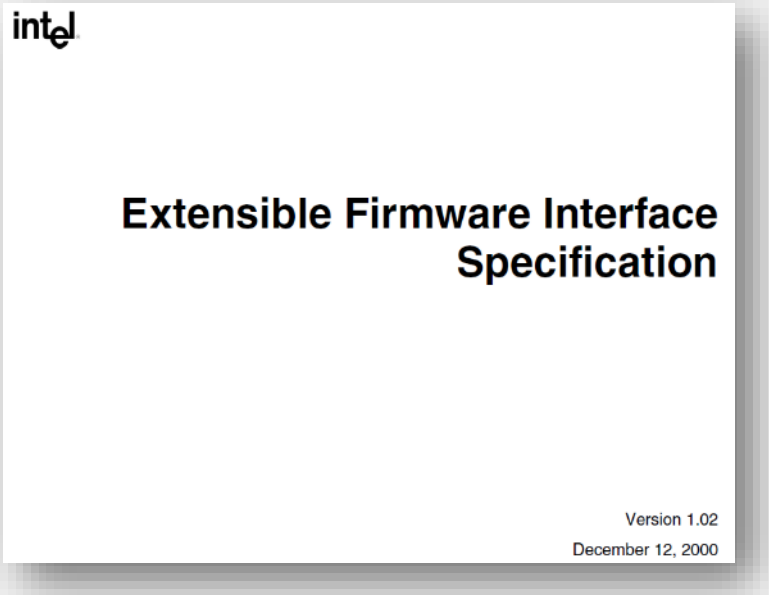
PC/AT BIOS -> EFI

IPF Windows/Linux

EFI

(Intel Standard)

IPF (Merced)



2000 Extensible Firmware Interface

Intel/HP IPF

Broader adoption

Windows/Linux

UEFI

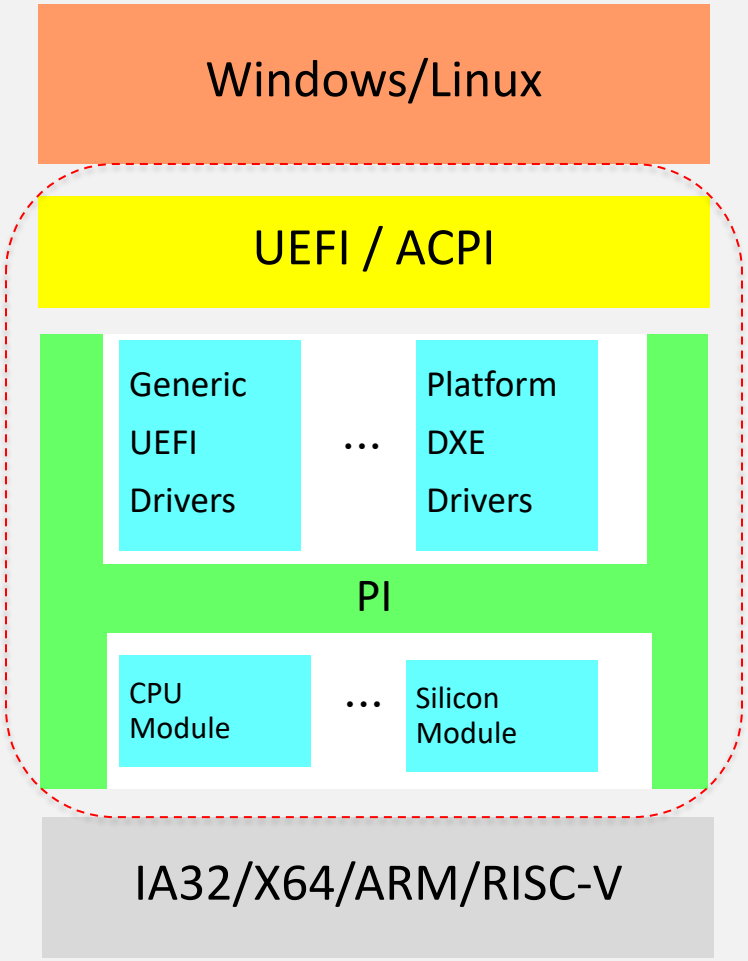
(Industry Standard)

IA32/X64/IPF/ARM

2006 January

Unified Extensible Firmware Interface

Today



UEFI / ACPI + PI

2006 Aug Platform Initialization

UEFI Background

UEFI – Unified Extensible Firmware Interface old school terms…(BIOS)

First up root of trust on the system

It hands over control to the operating system

- Rest of the magic then occurs ;)

UEFI Membership spans the compute spectrum

http://uefi.org/members

UEFI – sets up the platform to run…

Industry Transition

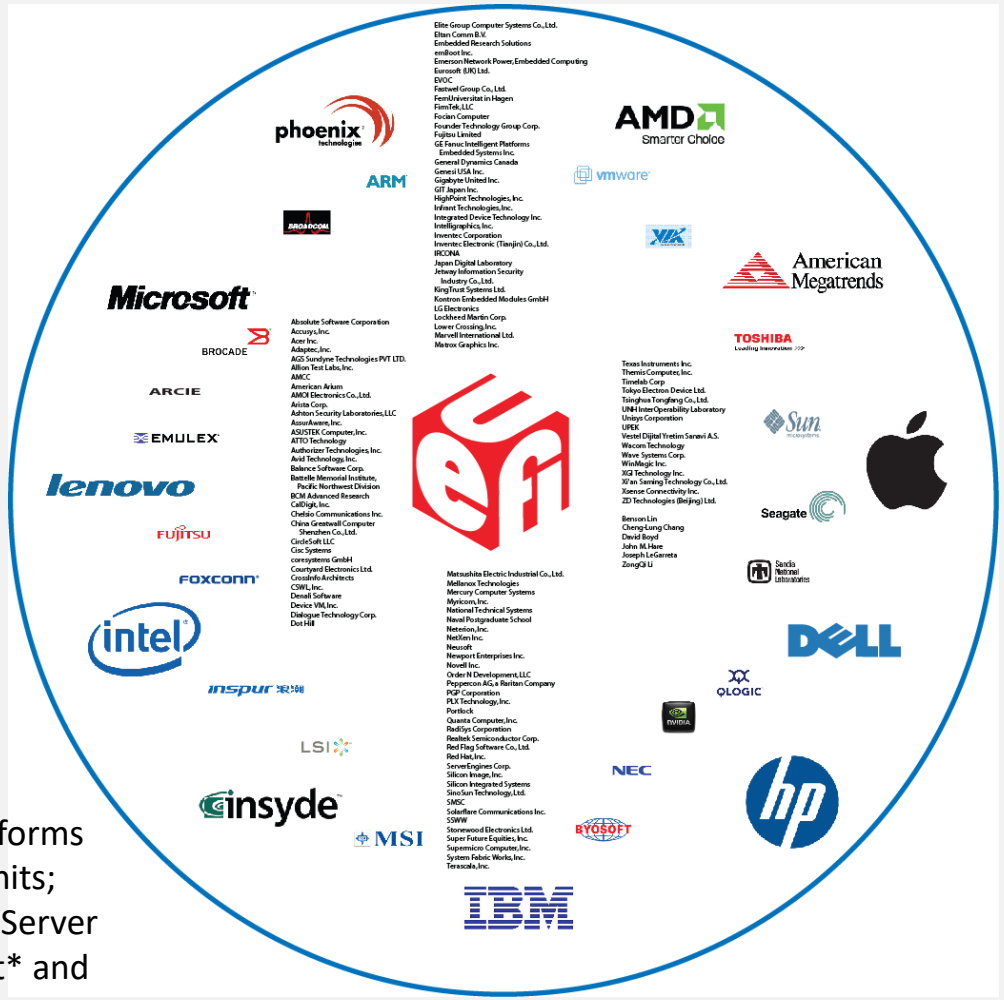
Pre-2000 All Platforms BIOS were proprietary

2000 Intel invented the Extensible Firmware Interface (EFI) and provided sample implementation under free BSD terms

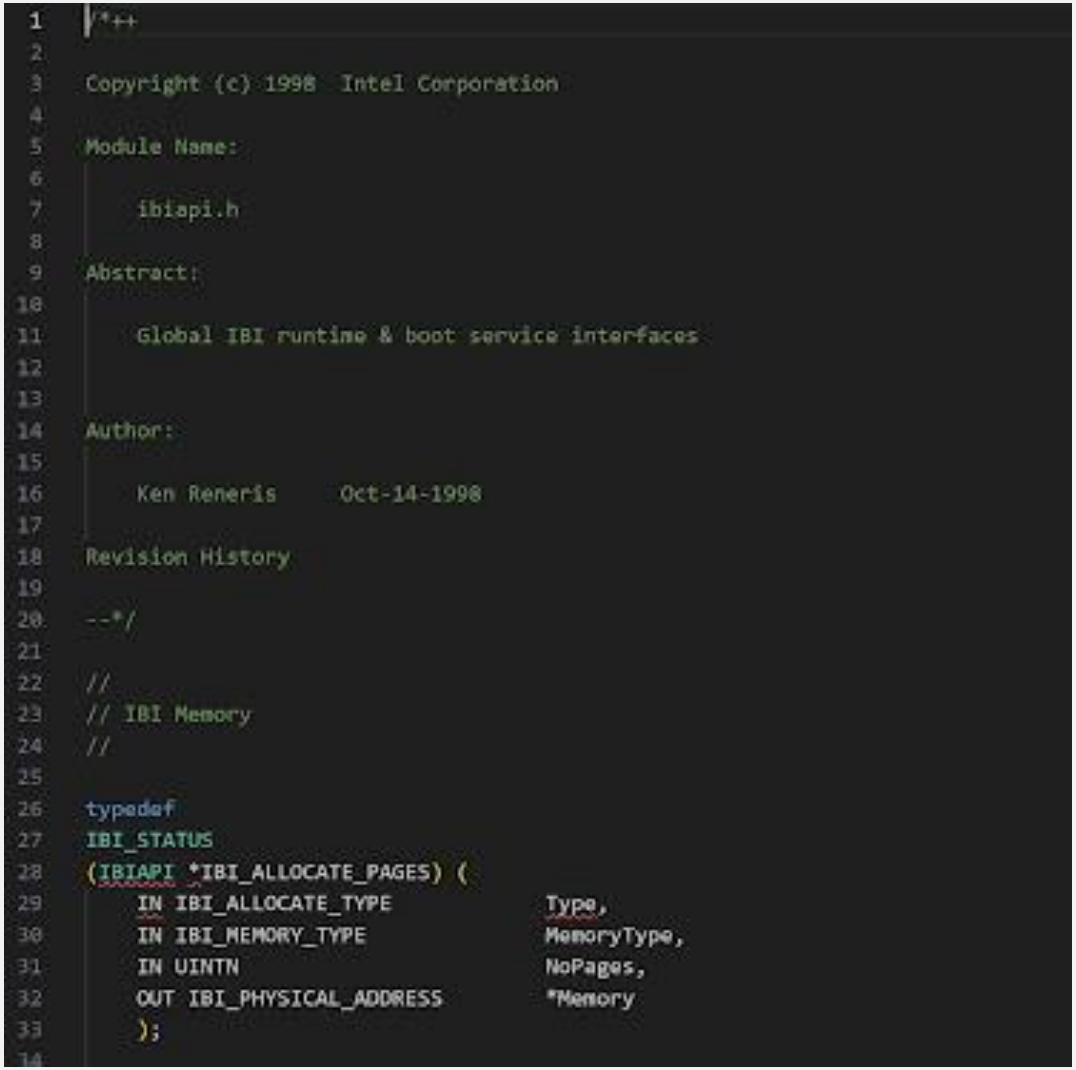
2004 **tianocore.org**, open source EFI community launched

2005 **Unified EFI (UEFI)** Industry forum, with 11 members, was formed to standardize EFI

2023 240 members and growing! Major MNCs shipping; UEFI platforms crossed most of IA worldwide units; Microsoft* UEFI x64 support in Server 2008, Vista* and Win7*; RedHat* and SuSEI* OS support. Mandatory for Windows 8 client. ARM 32 and 64 bit support. ACPI added. RISC-V added

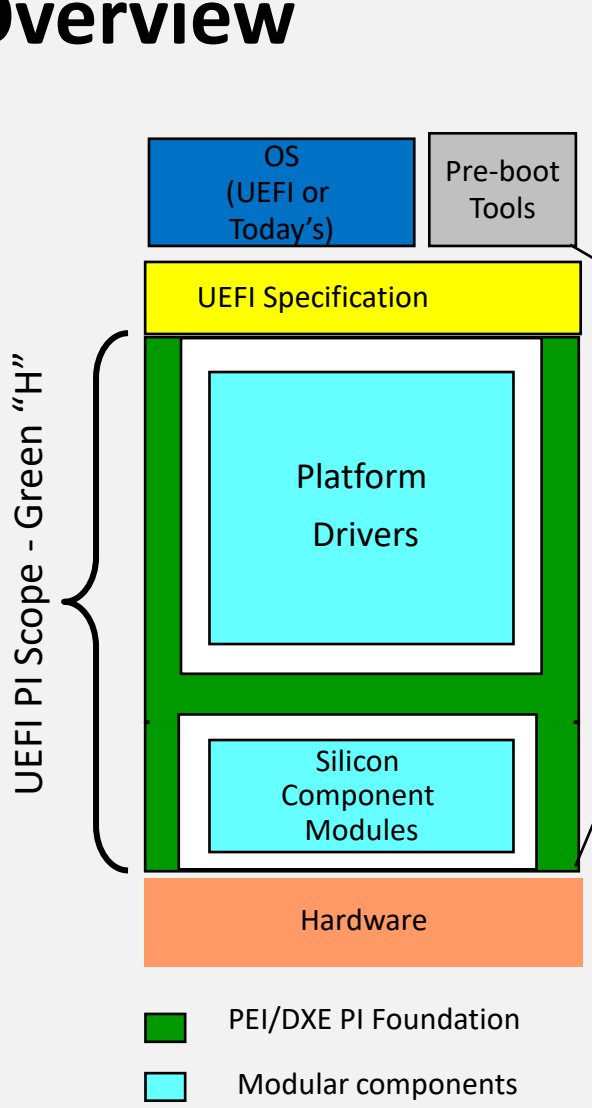


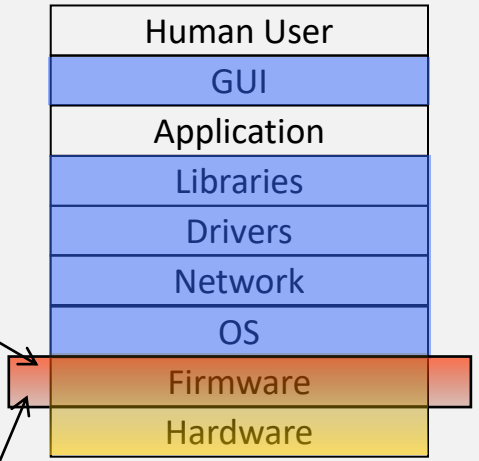
Happy birthday – 25 years of ‘EFI’ now



```
/*++

Copyright (c) 1998  Intel Corporation

Module Name:

    ibiapi.h

Abstract:

    Global IBI runtime & boot service interfaces


Author:

    Ken Reneris     Oct-14-1998

Revision History

--*/

//
// IBI Memory
//

typedef
IBI_STATUS
(IBIAPI *IBI_ALLOCATE_PAGES) (
    IN IBI_ALLOCATE_TYPE            Type,
    IN IBI_MEMORY_TYPE              MemoryType,
    IN UINTN                        NoPages,
    OUT IBI_PHYSICAL_ADDRESS        *Memory
    );
```

What is UEFI? UEFI Platform Initialization Overview





Full system stack (user -> hardware)

UEFI 2.10 specifies how firmware boots OS loader

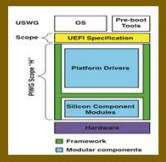
UEFI's Platform Initialization (PI) 1.8 Architecture specifies how Driver Execution Environment (DXE) Drivers and Pre-EFI Initialization (PEI) Modules (PEIMs) initialize SI and the platform

DXE is preferred UEFI Implementation

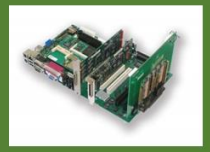
PEIMs, UEFI and DXE drivers implement networking, Update, other security features

What's in UEFI

Mostly written in C.
High code re-use.



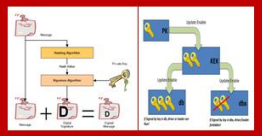
Emphasis on Specifications.
Standards compliance.



Better platform scaling. For e.g. removes shadow ROM limits.

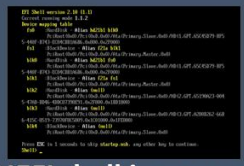
Storage.
GPT removes 2.2 TB MBR restriction.

CPU Architecture independent. Platform design flexibility.

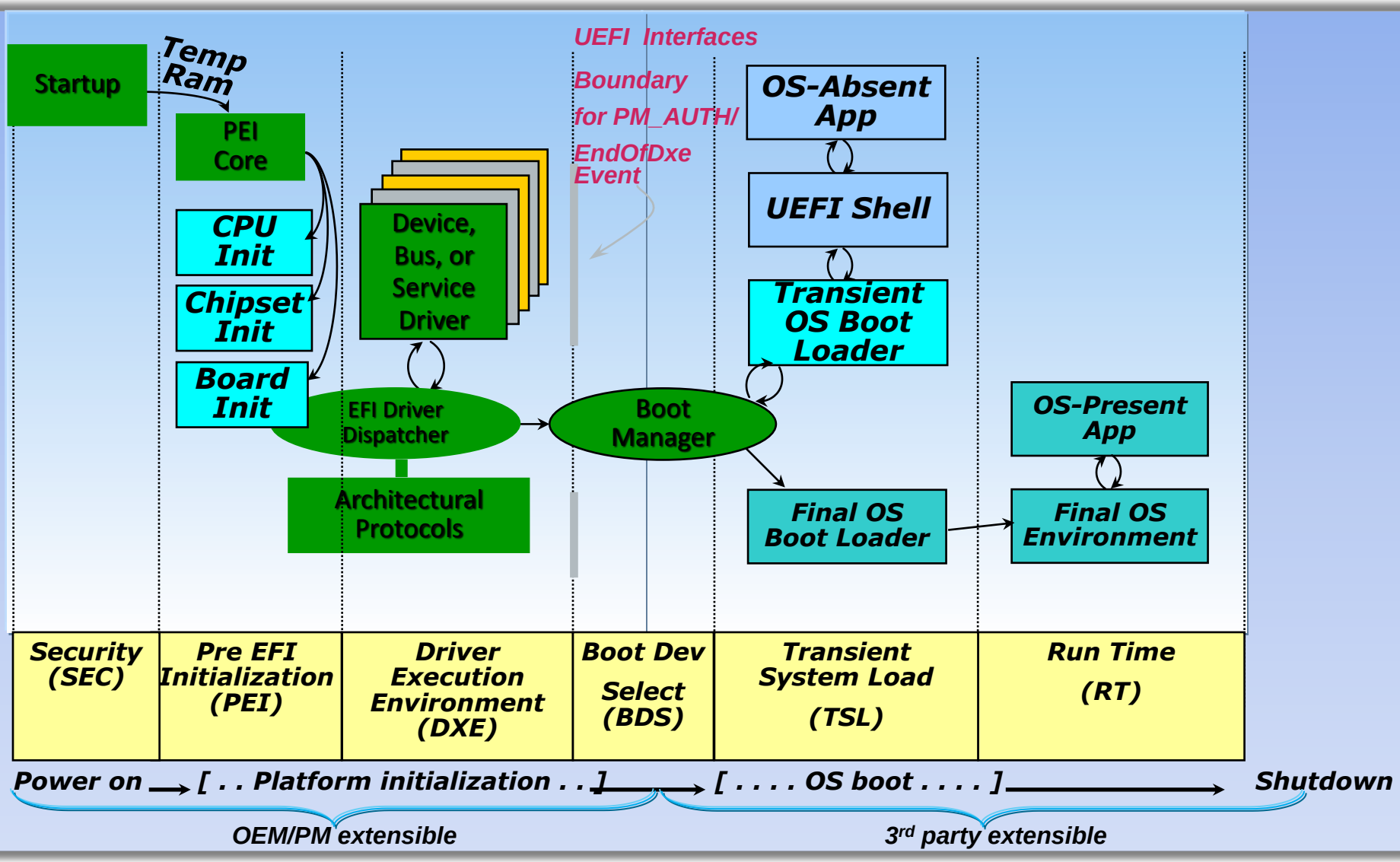


Secure boot solves "trust" related system integration challenges.

