ToorCamp 2014

Secure Boot, Network Boot, Verified Boot, oh my

Vincent Zimmer

Usual disclaimer-
These foils and opinions are mine and not necessarily those of my employer

Agenda

History

UEFI Overview

Secure boot

Network boot

Coreboot

Verified boot

Building it

Testing it

Background

ToorCamp 2012

- Talked about UEFI Secure boot in 2012
- New features, new ecosystem
- Open core/closed platform

ToorCamp 2014

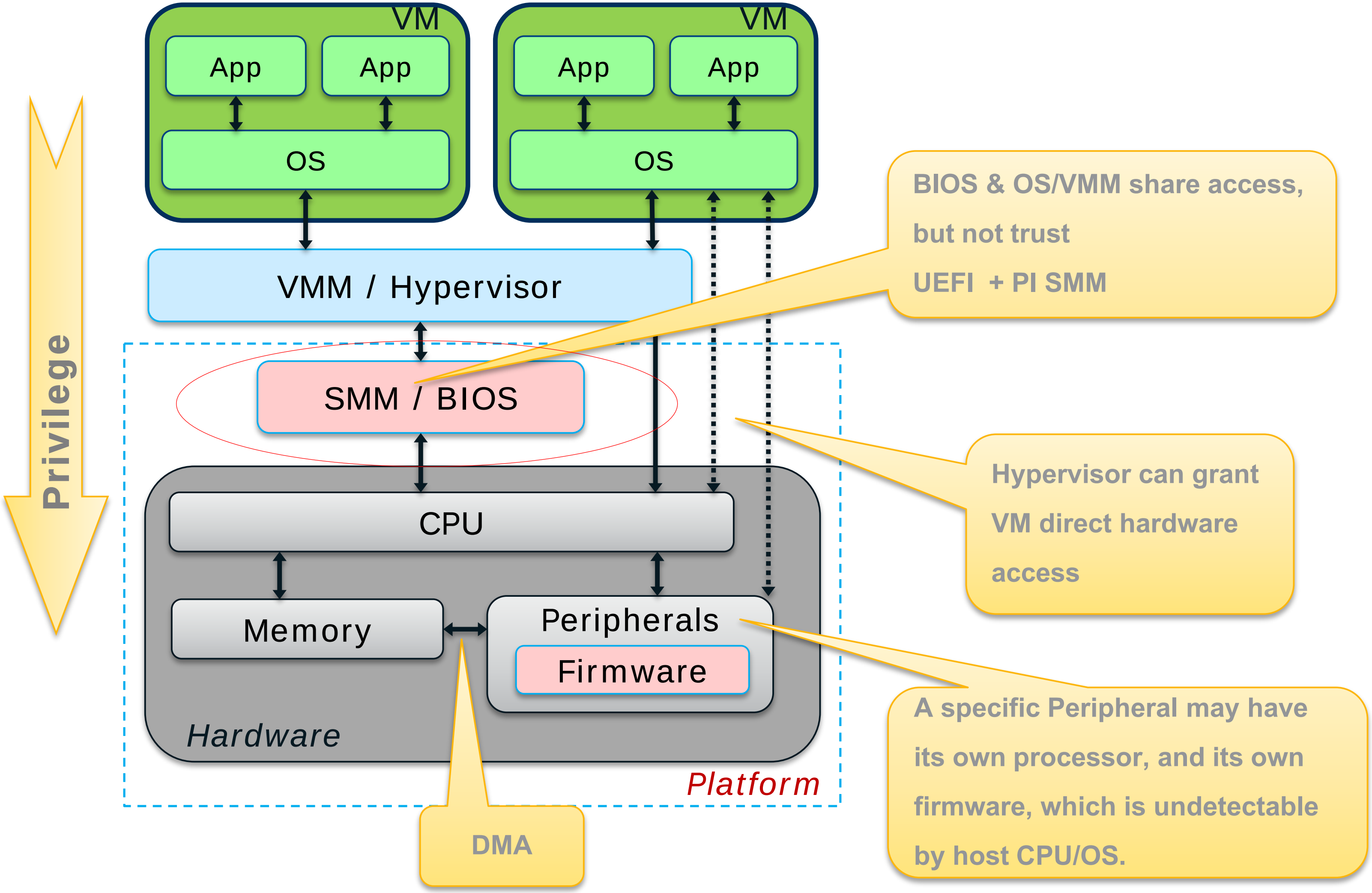
- Shipped several generations on UEFI
- More open platforms, ARM32/64 added, other fw

Challenge

- More attacks, more defenses, more scale

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

Where are we (BIOS / UEFI firmware / Coreboot)?