Pre-boot Networking.
Ipv4, Ipv6, PXE,
VLAN, iSCSI etc.



UEFI shell improves pre-boot testing & diagnostics experience.



Overall UEFI Boot Timeline

How to build it? EDKII

Industry Standards Compliance
- UEFI 2.0, UEFI 2.1, UEFI 2.2, UEFI 2.3, UEFI 2.4; PI 1.0, PI 1.1, PI 1.2, PI1.3, ACPI 5.1

Extensible Foundation for Advanced Capabilities
- Pre-OS Security
- Rich Networking
- Manageability

Support for UEFI Packages
- Import/export modules source/binaries to many build systems

Maximize Re-use of Source Code**
- Platform Configuration Database (PCD) provides “knobs” for binaries
- ECP provides for reuse of EDK1117 (EDK I) modules
- Improved modularity, library classes and instances
- Optimize for size or speed

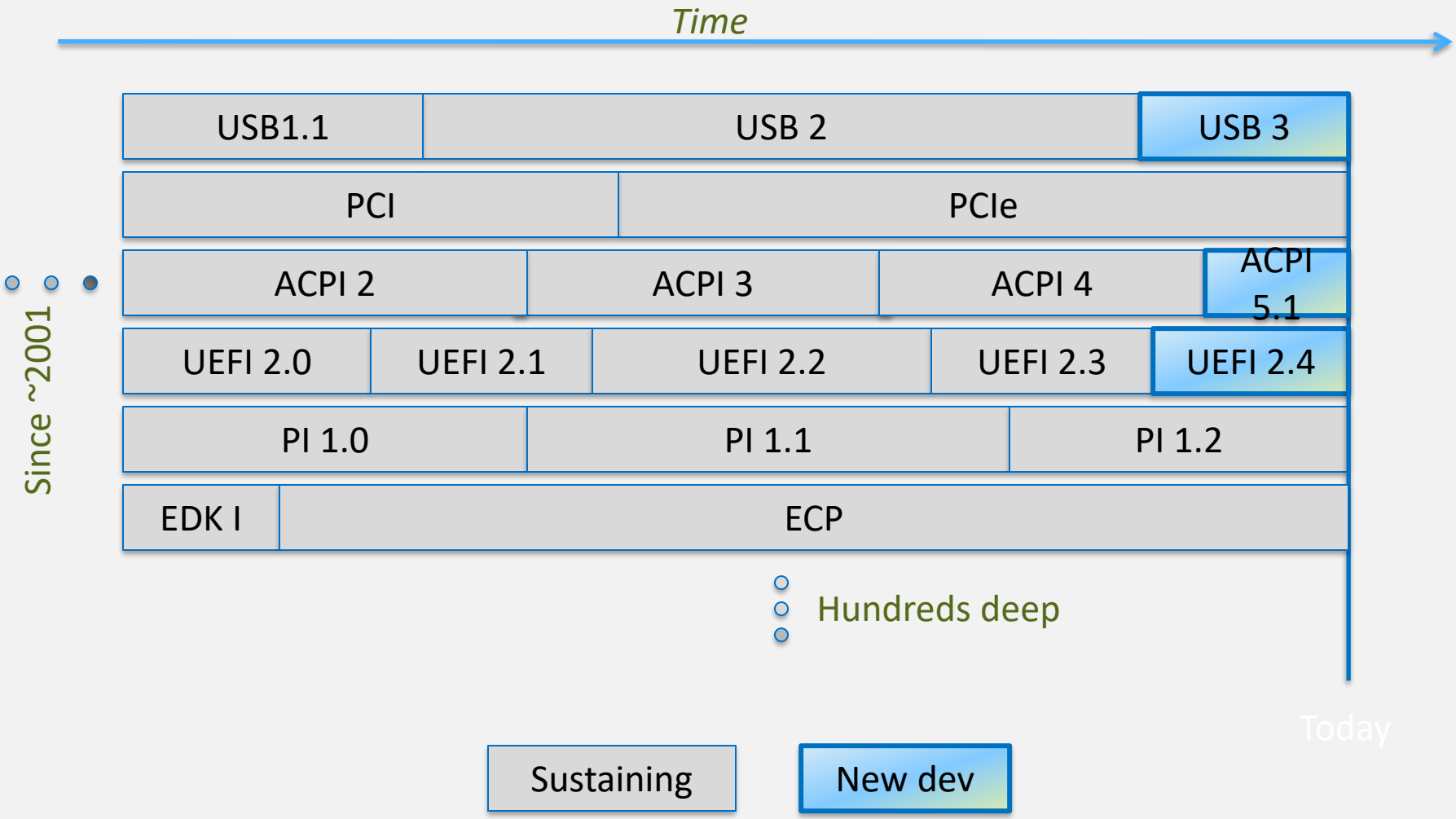
Multiple Development Environments and Tool Chains**
- Windows, Linux, OSX
- VS2003, VS2005, WinDDK, Intel, GCC

Fast and Flexible Build Infrastructure**
- 4X+ Build Performance Improvement (vs EDKI)
- Targeted Module Build Flexibility

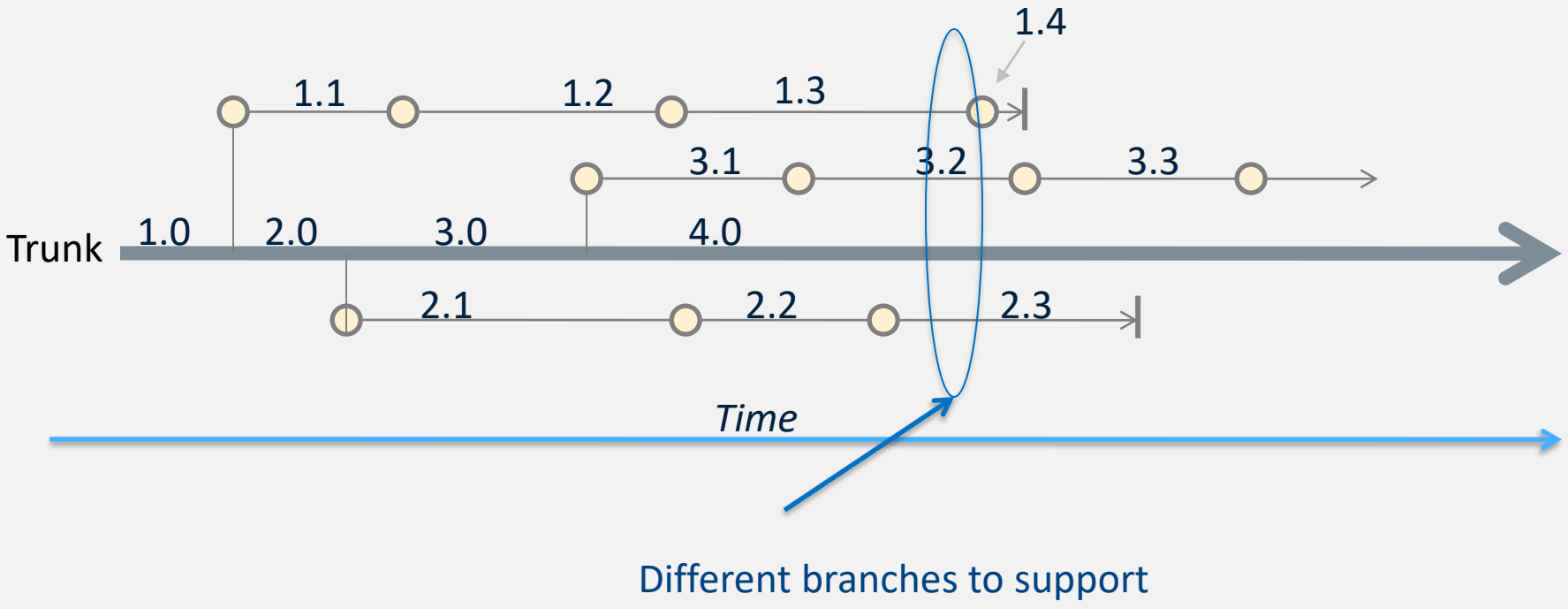
*** benefit of EDK II codebase*

Maximize the open source at www.tianocore.org

Contents

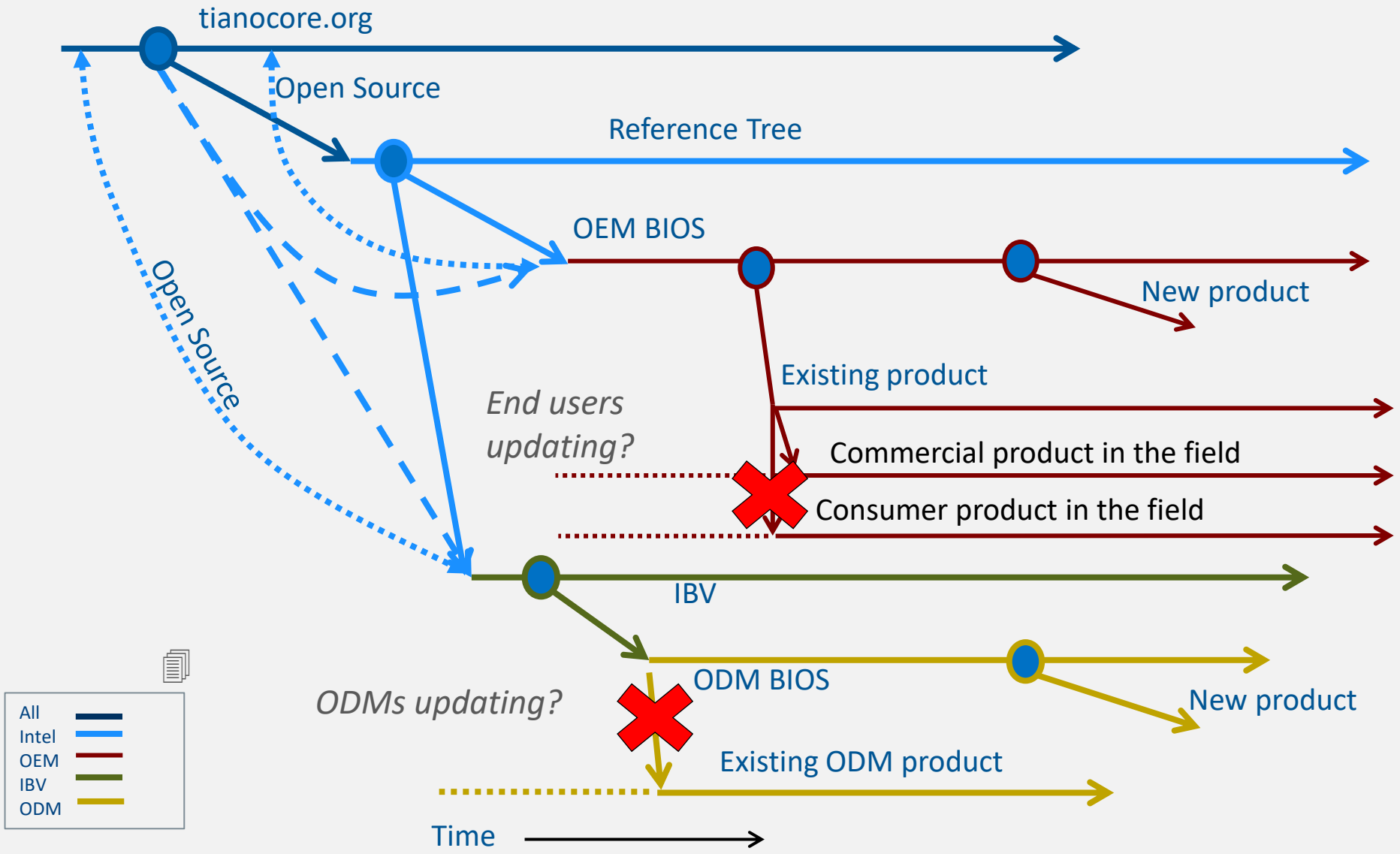

Time
USB1.1
USB 2
USB 3
PCI
PCIe
ACPI 2
ACPI 3
ACPI 4
ACPI 5.1
UEFI 2.0
UEFI 2.1
UEFI 2.2
UEFI 2.3
UEFI 2.4
PI 1.0
PI 1.1
PI 1.2
EDK I
ECP
Since ~2001
Hundreds deep
Today
Sustaining
New dev

Core evolution

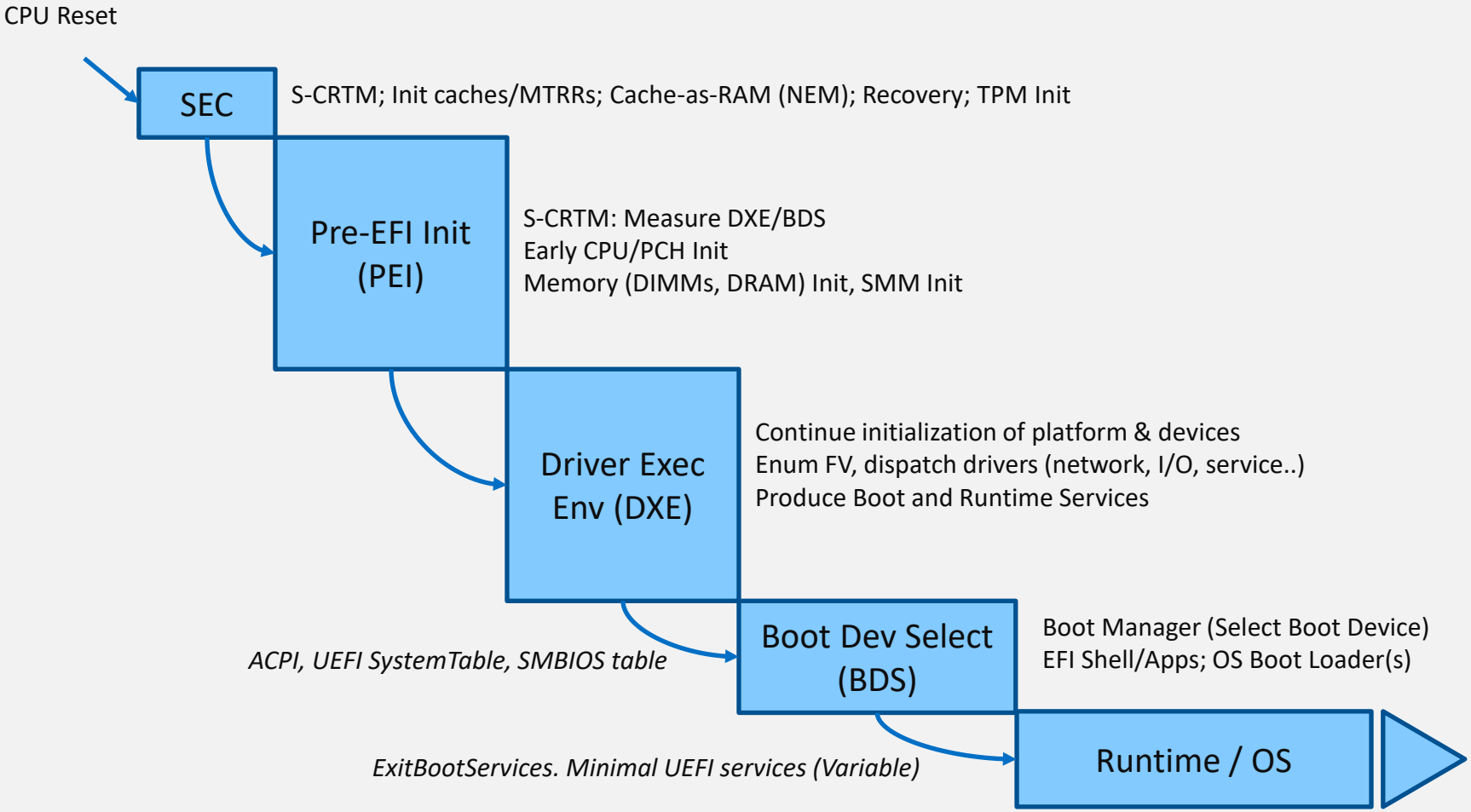


Different branches to support

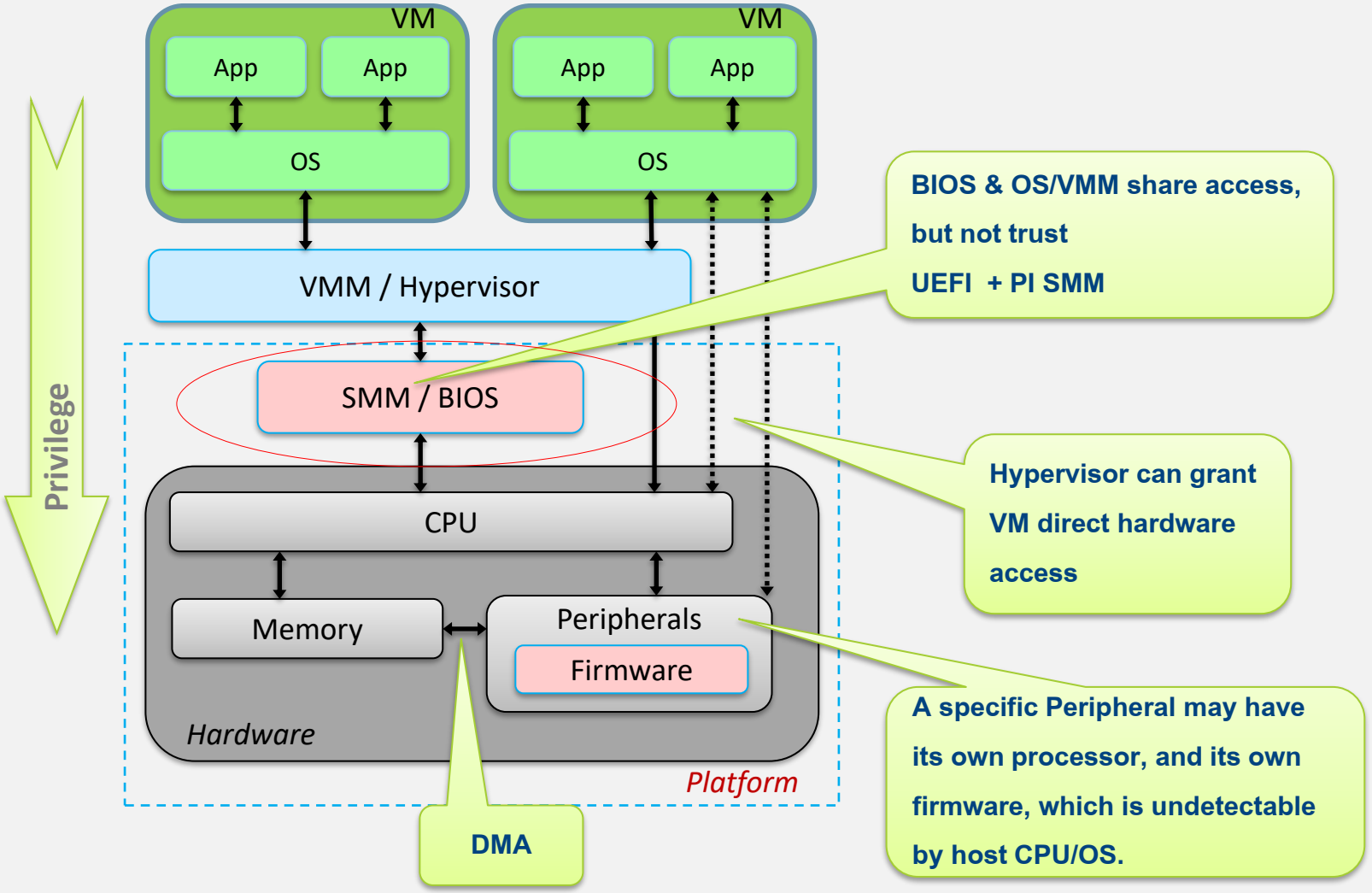
The road from core to platform



UEFI [Compliant] Firmware

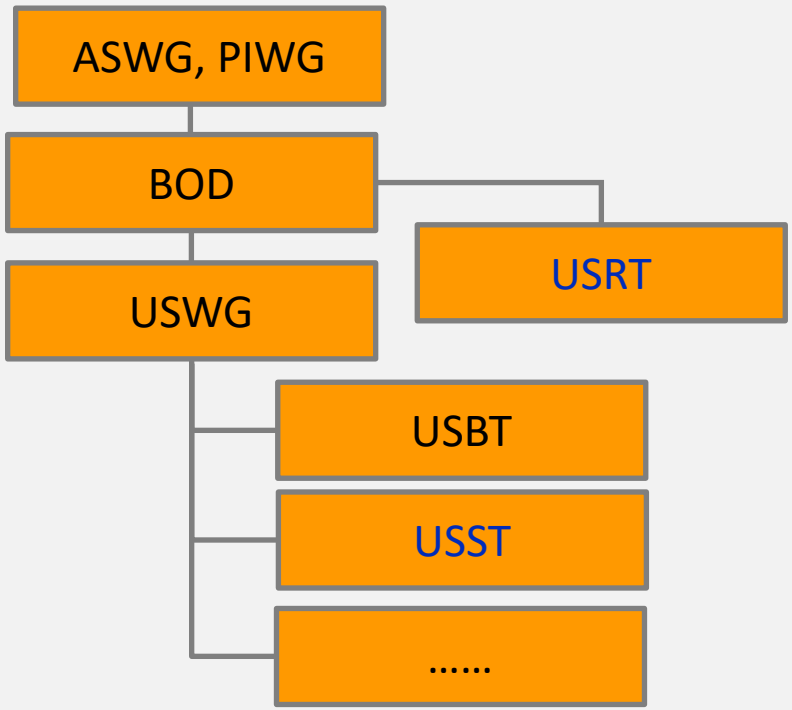


Where are we (UEFI firmware)?



Working Groups in the Forum

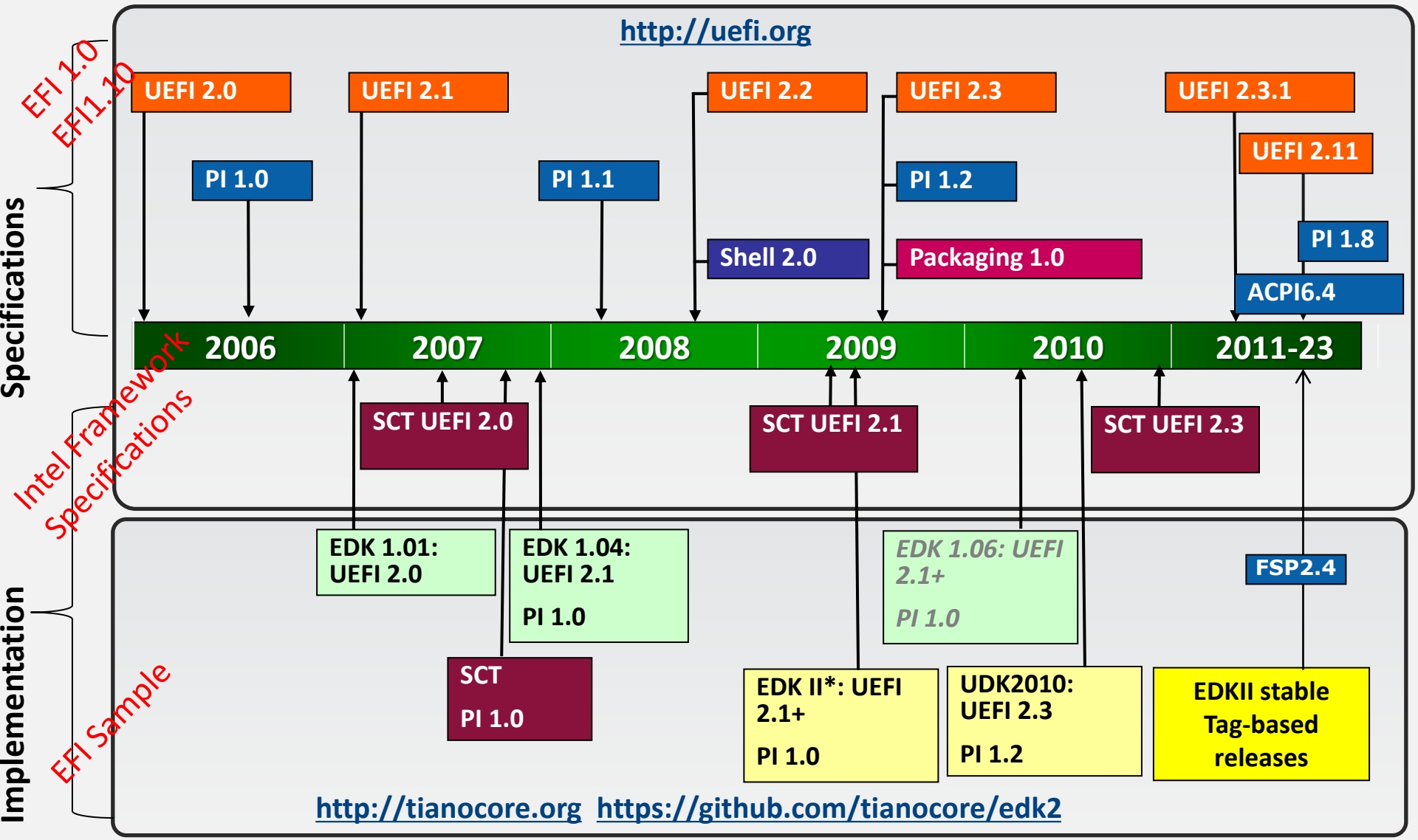
www.uefi.org



- **USWG**
  - **U**EFI **S**pecification **W**orking **G**roup
- **PIWG**
  - **P**latform **I**nitialization **W**orking **G**roup
- **ASWG**
  - **A**CPI **S**pecification **W**orking **G**roup
- **BOD**
  - **B**oard **O**f **D**irectors
- **USST**
  - **U**SWG **S**ecurity **S**ub-**t**eam
  - Chaired by Vincent Zimmer (Intel)
  - Responsible for all security related material and the team that has added security infrastructure in the UEFI spec
- **USRT**
  - UEFI Security Response Team
  - Chaired by Dick Wilkins (Phoenix)
  - Provide response to security issues.
- **USBT**
  - **U**EFI **SB**om-**t**eam
  - Explores SBOM requirements for UEIF

Note: Engaged in firmware/boot
Related WG's of Trusted Computing Group (TCG), IETF, DMTF

Specifications and code



All products, dates, and programs are based on current expectations and subject to change without notice.

UEFI Ecosystem Overview

UEFI Ecosystem is an “Onion”. Layers upon layers…

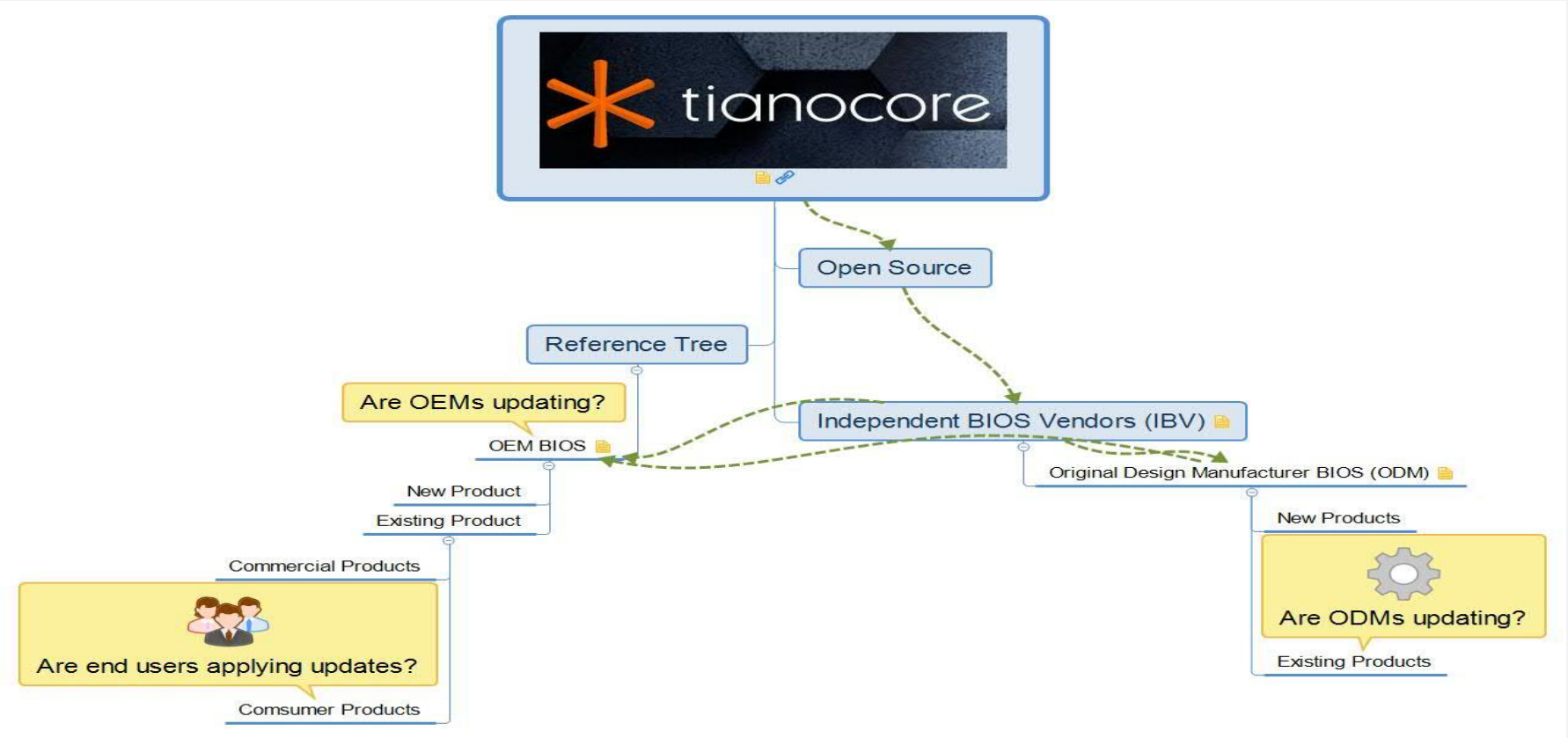
Peel the onion and you have:

- Tianocore (Open Source EDK II) Intel is the sole maintainer
- IBV – Independent BIOS Vendors
- OEM – Other Equipment Manufacturers (The folks building your systems)
- ODM – Original Design Manufacturer
- Consumers (deliberate action to download/install updates)

If a vuln mitigation goes out it has to navigate the onion

- **Additional update lag time is introduced because end users have to take deliberate action to download/install updates**

UEFI Ecosystem



UEFI USRT

USRT is comprised by Firmware Engineers from member companies

To report a security issue in UEFI Firmware implementation from a vendor:

- Send email to security@uefi.org
- Encrypt sensitive info with their PGP public key - security@uefi.org

Please provide as much information as possible, including:

- The products and versions affected
- Detailed description of the vulnerability
- Steps to demonstrate the vulnerability or reproduce the exploit, including specific configurations or peripherals, if relevant
- Potential impact of the vulnerability, when exploited
- Information on known exploits

EDKII Bugzilla

The EFI Developer Kit II (EDKII) provides an open source implementation http://www.tianocore.org/

- Core features https://github.com/tianocore/edk2
- Platform examples https://github.com/tianocore/edk2-platforms

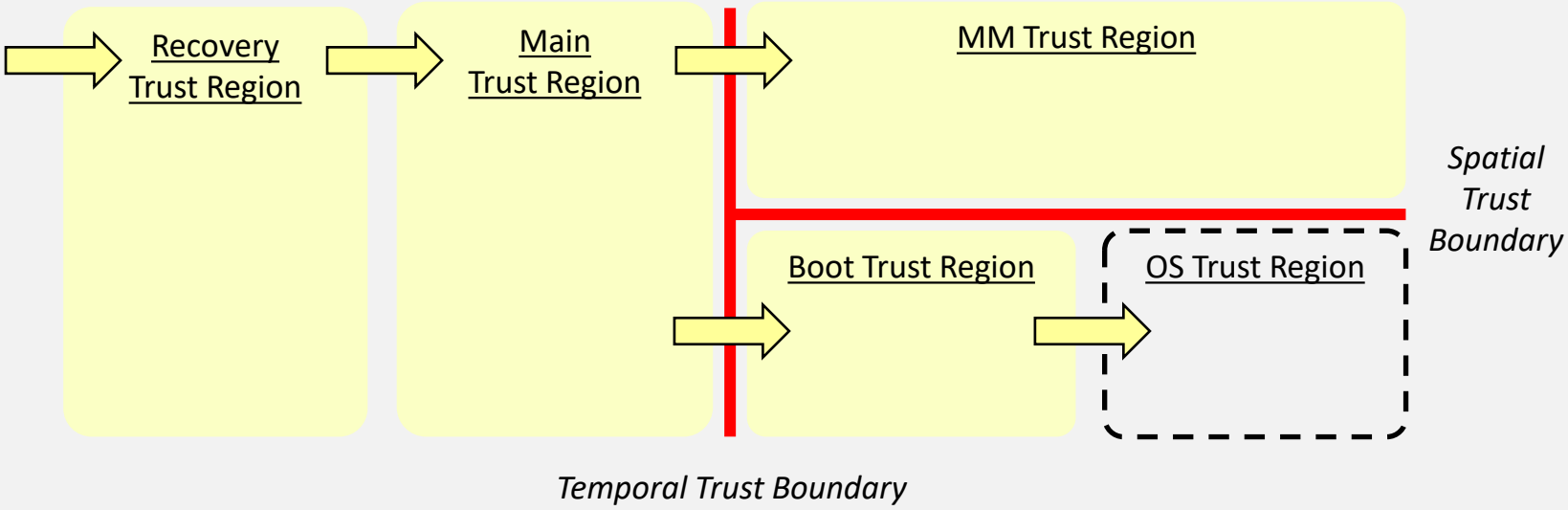
Reporting security issues on open source https://github.com/tianocore/tianocore.github.io/wiki/Reporting-Security-Issues

- Advisories https://www.gitbook.com/book/edk2-docs/security-advisory/details
- Now a CVE Naming Authority (CNA)
- Moving to Github Security Advisory (GHSA) process for code vulnerability and issue management