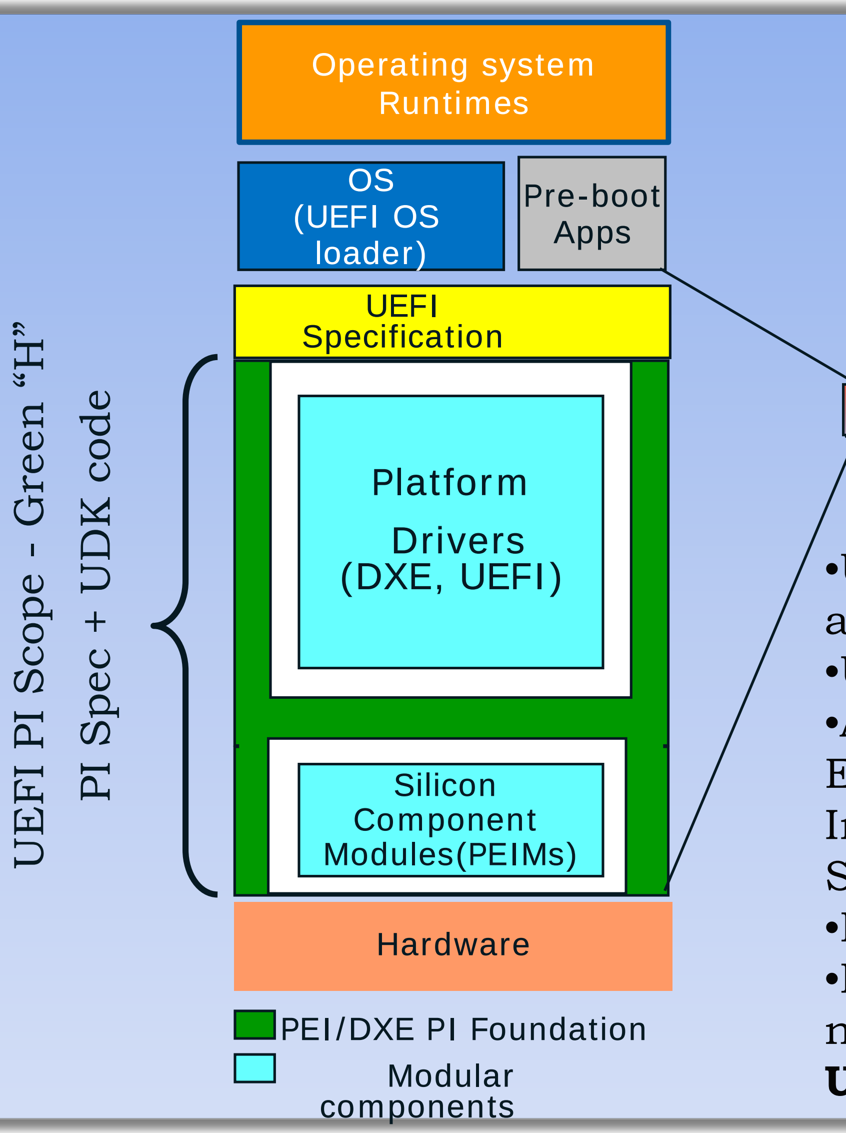


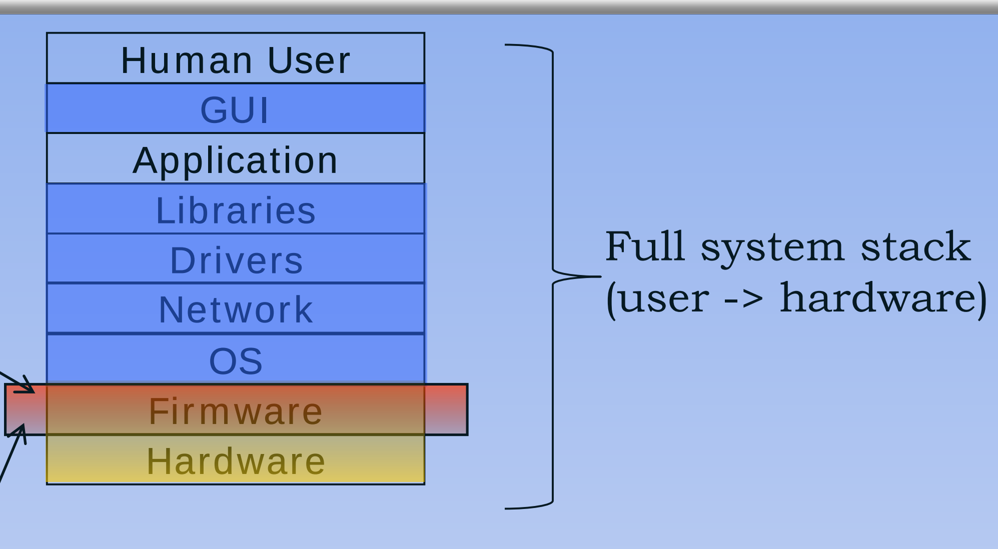
BIOS & OS/VMM share access, but not trust