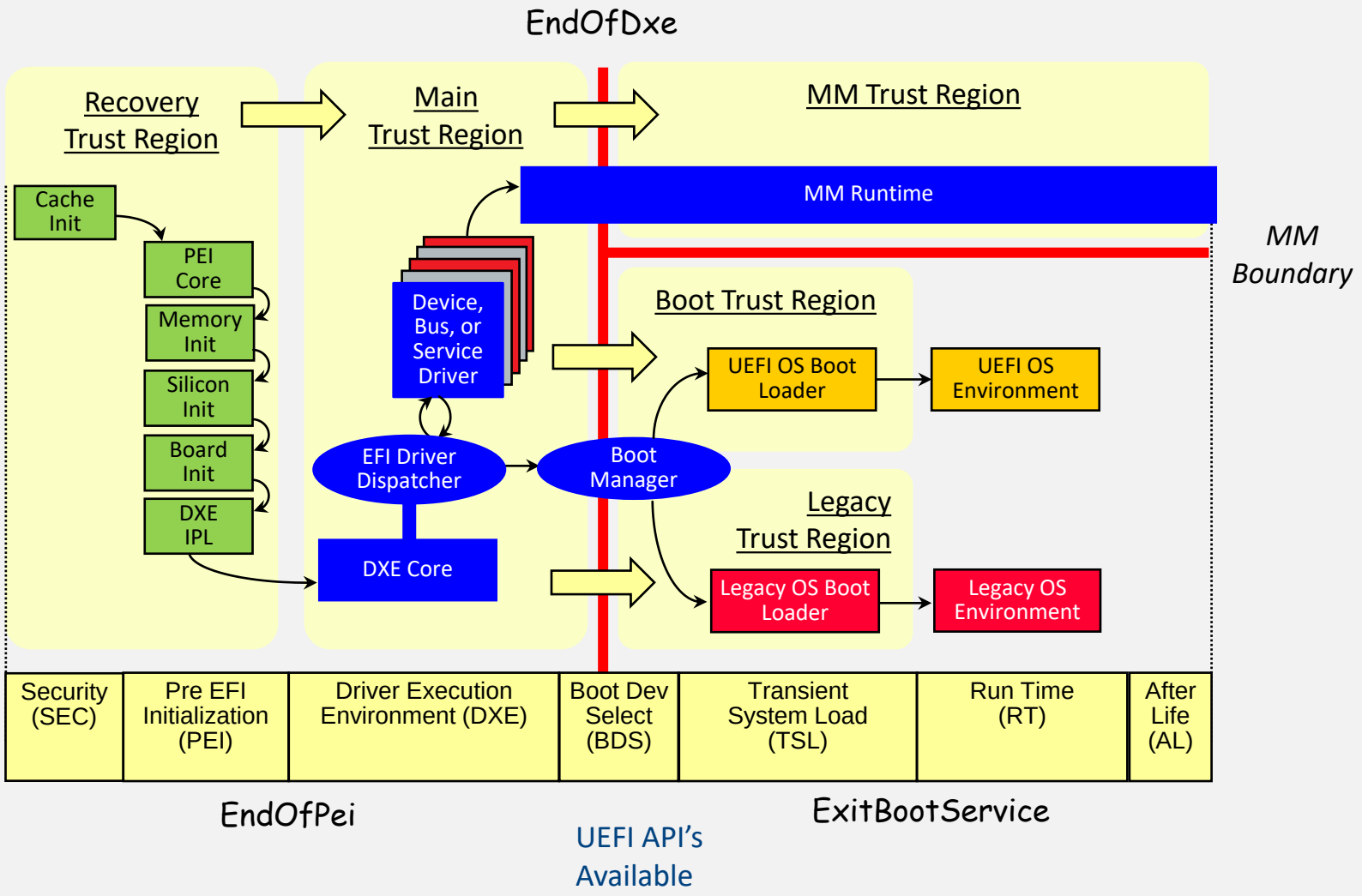
Security Fundamentals

Security Fundamentals

Boot trust boundaries revisited



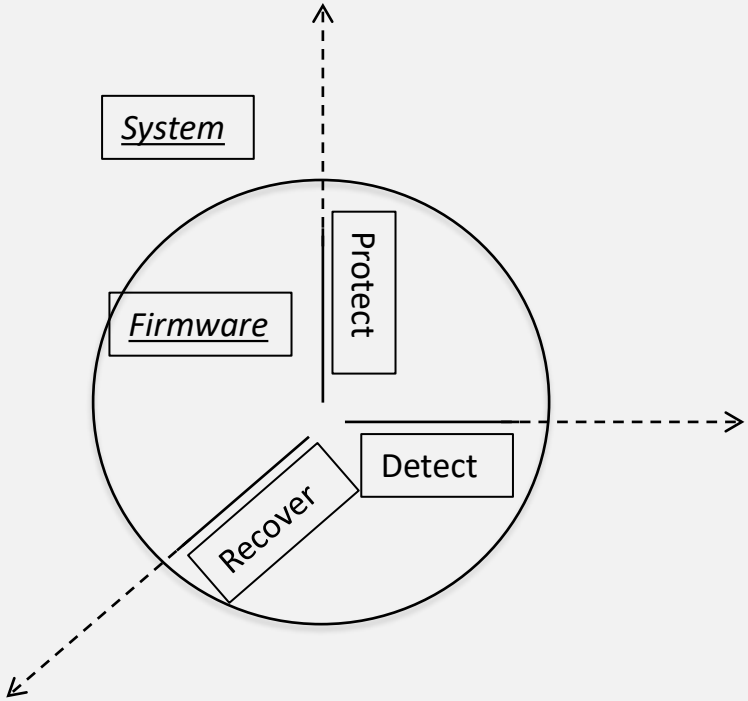
Boot trust boundaries for UEFI PI code



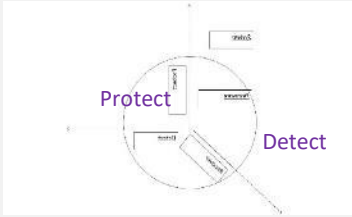
Use of EDKII Defenses

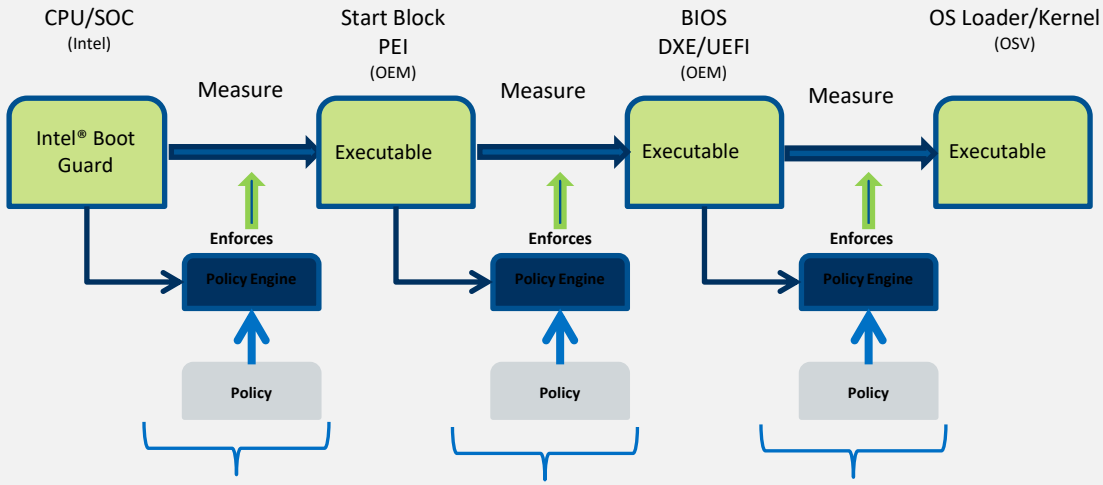
UEFI is the specification

EDK II is the code



Fully Verified 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

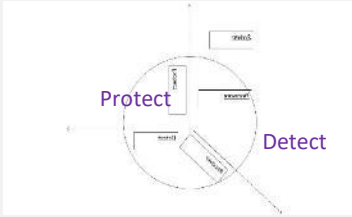
OEM PI Verification Using PI Signed Firmware Volumes

Vol 3, section 3.2.1.1
of PI 1.3 Specification

OEM UEFI 2.10 Secure Boot

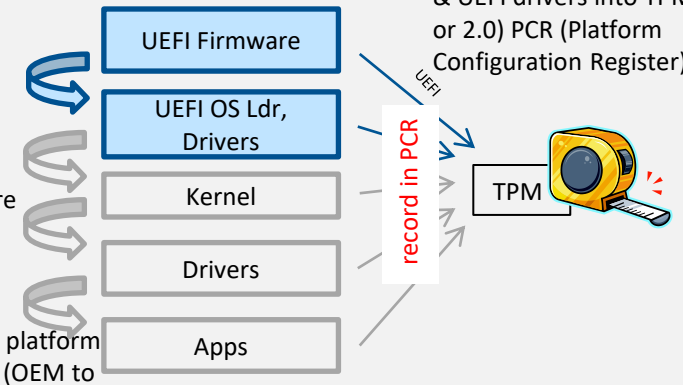
Chapter 27.2 of
The UEFI 2.4
Specification

Trusted Versus Secure Boot



UEFI authenticate OS loader
(pub key and policy)

Check signature of before loading

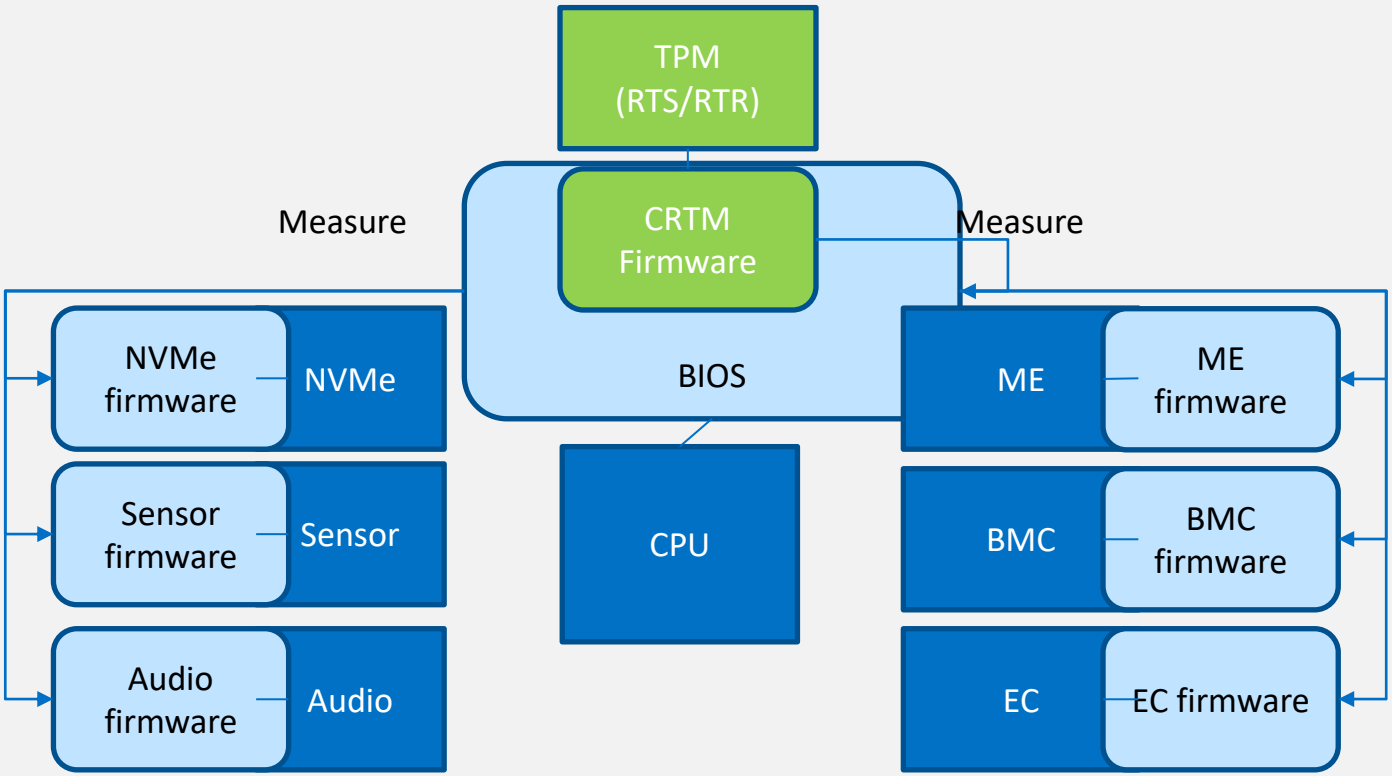


UEFI PI will measure OS loader & UEFI drivers into TPM (1.2 or 2.0) PCR (Platform Configuration Register)

- UEFI Secure boot will stop platform boot if signature not valid (OEM to provide remediation capability)
- UEFI will require remediation mechanisms if boot fails

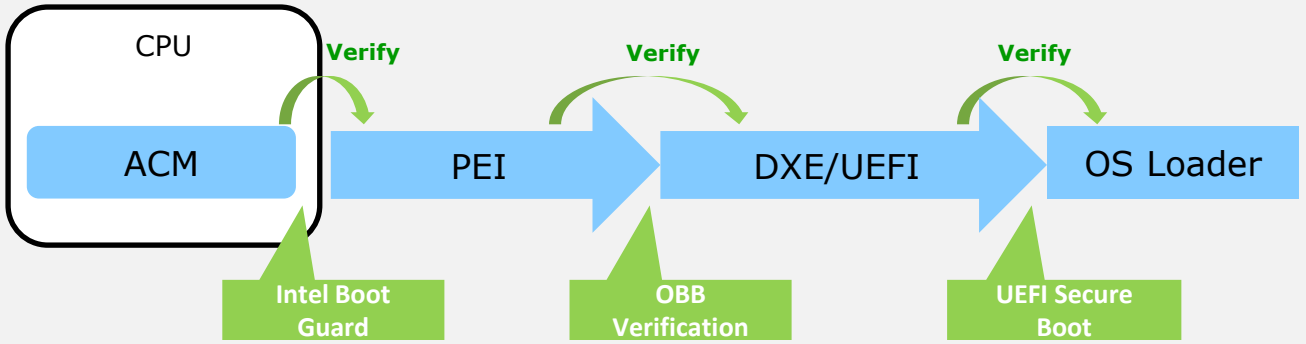
- TCG Trusted boot will never fail
- Incumbent upon other software to make security decision using attestation

The TPM and Measurements

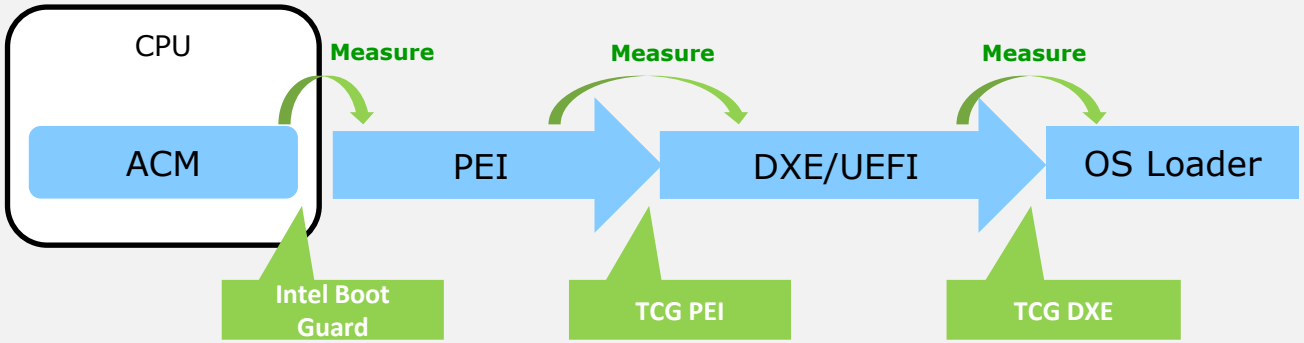


Secure versus trusted boot in practice

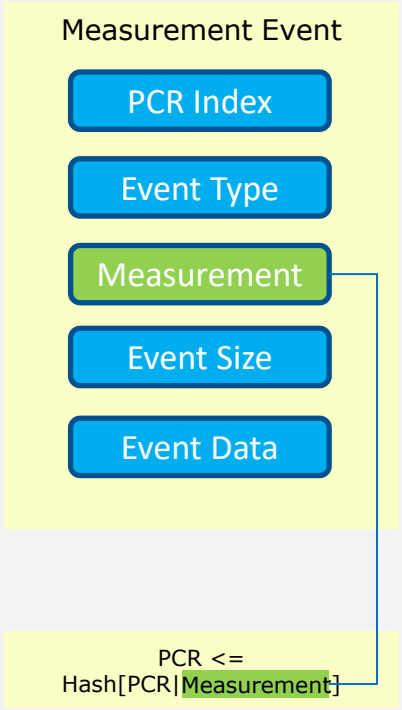
Secure Boot



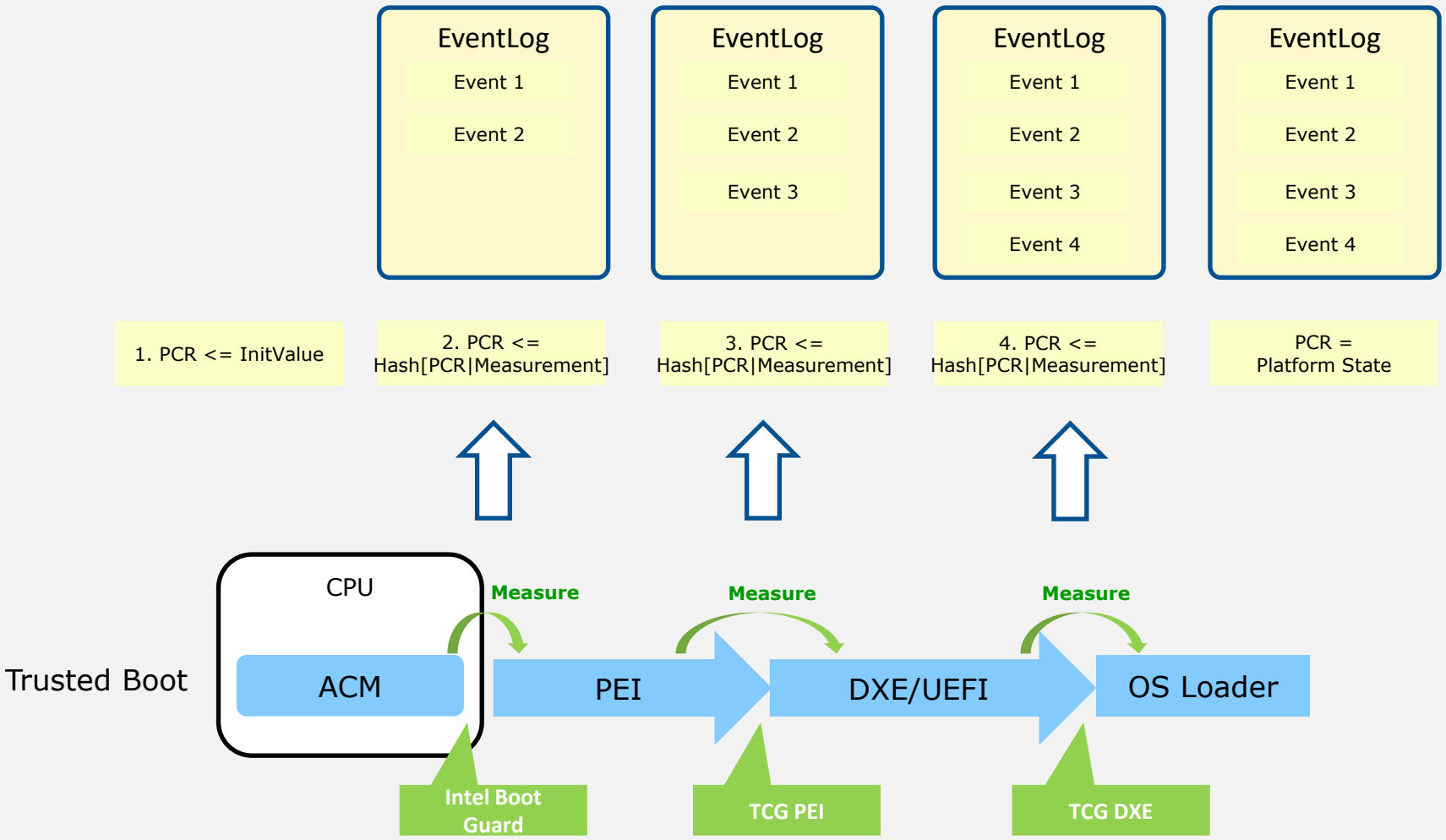
Trusted Boot



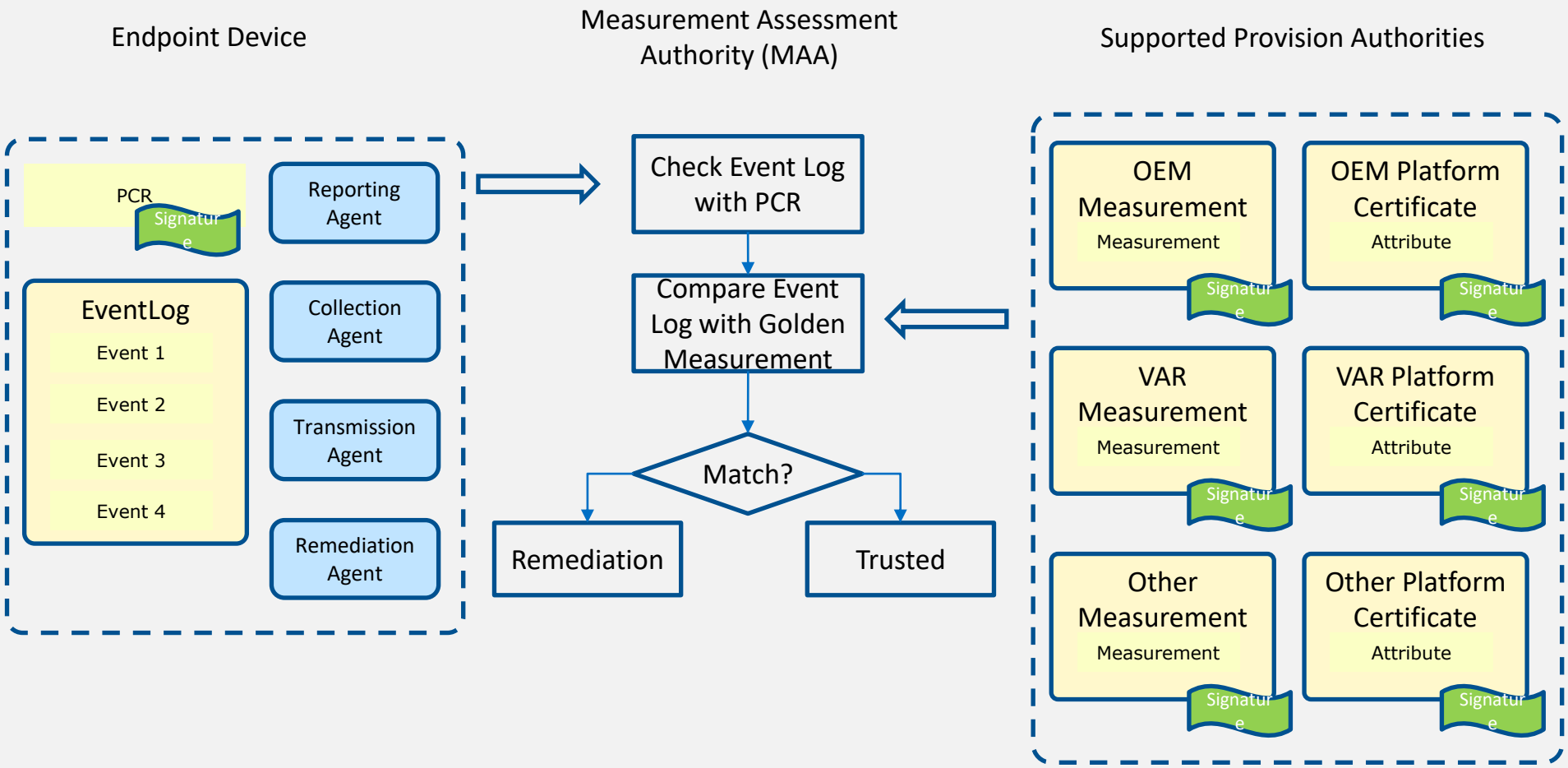
Details on measurements



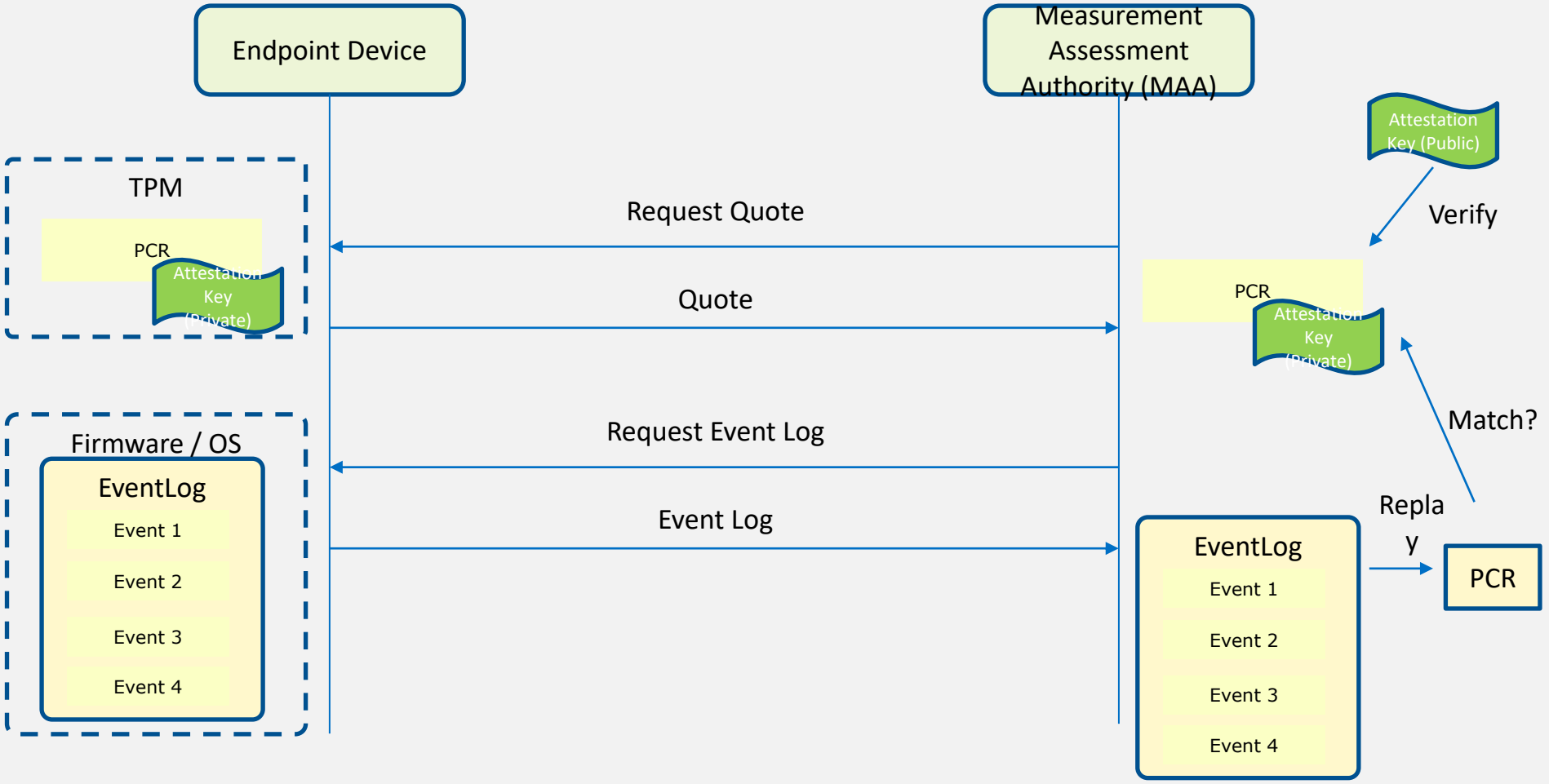
Evolving measurements during boot



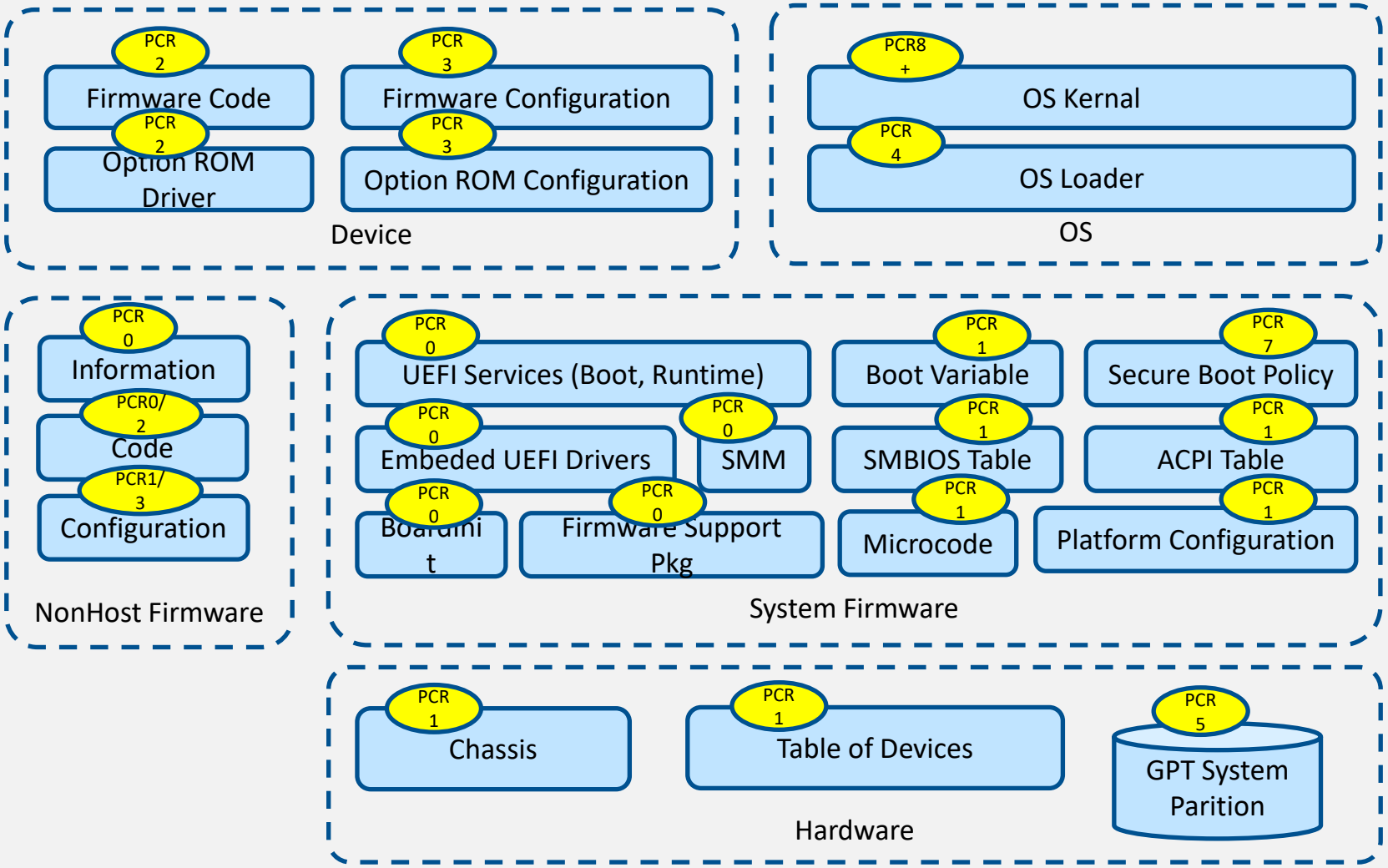
Trust ecosystem around measurements



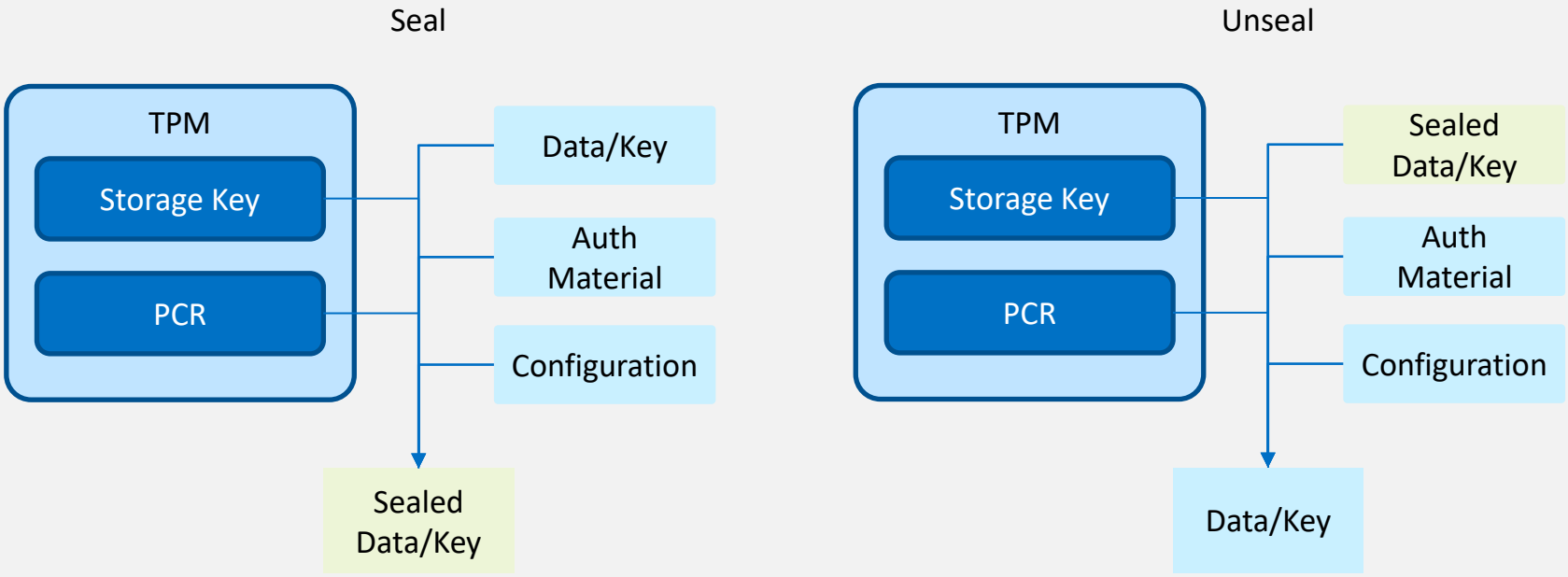
Attestation flow w/ measurements



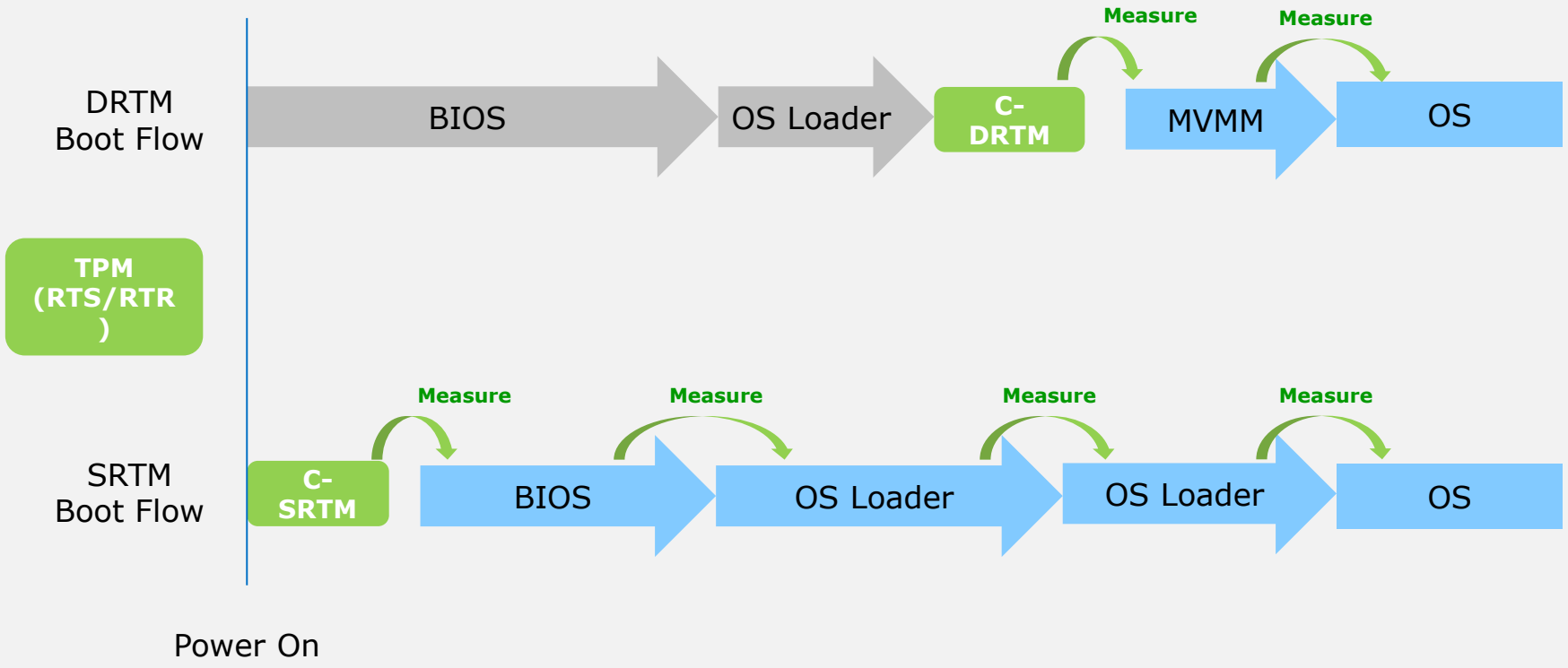
Objects measured during boot



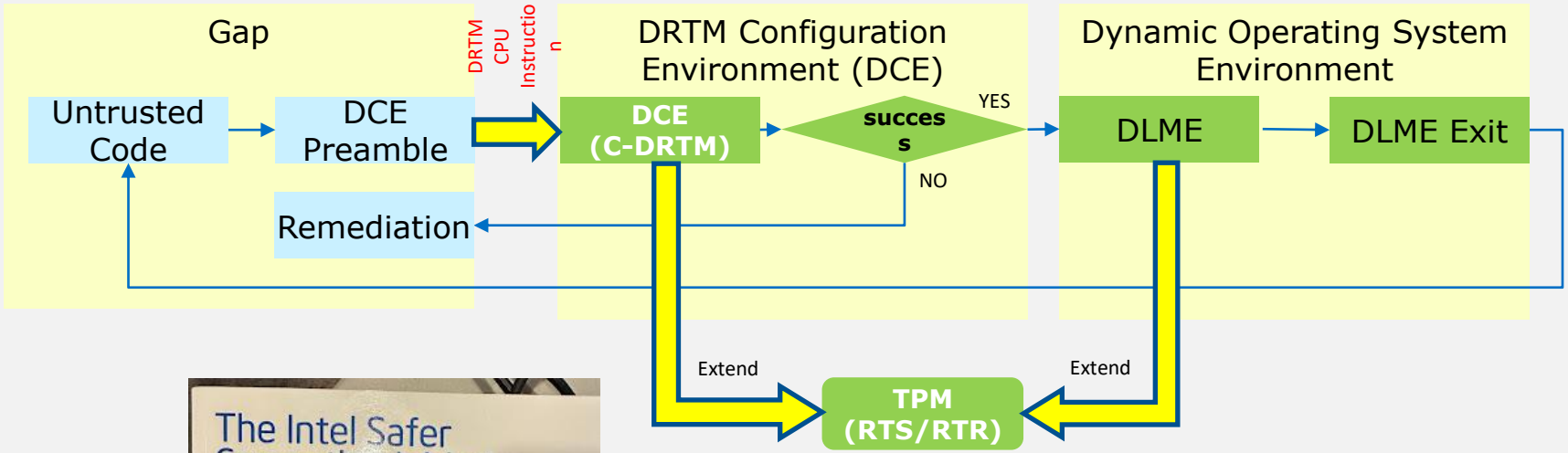
Use of measurements to release secrets

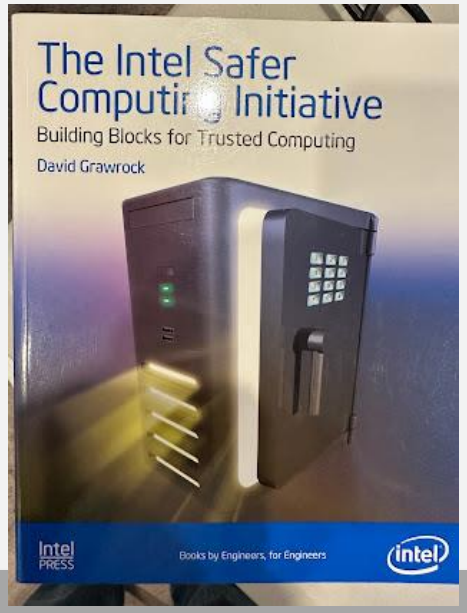


SRTM V.S. DRTM

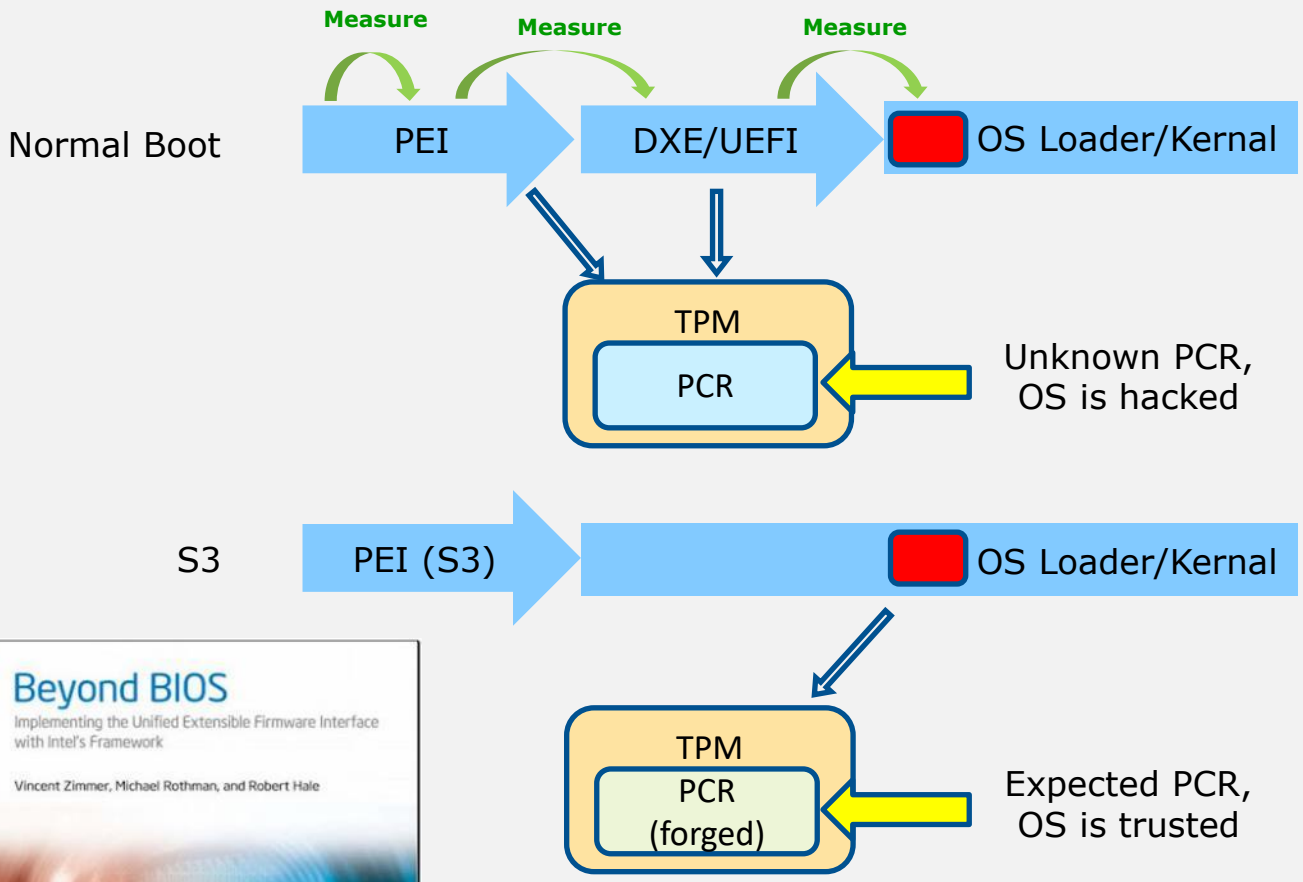

DRTM
Boot Flow
BIOS
OS Loader
C-
DRTM
Measure
MVMM
Measure
OS
TPM
(RTS/RTR
)
SRTM
Boot Flow
C-
SRTM
Measure
BIOS
Measure
OS Loader
Measure
OS Loader
Measure
OS
Power On

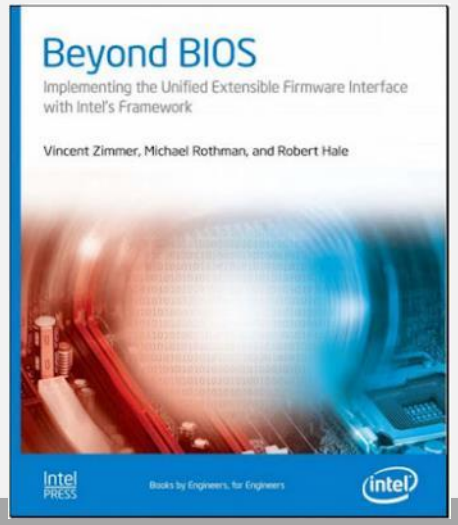
DRTM Boot


Gap
DRTM CPU Instruction
DRTM Configuration Environment (DCE)
Dynamic Operating System Environment
Untrusted Code
DCE Preamble
DCE (C-DRTM)
success
YES
NO
DLME
DLME Exit
Remediation
Extend
Extend
TPM (RTS/RTR)


The Intel Safer Computing Initiative
Building Blocks for Trusted Computing
David Grawrock
Intel PRESS
Books by Engineers, for Engineers
intel

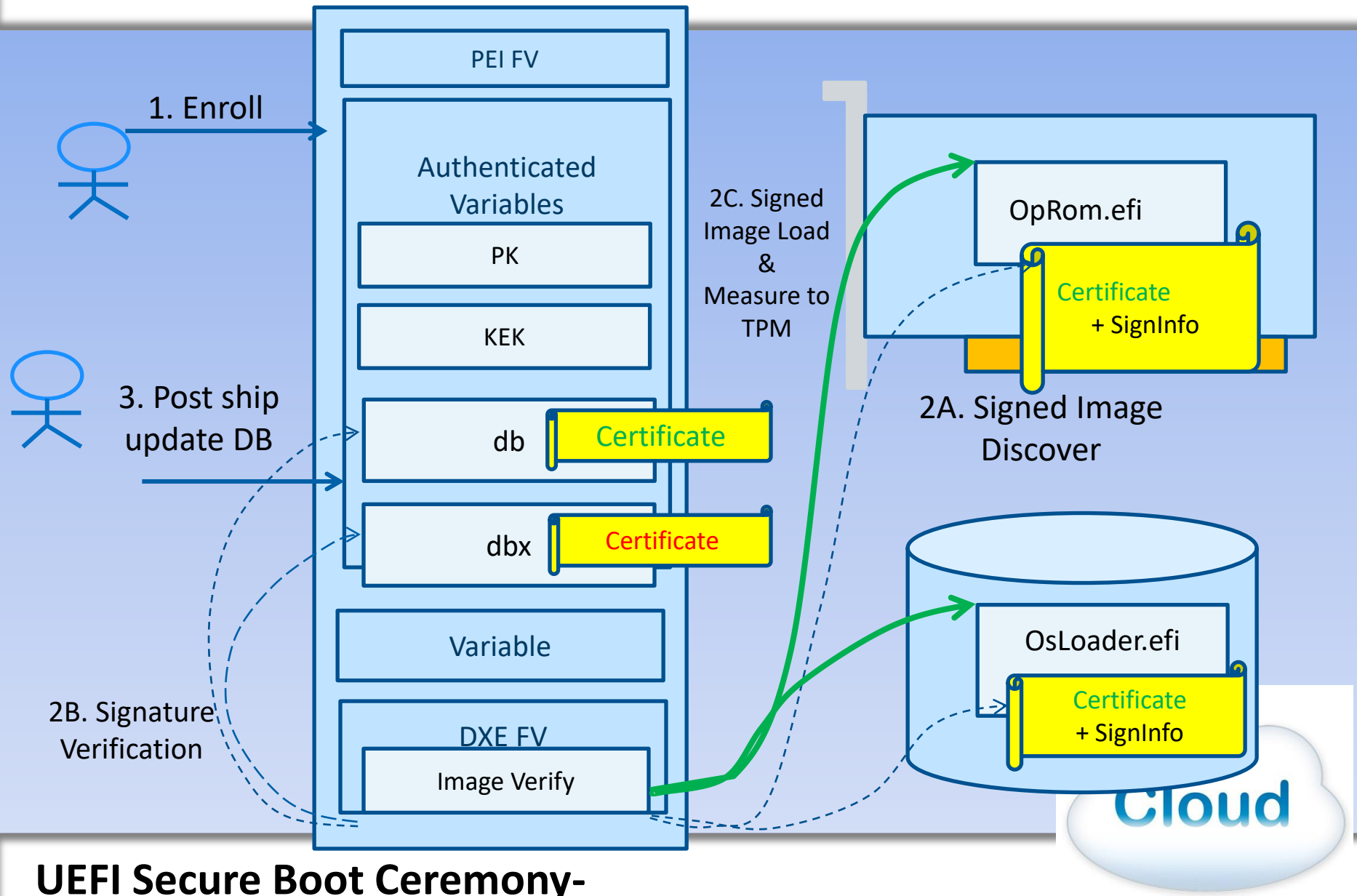
Static Root of Trust for Measurement w/ UEFI





Heaseman 2007
https://www.blackhat.com/presentations/bh-usa-07/Heasman/Presentation/bh-usa-07-heasman.pdf

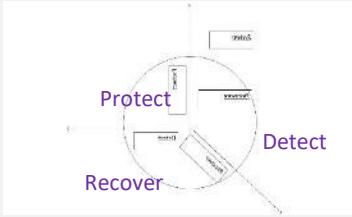
Hacking the Extensible Firmware Interface



UEFI Secure Boot Ceremony-

Ellison, Carl M. "**Ceremony** Design and Analysis," IACR Cryptology ePrint Archive (2007): 399. https://eprint.iacr.org/2007/**399**.**pdf**

EDKII building blocks



UEFI Secure and TCG Measured boot - Upper layer API's defined in UEFI and TCG spec

- Implementation source + document for UEFI Secure & TPM measured boot
  - https://github.com/tianocore/edk2/tree/master/SecurityPkg
  - https://firmware.intel.com/sites/default/files/resources/A_Tour_Beyond_BIOS_Implementing_TPM2_Support_in_EDKII.pdf
- http://bluestop.org/edk2/docs/specs/A_Tour_Beyond_BIOS_into_UEFI_Secure_Boot_White_Paper.pdf

Signed updates - Capsules defined in UEFI Spec

- Signed updates required by NIST 800-147 http://csrc.nist.gov/publications/nistpubs/800-147/NIST-SP800-147-April2011.pdf for core and NIST-193 http://csrc.nist.gov/publications/drafts/800-193/sp800-193-draft.pdf for platform components
- https://github.com/tianocore/edk2/tree/master/SignedCapsulePkg
- https://github.com/tianocore-docs/Docs/raw/master/White_Papers/A_Tour_Beyond_BIOS_Capsule_Update_and_Recovery_in_EDK_II.pdf

A reminder from the KGB school of cipher security:
*"You never attack the **standard**, you attack the **implementation**, including the process." - Grugq*

More information on attacks (and defenses)

https://darkmentor.com/timeline.html

https://arxiv.org/abs/2311.03809

Computer Science > Cryptography and Security

[Submitted on 7 Nov 2023]

SoK: Security Below the OS -- A Security Analysis of UEFI

Priyanka Prakash Surve, Oleg Brodt, Mark Yampolskiy, Yuval Elovici, Asaf Shabtai

The Unified Extensible Firmware Interface (UEFI) is a linchpin of modern computing systems, governing secure
surge in UEFI-related attacks and vulnerabilities in recent years. Motivated by this urgent concern, we undertake

					Security Controls
				EFI firmware of an Apple other two tools. SeaPea operates in the kernel space of Mac OSX, providing stealth and privilege to user-space implants. NightSkies functions at the user-space level, establishing communication with a centralized listening post and enabling remote command and control capabilities [8].	SC-P-007, SC-P-004, SC-P-005, SC-D-002, SC-R-001
2	Sonic Screw-driver	2012	Attack	WikiLeaks claimed that the Sonic Screwdriver tool was created by the Central Intelligence Agency (CIA) to use the Thunderbolt interface to introduce malicious code into the firmware of the targeted Mac OS systems. The tool used the Direct Memory Access (DMA) capabilities of the Thunderbolt interface to infect the firmware even when a firmware password is enabled [9].	SC-D-003, SC-P-008, SC-M-001
3	Der Starke	2013	Attack	An advanced automated implant developed by the CIA for Mac OS X. It operates without relying on a physical disk. This implant maintains its presence within the computer's EFI firmware, making it challenging to detect and remove. Once activated on a target system, Der Starke operates within the disk arbitration process in the macOS and often uses network communications through a web browser to evade detection by personal security programs (PSPs) like Little Snitch [10].	SC-D-007, SC-D-008