UEFI + PI SMM

Hypervisor can grant VM direct hardware access

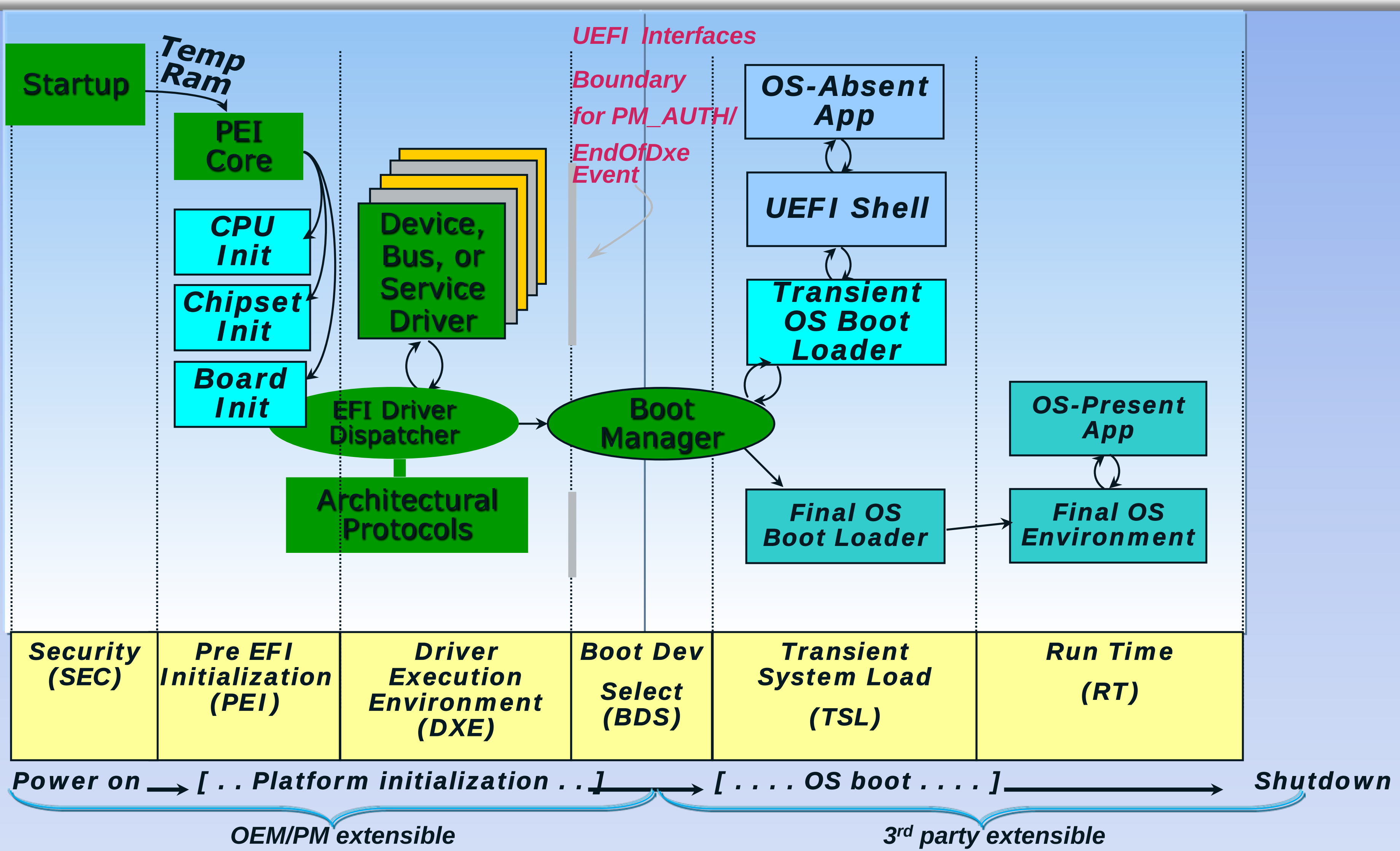
A specific Peripheral may have its own processor, and its own firmware, which is undetectable by host CPU/OS.





- UEFI 2.4 specifies how firmware boots OS loader and drivers– **3rd party extensible**
- UEFI's Platform Initialization (PI) 1.2.1
- Architecture specifies how Driver Execution Environment (DXE) Drivers and Pre-EFI Initialization (PEI) Modules (PEIMs) initialize Silicon and the platform
- DXE is preferred UEFI Implementation
- PEIMs, UEFI and DXE drivers implements networking, Update, other security features - **Only Update by the OEM**

Stacking UEFI – Platform Initialization (PI)



UEFI Interfaces

Boundary for PM_AUTH/ EndOfDxe Event

Startup

Temp Ram

PEI Core

CPU Init

Chipset Init

Board Init

Device, Bus, or Service Driver

EFI Driver Dispatcher

Architectural Protocols

Boot Manager

OS-Absent App

UEFI Shell

Transient OS Boot Loader

Final OS Boot Loader

OS-Present App

Final OS Environment

Security (SEC)

Pre EFI Initialization (PEI)

Driver Execution Environment (DXE)

Boot Dev Select (BDS)

Transient System Load (TSL)

Run Time (RT)