4	DreamBoot	2013	PoC	In 2013, Sébastien Kaczmarek developed the "Dreamboot" PoC bootkit, which aimed to exploit the UEFI to attack the OS bootloader. It is noteworthy that Dreamboot could only operate when the Secure Boot mechanism was disabled. Secure Boot is a security feature implemented to ensure that	SC-D-007, SC-D-008, SC-P001

Security Assurance

Tactics	Method	Example
Eliminate Vulnerability	Reduce Attack Surface	• Remove Unnecessary Interface, e.g. SMI handler, private auth variable. • Adopt Firmware Security Best Practice (EDKII security docs, OCP Secure Firmware Development Best Practices)
Break Exploitation	• Data Execution Prevention (DPE) • Control Flow Guard (CFG) • Address Space Layout Randomization (ASLR)	• Non-executable Data Page. Read-only Code page. • Stack Cookie • Intel Control Flow Enforcement Technology (CET) – Shadow Stack (SS), Indirect Branch Tracking (IBT). • ARM Pointer Authentication Code (PAC), Branch Target Identification (BTI). • ASLR in DXE/SMM
Contain Damage	Deprivilege	Ring-3 Third Party Option ROM. Ring-3 OEM SMM
Limit Attack Window		• Live Patching Runtime Component • Firmware Vulnerability Scan • Supply chain - firmware manifest (SBOM)

Reference: https://universalscalablefirmware.github.io/documentation/5_security.html

Possible Security Hardening

- **Data Execution Protection (DEP)**

& Arbitrary Code Guard (ACG)

  - Image Protection
  - Non-Executable Memory protection
  - OS Loader Protection
  - SMM Code Access Check
- **NULL pointer detection**
- **Address Space Layout Randomization (ASLR)**
  - Data Buffer Shift
  - Image Shuffle
- **Buffer Overflow Detection**
  - Heap Guard
  - Stack Cookie
  - Address Sanitizer
- **Misc Runtime Check**
  - Undefined Behavior Sanitizer (Type Cast)
  - Memory Sanitizer (Uninitialized Access)
- **Control Flow**
  - Backward: CET Shadow Stack, ARM PAC
  - Forward: CET IBT, ARM BTI

Reference: https://github.com/jyao1/SecurityEx/blob/master/Summary.md

UEFI/EDKII Security Enhancement summary Code Integrity Guard (CIG)

UEFI Secure Boot

Technology: UEFI image signature verification

Status: *Production*

The platform variable region need use EFI_AUTHENTICATED_VARIABLE_GUID format.

The variable driver need link AuthVariableLib instance.

UEFI secure boot enable/disable is controlled by variable EFI_SECURE_BOOT_ENABLE_NAME:gEfiSecureBootEnableDisableGuid.

PI FV verified boot

Technology: PI firmware volume verification

Status: *Production*

The platform PEI (initial boot block) need verify the OEM boot block (OBB) by using FvReportPei, after memory is discovered.

The platform need install EDKII_PEI_FIRMWARE_VOLUME_INFO_STORED_HASH_FV_PPI to convey FVs and hash information of a specific platform.

Execution Protection (DEP) & Arbitrary Code Guard (ACG)

Image Protection

Technology: Set PE image code region to readonly, data region to be non-executable.

Status: *Production*

DXE controlled by: gEfiMdeModulePkgTokenSpaceGuid.PcdImageProtectionPolicy in MdeModulePkg.dec, SMM enabled by default.

Non-Executable Memory protection

Technology: Set data region to be non-executable

Status: *Production*

DXE controlled by: gEfiMdeModulePkgTokenSpaceGuid.PcdDxeNxMemoryProtectionPolicy in MdeModulePkg.dec, SMM enabled by default.

OS Loader Protection

Technology: BIOS publishes the mem_attribute_protocol. OS loader can use it to protect the image.

Status: *Prototype*

DXE driver is CpuDxe. See Bugzilla 3519

SMM controlled by: gUefiCpuPkgTokenSpaceGuid.PcdCpuSmmCodeAccessCheckEnable in UefiCpuPkg.dec.

Null pointer and ASLR

NULL pointer detection

Technology: mark the first 4K page to be not present to detect NULL pointer dereference

Status: *Production*

Controlled by: gEfiMdeModulePkgTokenSpaceGuid.PcdNullPointerDetectionPropertyMask in MdeModulePkg.dec.

Address Space Layout Randomization (ASLR)

Image Shuffle

Technology: Shuffle the loaded image

Status: *Prototype*

ImageShuffle is configured by PcdImageShuffleEnable. DXE prototype is at DxeCore, SMM prototype is at PiSmmCore.

Data Buffer Shift

Technology: Shift the data buffer - heap and stack

Status: *Prototype*

Randomization is configured by PcdASLRMinimumEntropyBits, DXE prototype is at DxeCore and DxeIpl, SMM prototype is at PiSmmCore

Buffer overflow detection

Stack Guard

Technology: Use guard page to detect global stack overflow.

Status: *Production*

DXE controlled by: gEfiMdeModulePkgTokenSpaceGuid.PcdCpuStackGuard in MdeModulePkg.dec, SMM controlled by: gUefiCpuPkgTokenSpaceGuid.PcdCpuSmmStackGuard in UefiCpuPkg.dec.

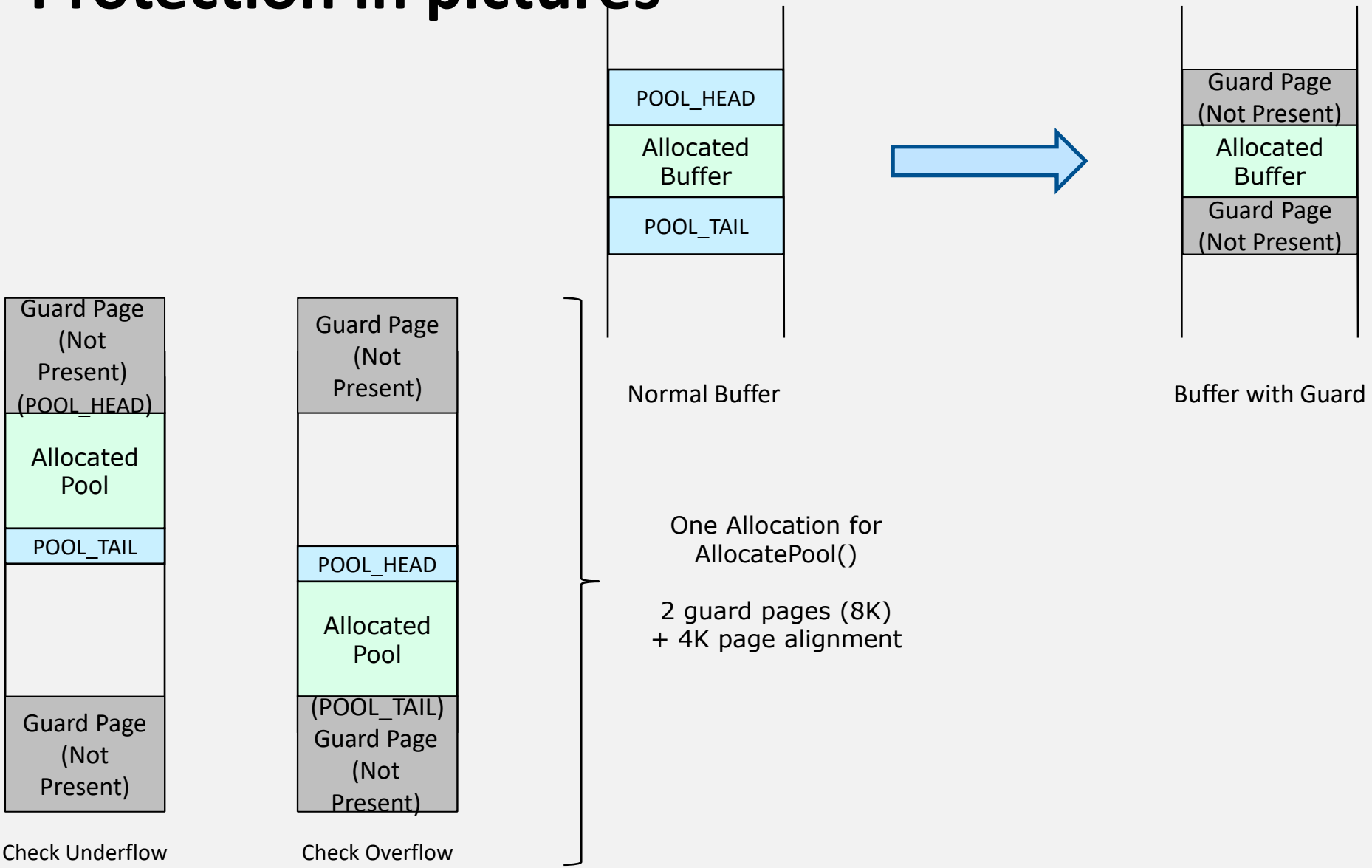
Heap Guard

Technology: Use guard page to detect heap overflow.

Status: *Debug*

Controlled by: gEfiMdeModulePkgTokenSpaceGuid.PcdHeapGuardPropertyMask in MdeModulePkg.dec, gEfiMdeModulePkgTokenSpaceGuid.PcdHeapGuardPageType in MdeModulePkg.dec, gEfiMdeModulePkgTokenSpaceGuid.PcdHeapGuardPoolType in MdeModulePkg.dec.

Protection in pictures



More buffer overflow detection

Stack Canary

Technology: Use compiler to insert cookie to detect local stack overflow (need compiler support)

Status: *Prototype*

MSVC compiler stub (/GS) prototype is at GSStub.c, GCC/LLVM compiler stub (-fstack-protector-strong) prototype is at StackProtectorStub.c.

Address Sanitizer

Technology: Use compiler to insert redzone to detect buffer overflow (need compiler support)

Status: *Prototype*, *Debug*

MSVC compiler stub (/RTCs) prototype is at RTCsStub.c, LLVM compiler stub (-fsanitize=address) prototype is at ASanStub.c.

Miscellaneous runtime checks

Undefined Behavior Sanitizer (Type Cast)

Technology: Use compiler to insert runtime check for undefined behavior such as type cast. (need compiler support)

Status: *Prototype*, *Debug*

MSVC compiler stub (/RTCc) prototype is at RTCcStub.c, LLVM compiler stub (-fsanitize=undefined) protype is at UBSanStub.c.

Memory Sanitizer (Uninitialized Access)

Technology: Use compiler to insert check to detect uninitialized data read. (need compiler support)

Status: *Prototype*, *Debug*

MSVC compiler stub (/RTCu) prototype is at RTCuStub.c, LLVM (-fsanitize=memory) cannot be enabled because it does not support windows platform yet.

Control flow

Shadow Stack (Intel CET-SS)

Technology : return address protection to defend against Return Oriented Programming

Status: SMM *production*, DXE *prototype*

SMM shadow stack is controlled by gEfiMdePkgTokenSpaceGuid.PcdControlFlowEnforcementPropertyMask in MdePkg.c, DXE shadow stack prototype is at DxeCet.

Indirect Branch Tracking (Intel CET-IBT)

Technology : free branch protection to defend against Jump/Call Oriented Programming (need compiler support)

Status: *Prototype*

Prototype is at Ibt. The IBT cannot be enabled in MSVC, because the compiler does NOT support it yet.

Software Control Flow Integrity/Guard (CFI/CFG)

Technology : Use compiler to insert control flow check to detect control flow attack (need compiler support)

Status: *Prototype*

MSVC compiler stub (/guard:cf) prototype is at CfgStub.c, LLVM compiler stub (-fsanitize=cfi) prototype is at CfiStub.c.

Pre-boot DMA

IOMMU Engine Based Protection (Intel VTd)

Technology : Enable IOMMU in BIOS to prevent DMA attack from device.

Status: *Production*

DXE enabled by: IntelVTdDxe, PEI enabled by: IntelVTdDmarPei.

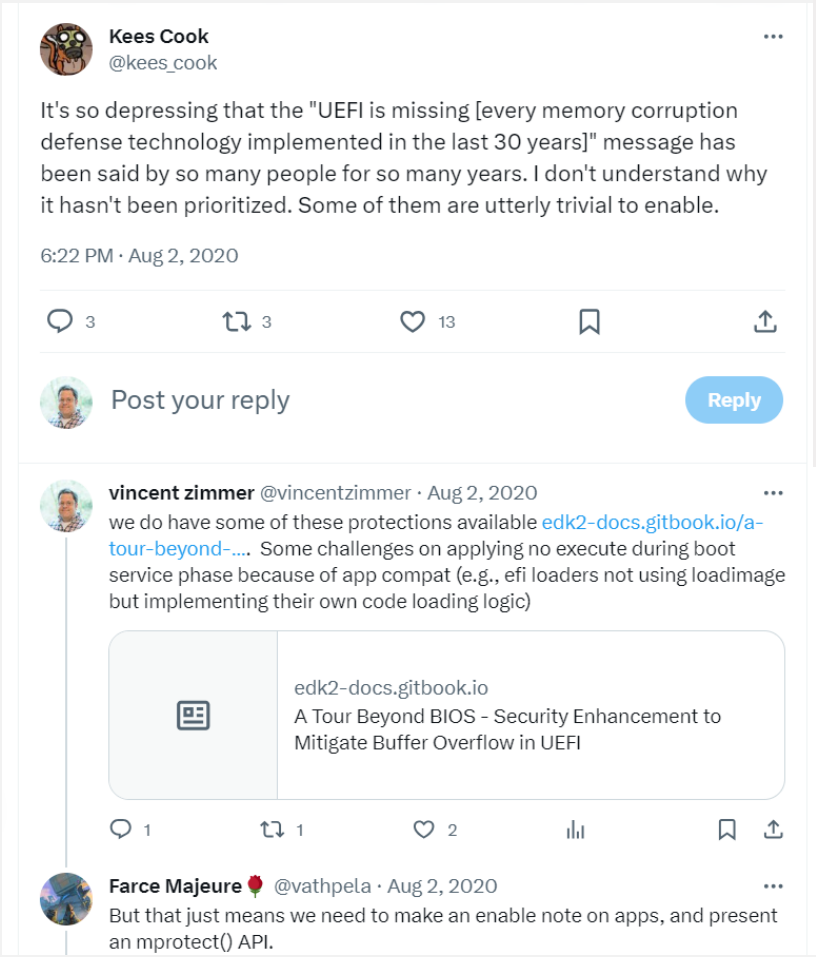
Silicon specific DMA Protection (Intel VTd PMR)

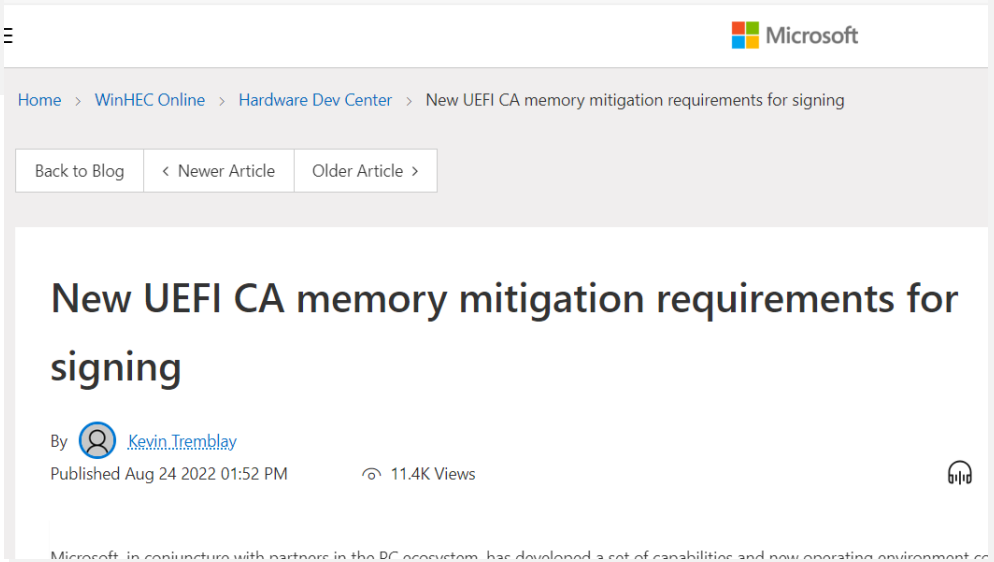
Technology : Enable Protected Memory Region (PMR) in PEI phase as a lightweight solution.

Status: *Production*

PEI enabled by: IntelVTdPmrPei.

Compatibility





MS UEFI CA Requirement for 4K

EFI_MEMORY_ATTRIBUTE_PROTOCOL

https://twitter.com/kees_cook/status/1290095780984786952

The reality of shipping products

Features
Value-add

Quality /
assurance

Budget

Schedule

However …

- UEFI / PI / APCI are interface specifications
- How do we let end users know what protection is available?

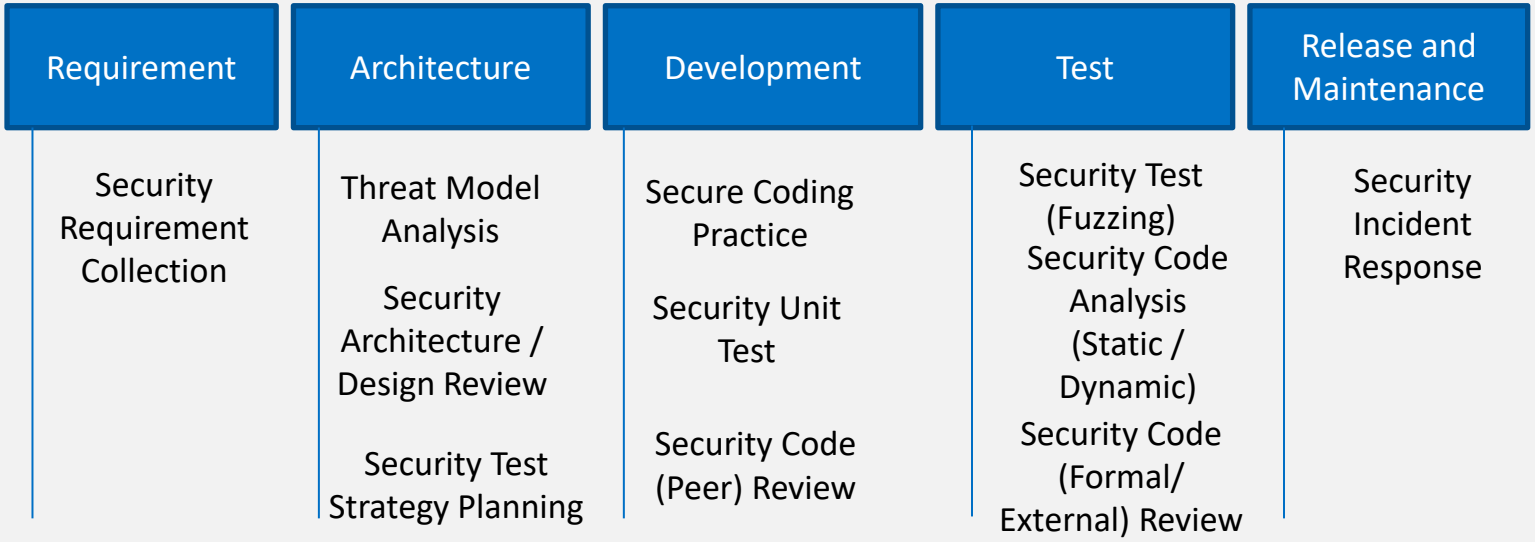
Example

- **Windows SMM Security Mitigation Table (WSMT)**
  - Allows system firmware to confirm to the operating system that certain security best practices have been implemented in SMM
  - https://download.microsoft.com/download/1/8/a/18a21244-eb67-4538-baa2-1a54e0e490b6/wsmt.docx
- **Windows Hardware Security Test Interface (HSTI)**
  - Specifies a standard test interface for proprietary platform security technologies that enforce the Secure Boot promis
  - https://learn.microsoft.com/en-us/windows-hardware/test/hlk/testref/hardware-security-testability-specification
- **TCG Platform Firmware Integrity Measurement**
  - Platform Firmware Assertions can be reported in the platform certificate.
  - E.g. HardwareSRTM, SecureBoot, sp800-147, sp800-193, fwSetupAuthLocal, SMMProtection, fwKernelDMAProtection, etc.
  - https://trustedcomputinggroup.org/resource/tcg-pc-client-platform-firmware-integrity-measurement/

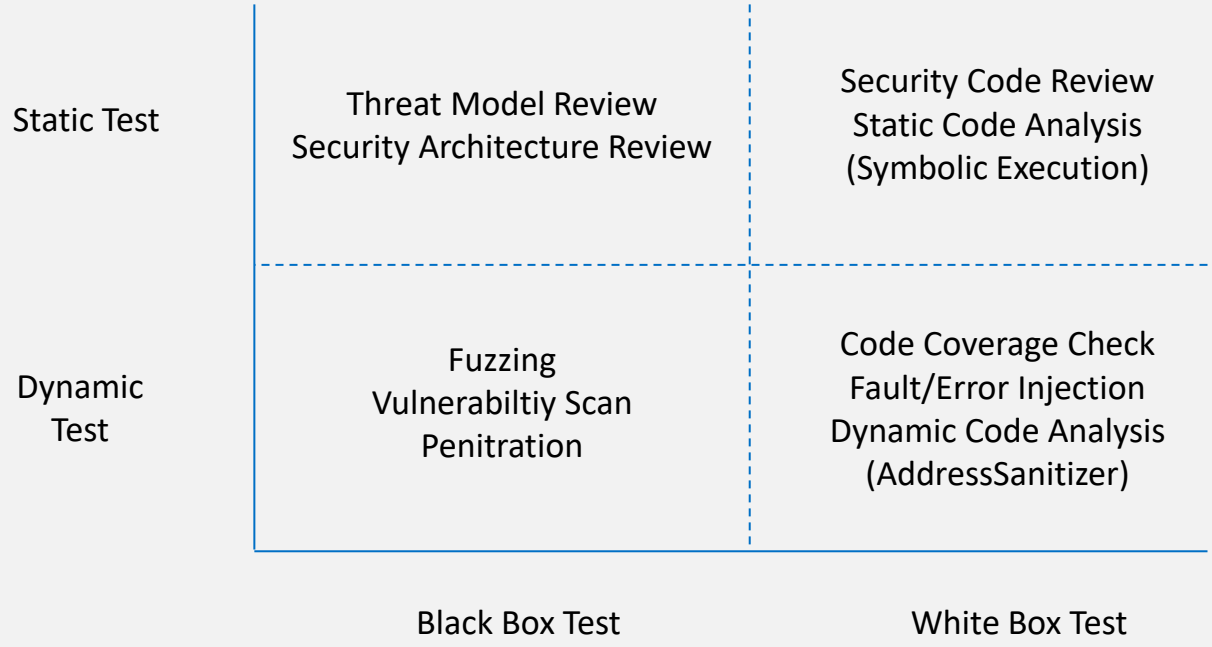
Request for comment

- **Platform Integrity Mitigation Table (PIMT)**
  - Specifies the mitigation applied in the system firmware
    - DEP.CodeProtection, DEP.NonExecutableData, NULLPointerProtection, ASLR.BufferShift, ASLR.ImageShuffle, CFG.Backward, CFG.Forward.
  - Could be ACPI table or GUIDed UEFI system table
    - ACPI better since all of ACPI most common across all platform implementations (slim, core, and EDKII)

The lifecycle of activities for firmware



Spectrum of testing opportunities



Open source tool examples include:

https://github.com/intel/tsffs ← https://ieeexplore.ieee.org/document/9218694

https://github.com/tianocore/tianocore.github.io/wiki/Host-Based-Firmware-Analyzer

https://github.com/chipsec/chipsec

Putting it together

- Provide an example platform to the security community https://www.intel.com/content/dam/develop/external/us/en/documents/stts003-sf15-stts003-100f-820238.pdf

- Open source core
  - www.github.com/tianocore/edk2

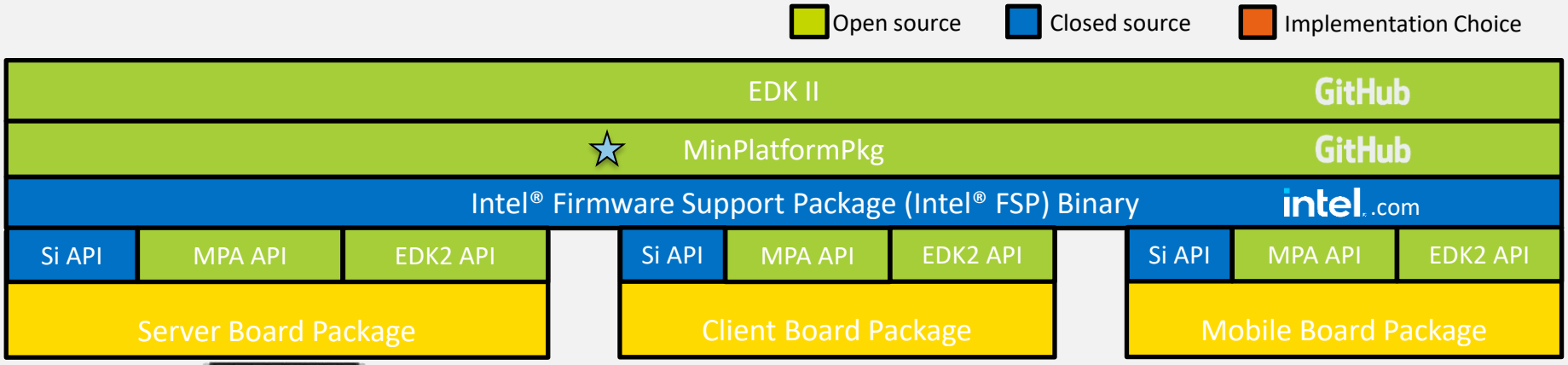
+

- Open source platform code
  - www.github.com/tianocore/edk2-platforms

+

- Binaries
  - www.github.com/intel/fsp
  - www.github.com/tianocore/edk2-non-osi