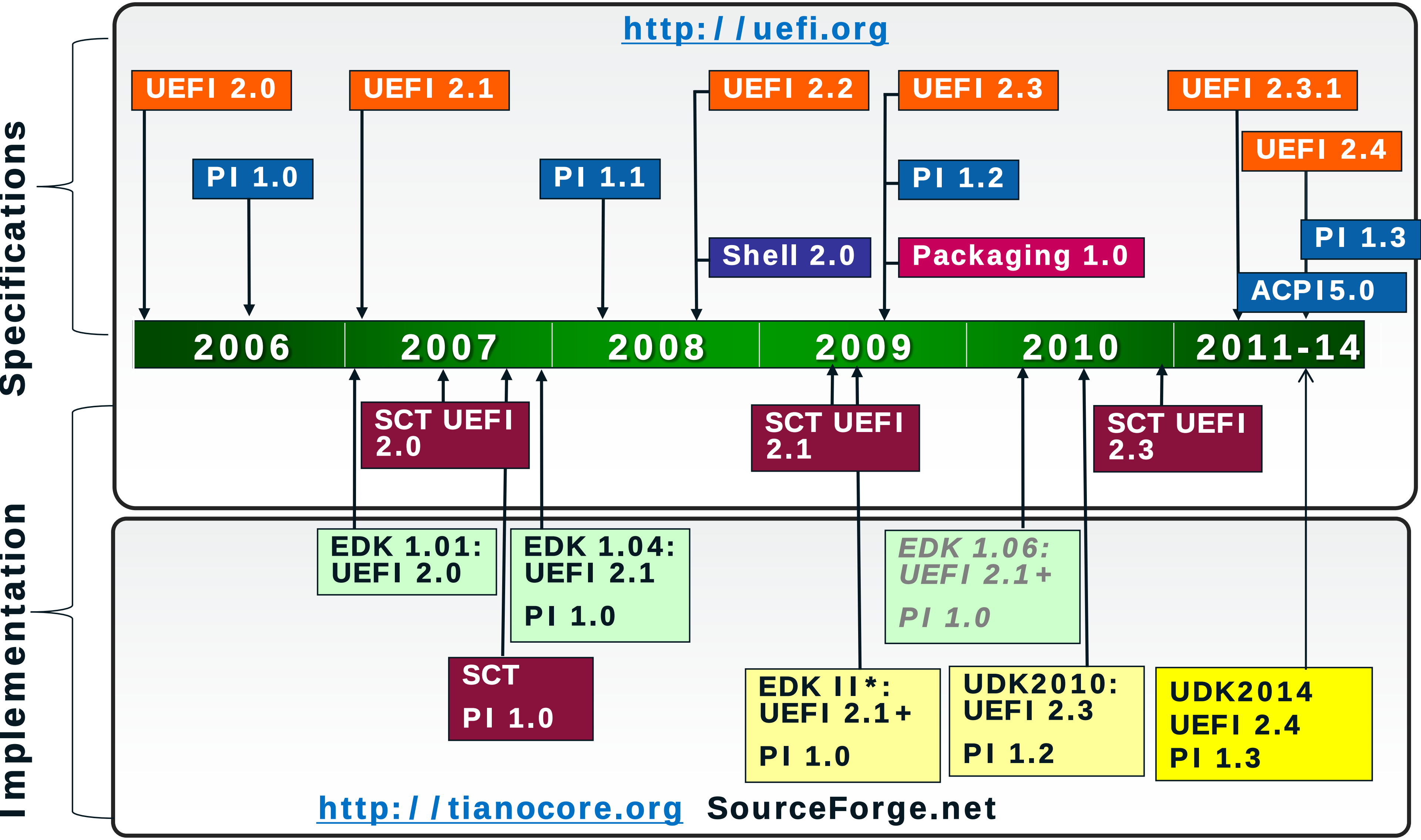
Power on → [. . Platform initialization . .] → [. . . . OS boot . . . .] → Shutdown