Intel Open Platform Firmware Stack - MinPlatform



MinPlatform + Intel® FSP

Consistent boot flows and interfaces
Approachable across the ecosystem
Scalable from pre-silicon to derivatives

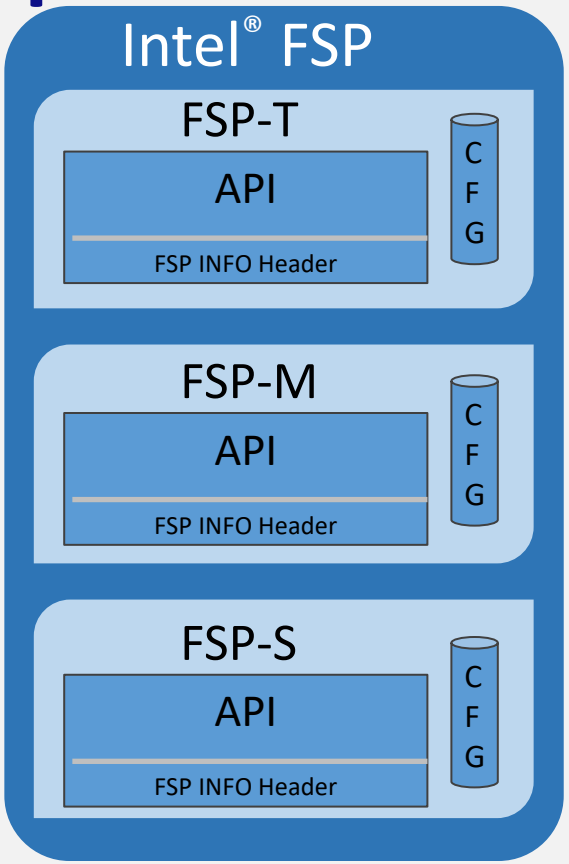
Whitley Open Board Platform support for Aowanda (IceLake-SP) added June 2022

Intel® FSP V2.4 Binary Component View

Firmware Volume Layout of the Intel FSP Binary

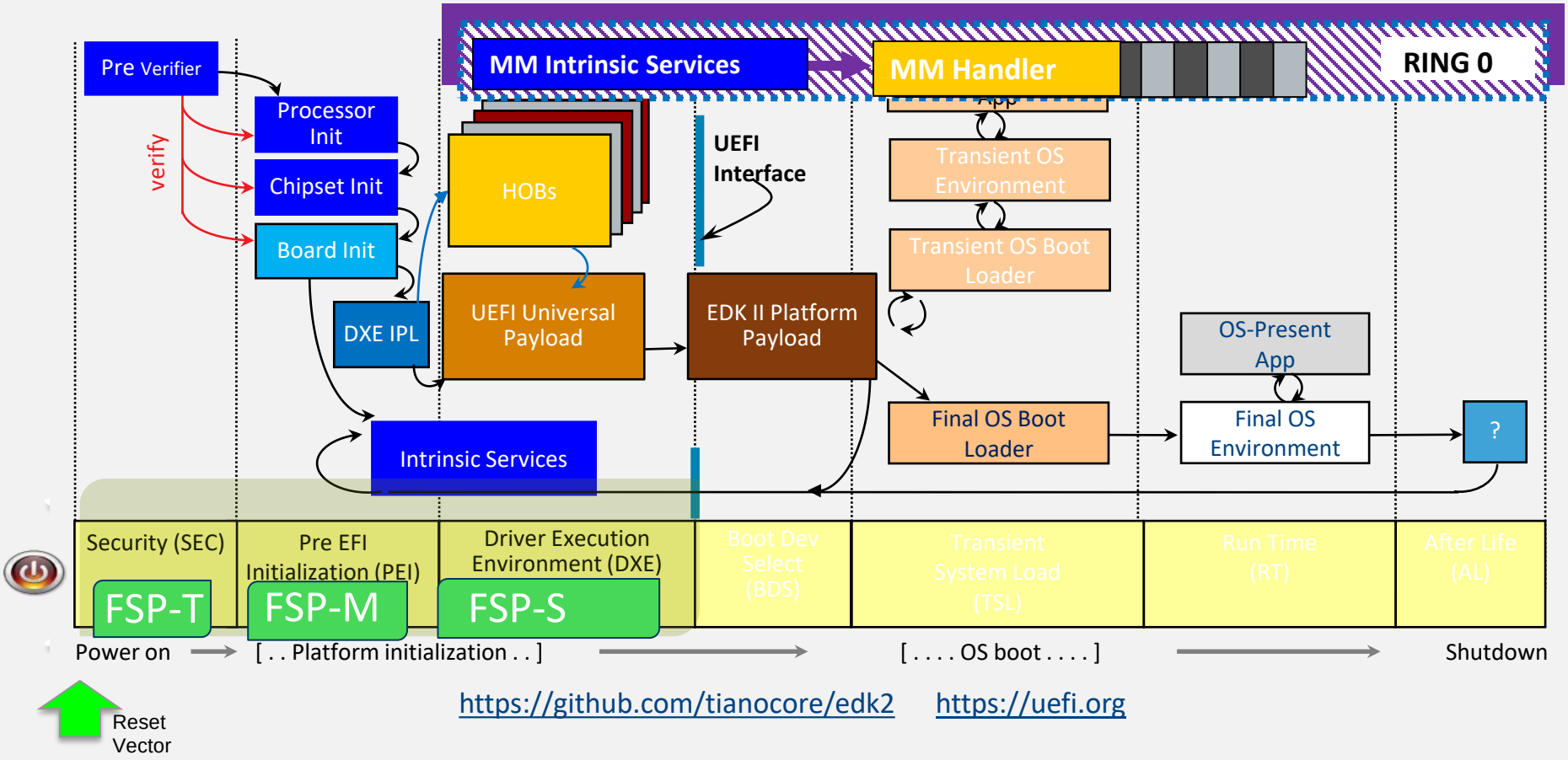
- FSP-T: Temporary RAM initialization phase
  - `TempRamInit()`
- FSP-M: Memory initialization phase
  - `FspMemoryInit()`
  - `TempRamExit()`
- FSP-S: Silicon initialization phase
  - `FspSiliconInit()`
  - `NotifyPhase()`
  - `FspMultiPhaseSiInit()`

https://www.intel.com/fsp
https://github.com/intel/fsp



UEFI – PI & EDK II Boot Flow – FSP w/ USF

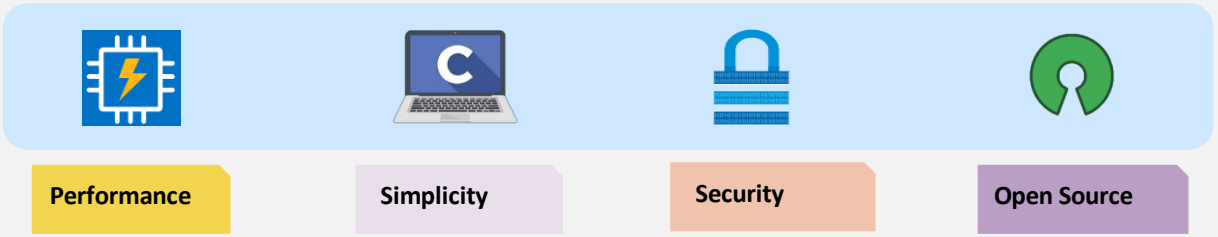
tianocore



https://github.com/tianocore/edk2 https://uefi.org

- Core (essential, stripped-down firmware) boot (to boot the platform)

- Idea: Perform basic hardware initialization before passing control to a payload** that boots the OS***
- Principles:



https://coreboot.org

Slim bootloader Boot Stages

- Stage 1A
  - Reset Vector stage starts with assembly code
  - Basic initialization including setting up temporary memory, debug output
- Stage 1B
  - Memory initialization stage
  - Loads configuration data

Stage 2

Post Memory stage
Silicon initialization
ACPI, PCI Enumeration, etc

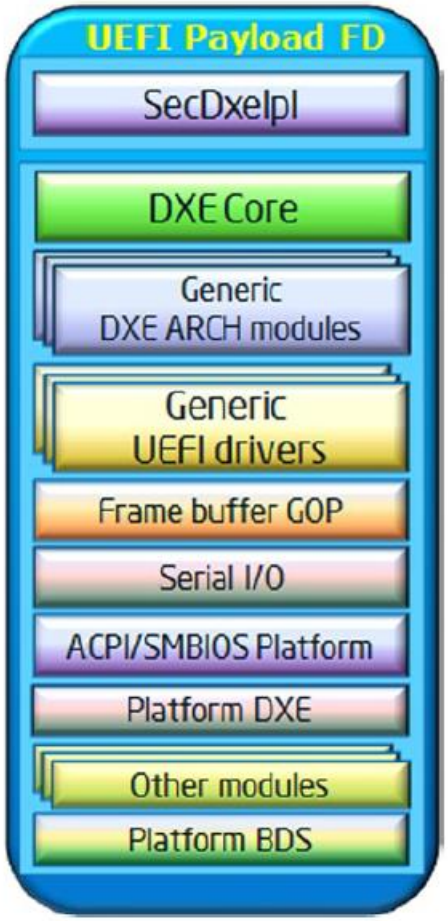
OsLoader / FWU Payload

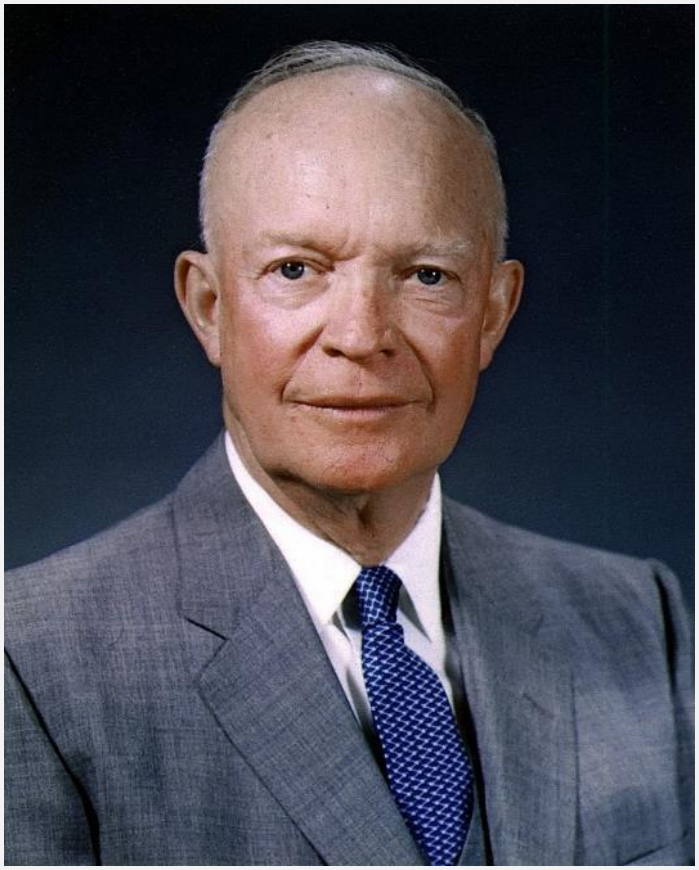
OS boot logic
Media drivers

https://slimbootloader.github.io/

UEFI support Payload – one way to share EDKII code across many systems

https://github.com/tianocore/edk2/tree/master/UefiPayloadPkg



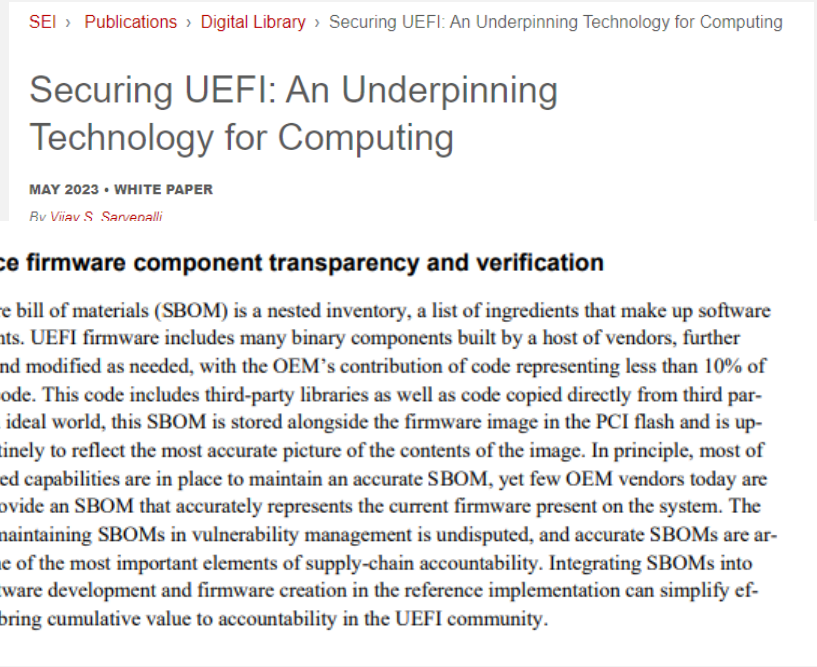


We will bankrupt ourselves in the vain search for absolute security.

Dwight D. Eisenhower

Recent focus on UEFI supply chain - SBOM

A Call to Action: Bolster UEFI Cybersecurity Now | CISA



Securing UEFI: An Underpinning Technology for Computing (cmu.edu)

Post quantum readiness

Table 3. UEFI firmware potential asymmetric algorithm touchpoint

Category	Use Case	Standard	Algorithm	Comment
Firmware Image Verification	UEFI Secure Boot – Image Verification	UEFI, Authenticode [15], [17]	Stateful HBS	Stateful HBS Algo ID WIP [18].
	UEFI FMP Signed Capsule	UEFI, PKCS7 [16]	Stateful HBS	Stateful HBS Algo ID WIP [18].
	PI Signed FV/Section	PI	Stateful HBS	Need add stateful HBS.
Firmware Data Verification	UEFI Secure Boot – Auth Variable Update	UEFI, PKCS7 [16]	Stateful HBS / PQC SIG	Stateful HBS Algo ID WIP [18].
Network Secure Communication	TLS (HTTPS boot, RedFish)	IETF TLS [21], [22]	PQC KEM + PQC SIG	PQC Support WIP [23]. PQC SIG Algo ID WIP [24].
	IPSec	IETF IPSec [25]	PQC KEM + PQC SIG	PQC Support WIP [26].
Device Secure Communication	SPDM	DMTF SPDM [27]	PQC KEM + PQC SIG	PQC Support WIP.

Table 2. CNSA 2.0 timing requirement (Source: [8])

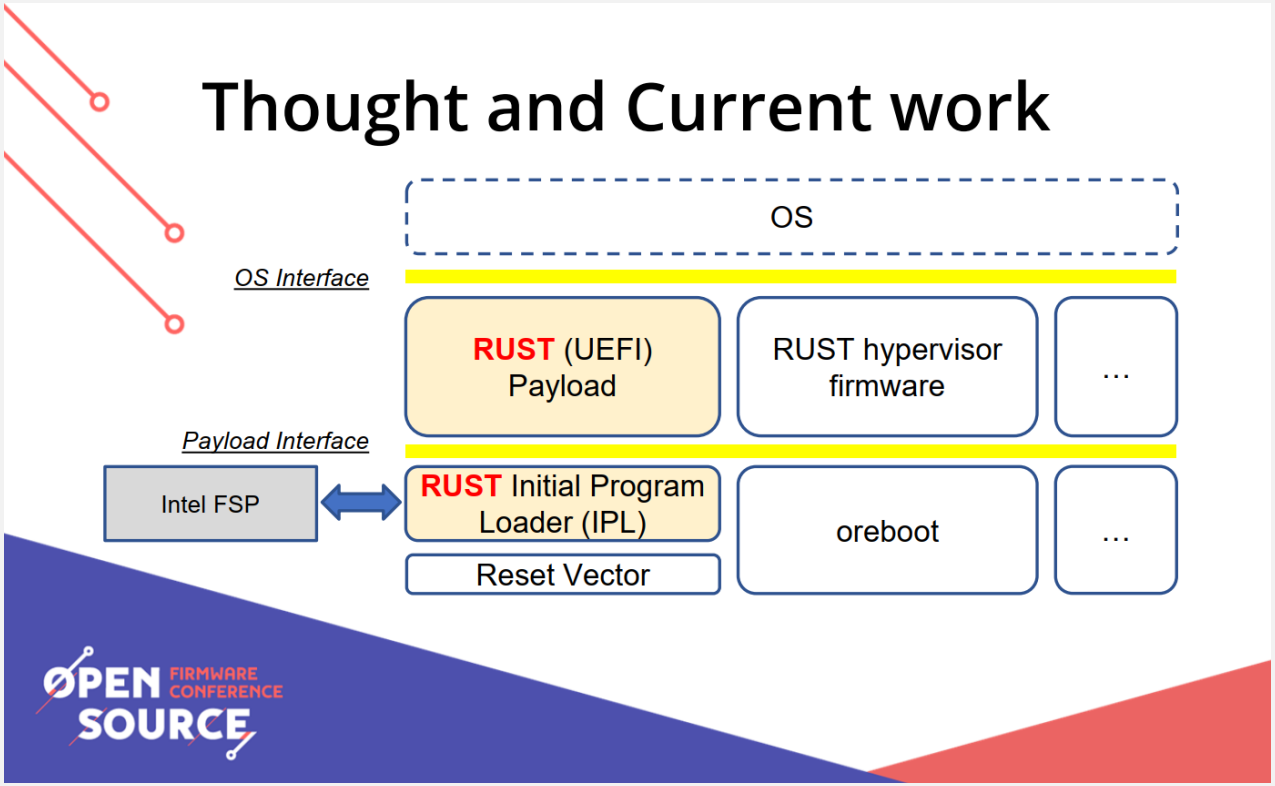
Category	Support and preferred	Exclusively used
Software and firmware signing	2025	2030
Web browsers/servers and cloud services	2025	2033
Traditional networking equipment	2026	2030
Operating systems	2027	2033
Niche equipment	2030	2033
Custom applications and legacy equipment	2033	2033

https://bugzilla.tianocore.org/show_bug.cgi?id=4087

Futures?

Rust Language

More reusable payloads of Features



https://www.osfc.io/2020/talks/enabling-rust-for-uefi-firmware

Learn more about system firmware development

https://link.springer.com/book/10.1007/978-1-4842-6106-4
https://link.springer.com/book/10.1007/978-1-4842-7939-7
https://link.springer.com/book/10.1007/978-1-4842-7974-8
https://www.oreilly.com/library/view/beyond-bios/9781501505836/
https://link.springer.com/book/10.1007/978-1-4842-0070-4
https://learning.oreilly.com/library/view/harnessing-the-uefi/9781501505812/

More references

https://github.com/tianocore/tianocore.github.io/wiki/EDK-II-Security-White-Papers

www.uefi.org

https://uefi.org/specifications

https://uefi.org/learning_center/presentationsandvideos

www.tianocore.org

Questions?