OEM/PM extensible

3rd party extensible

Overall UEFI Boot Timeline

Specification & Tianocore.org Timeline



Specifications

http://uefi.org

UEFI 2.0

PI 1.0

UEFI 2.1

PI 1.1

UEFI 2.2

Shell 2.0

UEFI 2.3

PI 1.2

Packaging 1.0

UEFI 2.3.1

UEFI 2.4

PI 1.3

ACPI5.0

2006

2007

2008

2009

2010

2011-14

Implementation

SCT UEFI 2.0

SCT UEFI 2.1

SCT UEFI 2.3

EDK 1.01:
UEFI 2.0

EDK 1.04:
UEFI 2.1
PI 1.0

EDK 1.06:
UEFI 2.1+
PI 1.0

SCT
PI 1.0

EDK II*:
UEFI 2.1+
PI 1.0

UDK2010:
UEFI 2.3
PI 1.2

UDK2014
UEFI 2.4
PI 1.3

http://tianocore.org SourceForge.net

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

Industry BIOS 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

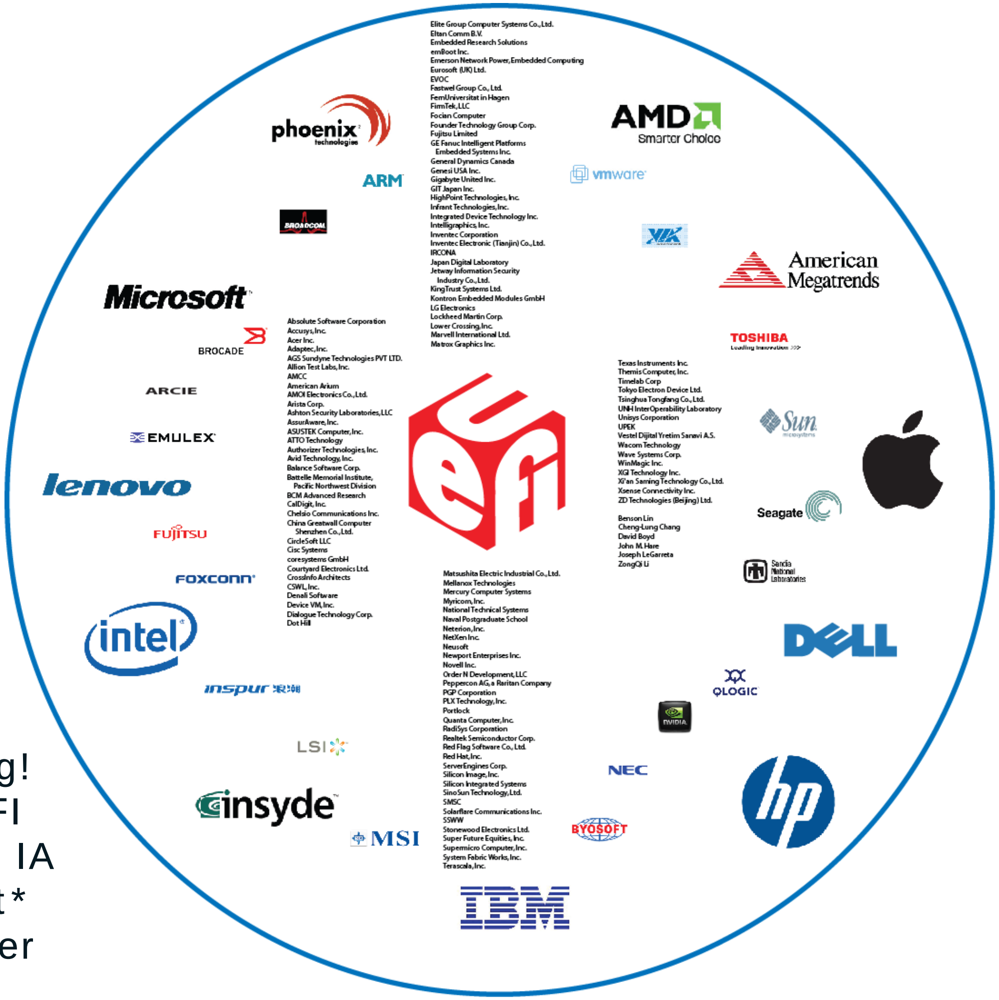
tianocore.org, open source EFI community launched

2005

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

2014

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 SuSEl* OS support. Mandatory for Windows 8 client. ARM 32 and 64 bit support. ACPI added.



Elite Group Computer Systems Co., Ltd.
Eltan Comm B.V.
Embedded Research Solutions
emBoot Inc.
Emerson Network Power, Embedded Computing
Eurosoft (UK) Ltd.
EVOC
Fastwel Group Co., Ltd.
FernUniversitat in Hagen
FirmTek, LLC
Focian Computer
Founder Technology Group Corp.
Fujitsu Limited
GE Fanuc Intelligent Platforms
Embedded Systems Inc.
General Dynamics Canada
Genesi USA Inc.
Gigabyte United Inc.
GIT Japan Inc.
HighPoint Technologies, Inc.
Infrant Technologies, Inc.
Integrated Device Technology Inc.
Intelligraphics, Inc.
Inventec Corporation
Inventec Electronic (Tianjin) Co., Ltd.
IRCONA
Japan Digital Laboratory
Jetway Information Security
Industry Co., Ltd.
KingTrust Systems Ltd.
Kontron Embedded Modules GmbH
LG Electronics
Lockheed Martin Corp.
Lower Crossing, Inc.
Marvell International Ltd.
Matrox Graphics Inc.

Absolute Software Corporation
Accusys, Inc.
Acer Inc.
Adaptec, Inc.
AGS Sundyne Technologies PVT LTD.
Allion Test Labs, Inc.
AMCC
American Arium
AMOI Electronics Co., Ltd.
Arista Corp.
Ashton Security Laboratories, LLC
AssurAware, Inc.
ASUSTEK Computer, Inc.
ATTO Technology
Authorizer Technologies, Inc.
Avid Technology, Inc.
Balance Software Corp.
Battelle Memorial Institute,
Pacific Northwest Division
BCM Advanced Research
CalDigit, Inc.
Chelsio Communications Inc.
China Greatwall Computer
Shenzhen Co., Ltd.
CircleSoft LLC
Cisc Systems
coresystems GmbH
Courtyard Electronics Ltd.
CrossInfo Architects
CSWL, Inc.
Denali Software
Device VM, Inc.
Dialogue Technology Corp.
Dot Hill

Texas Instruments Inc.
Themis Computer, Inc.
Timelab Corp
Tokyo Electron Device Ltd.
Tsinghua Tongfang Co., Ltd.
UNH InterOperability Laboratory
Unisys Corporation
UPEK
Vestel Dijital Yretim Sanavi A.S.
Wacom Technology
Wave Systems Corp.
WinMagic Inc.
XGI Technology Inc.
Xi'an Saming Technology Co., Ltd.
Xsense Connectivity Inc.
ZD Technologies (Beijing) Ltd.

Benson Lin
Cheng-Lung Chang
David Boyd
John M. Hare
Joseph LeGarreta
ZongQi Li

Matsushita Electric Industrial Co., Ltd.
Mellanox Technologies
Mercury Computer Systems
Myricom, Inc.
National Technical Systems
Naval Postgraduate School
Neterion, Inc.
NetXen Inc.
Neusoft
Newport Enterprises Inc.
Novell Inc.
Order N Development, LLC
Peppercon AG, a Raritan Company
PGP Corporation
PLX Technology, Inc.
Portlock
Quanta Computer, Inc.
RadiSys Corporation
Realtek Semiconductor Corp.
Red Flag Software Co., Ltd.
Red Hat, Inc.
ServerEngines Corp.
Silicon Image, Inc.
Silicon Integrated Systems
SinoSun Technology, Ltd.
SMSC
Solarflare Communications Inc.
SSWW
Stonewood Electronics Ltd.
Super Future Equities, Inc.
Supermicro Computer, Inc.
System Fabric Works, Inc.
Terascala, Inc.

How to build UEFI? UDK2014

Industry Standards Compliance

- UEFI 2.0, UEFI 2.1, UEFI 2.2, UEFI 2.3, UEFI2.4; PI 1.0, PI 1.1, PI 1.2, PI1.3, ACPI1.0-5.0

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

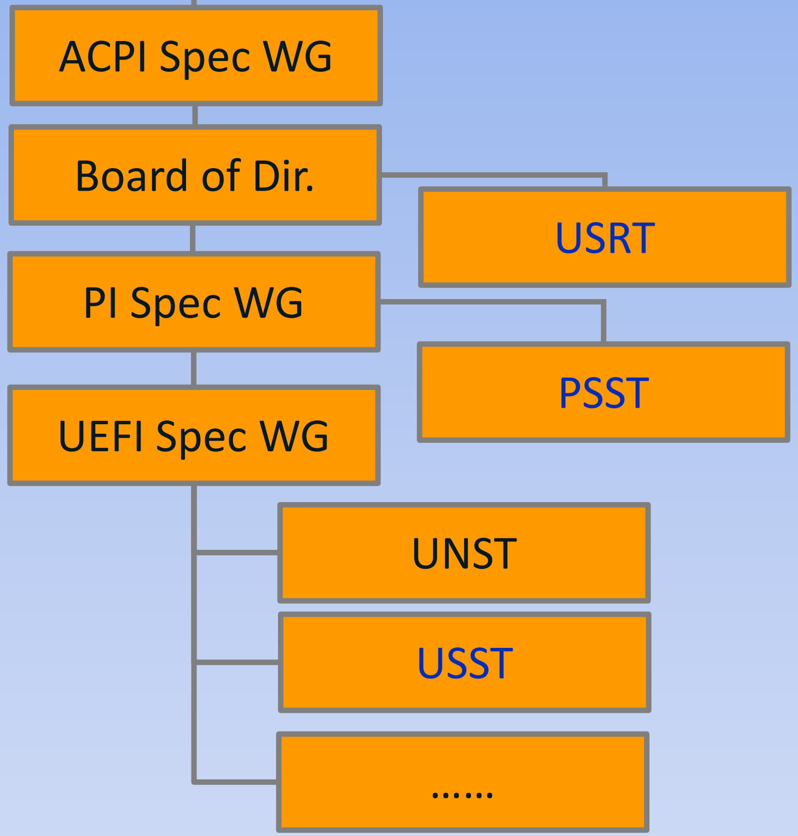
*** benefit of EDK II codebase*

Maximize the open source at www.tianocore.org

https://edk2.sourceforge.net

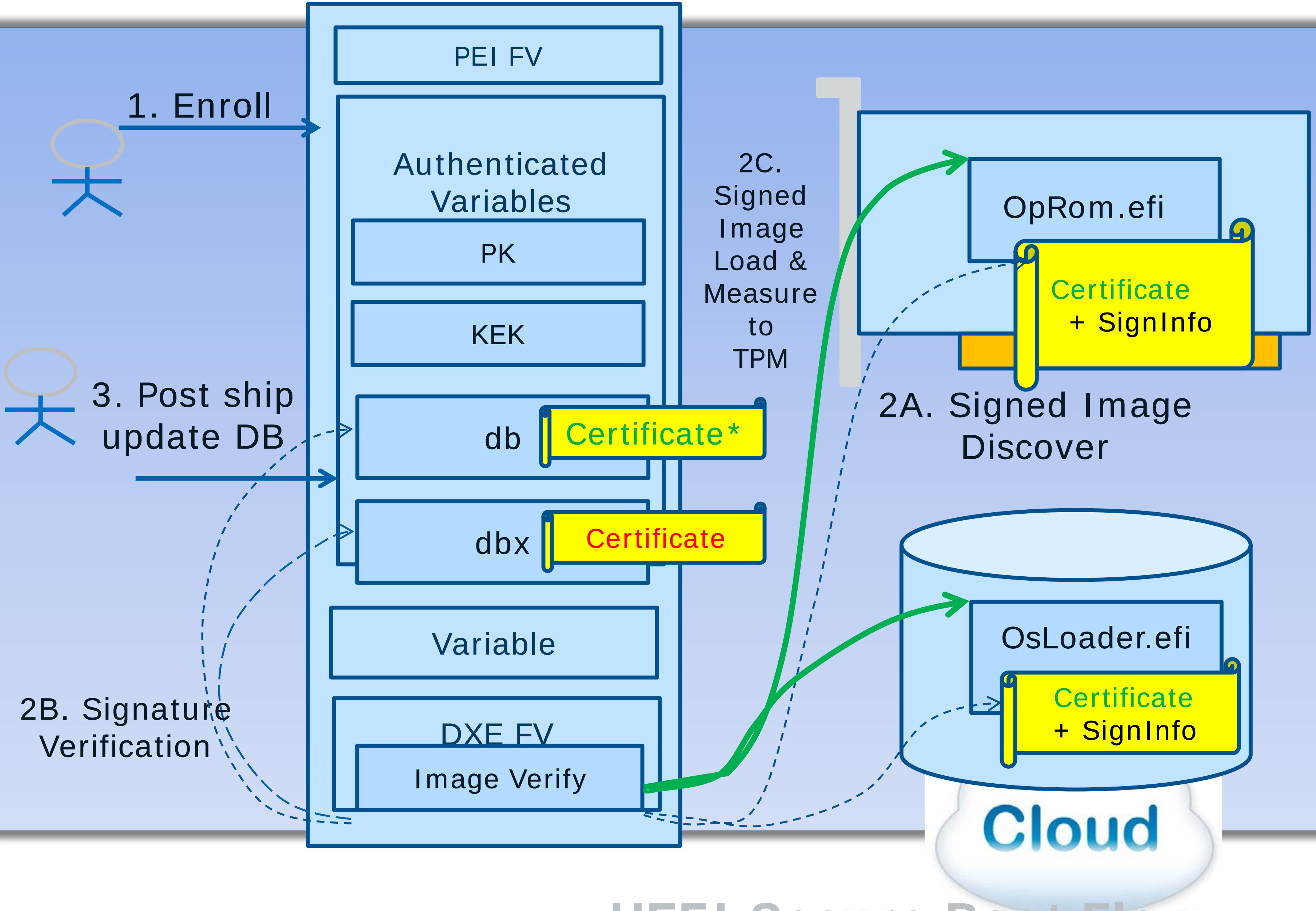
Security Working Groups in UEFI

www.uefi.org



- **UNST**
  - **U**EFI **N**etwork **S**ub-**T**eam
  - Chaired by Vincent Zimmer (Intel)
  - Responsible for evolving network boot – wireless, IPV6, data center, including network security
- **USST**
  - **U**SWG **S**ecurity **S**ub-**t**eam
  - Chaired by Vincent Zimmer (Intel)
  - Responsible for all security related material and the team has been responsible for the added security infrastructure in the UEFI
- **PSST**
  - PIWG Security Sub-team
  - Chaired by Vincent Zimmer (Intel)
  - Produce design guide(s) and security requirements, and identify architectural and implementation issues that cause the requirements not to be met.

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



1. Enroll

PEI FV

Authenticated Variables

PK

KEK

db

Certificate*

dbx

Certificate

Variable

DXE FV

Image Verify

3. Post ship update DB

2B. Signature Verification

2C. Signed Image Load & Measure to TPM

OpRom.efi

Certificate + SignInfo

2A. Signed Image Discover

OsLoader.efi

Certificate + SignInfo

Cloud

UEFI Secure Boot Flow

*Including UEFI CA

Network boot

What about networking in firmware?

- Rationale - How to get to an OS (i.e., boot)
  - Provisioning/installation
  - Diskless client/server nodes
  - Recovery

Today’s practice

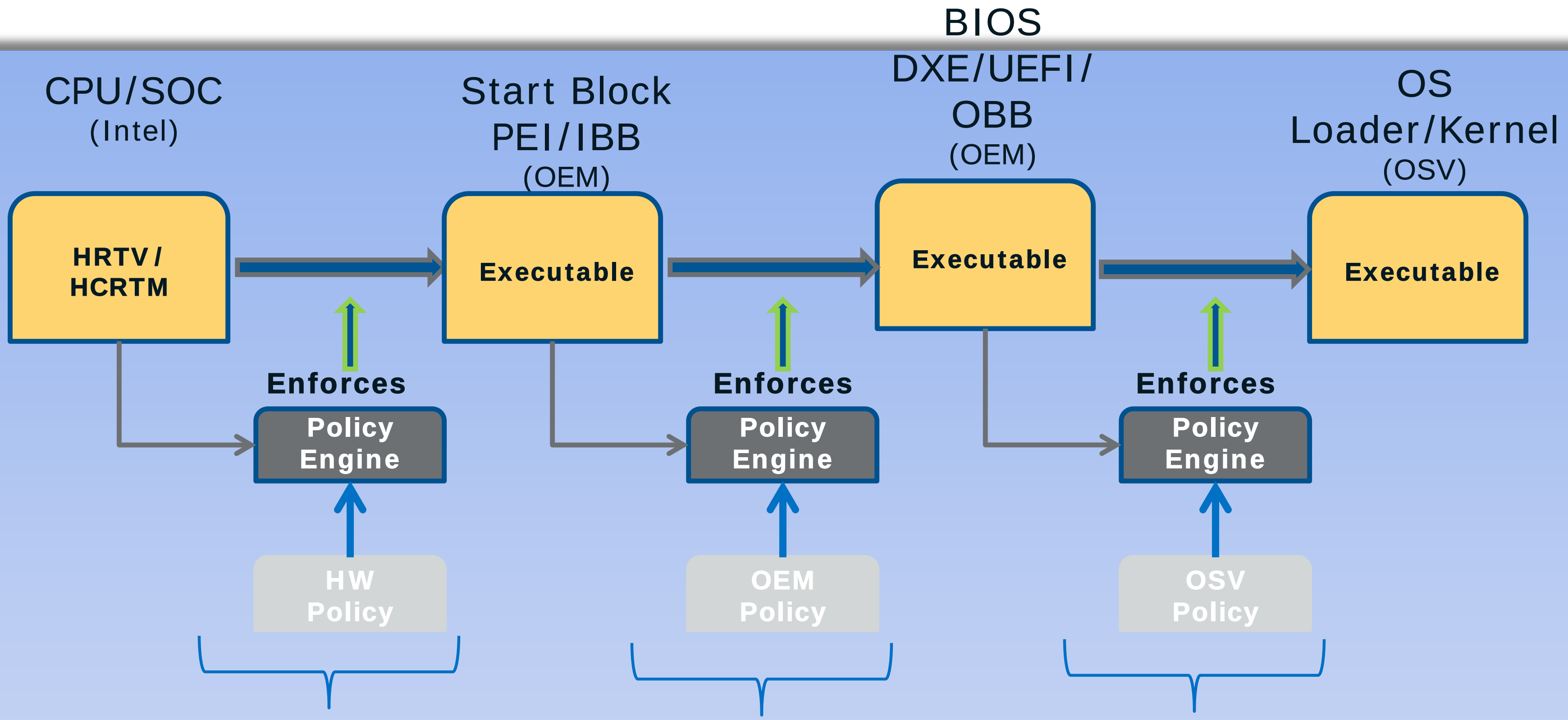
- Pxe2.1/Netboot6 using TFTP in standards, UEFI edk2 network pkg, ipxe, other - closed networks

Moving

- Boot from web-server, wireless, …., the internet

Challenge

- Credentialing, larger attack surface, complexity



CPU/SOC
(Intel)

HRTV / HCRTM

Start Block
PEI/IBB
(OEM)

Executable

BIOS
DXE/UEFI/
OBB
(OEM)

Executable

OS
Loader/Kernel
(OSV)

Executable

Enforces

Policy Engine

HW Policy

Enforces

Policy Engine

OEM Policy

Enforces

Policy Engine

OSV Policy

Hardware Secure Boot, such as "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 or Custom internal chain Maintenance (IPC to Sec, etc)

OEM UEFI 2.4 Secure Boot

Chapter 27.2 of The UEFI 2.4 Specification

Figure 5 of **http://www.uefidk.com/sites/default/files/resources/Platform_Security_Review_Intel_Cisco_White_Paper.pdf**

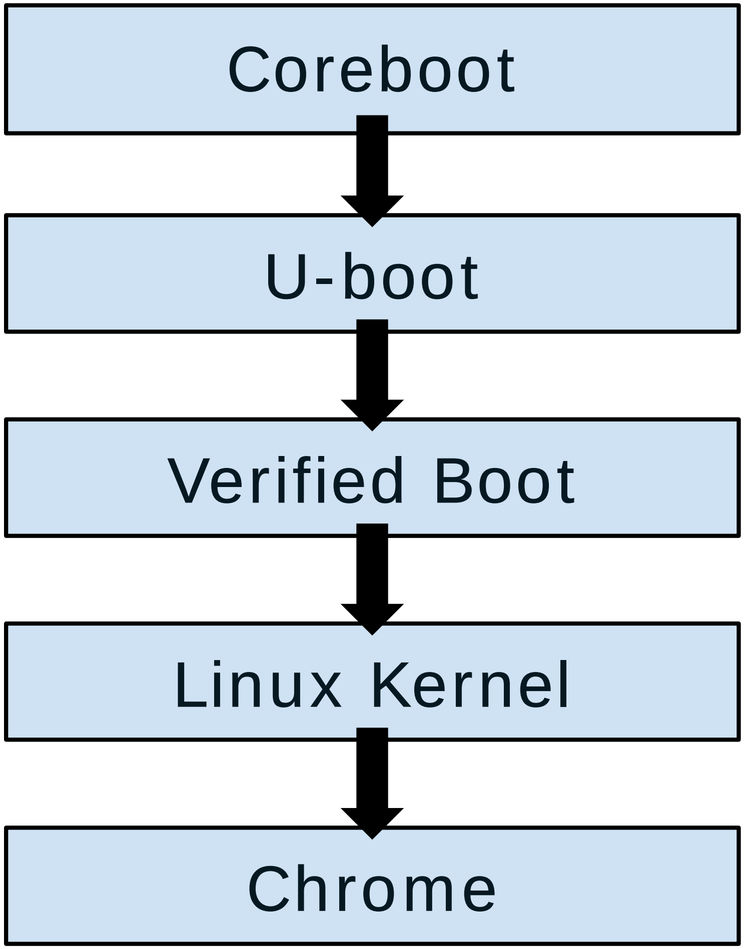
Different flavors of "Secure Boot'

Just UEFI / EDK2?
Also Intel booting via Coreboot for Chromebooks

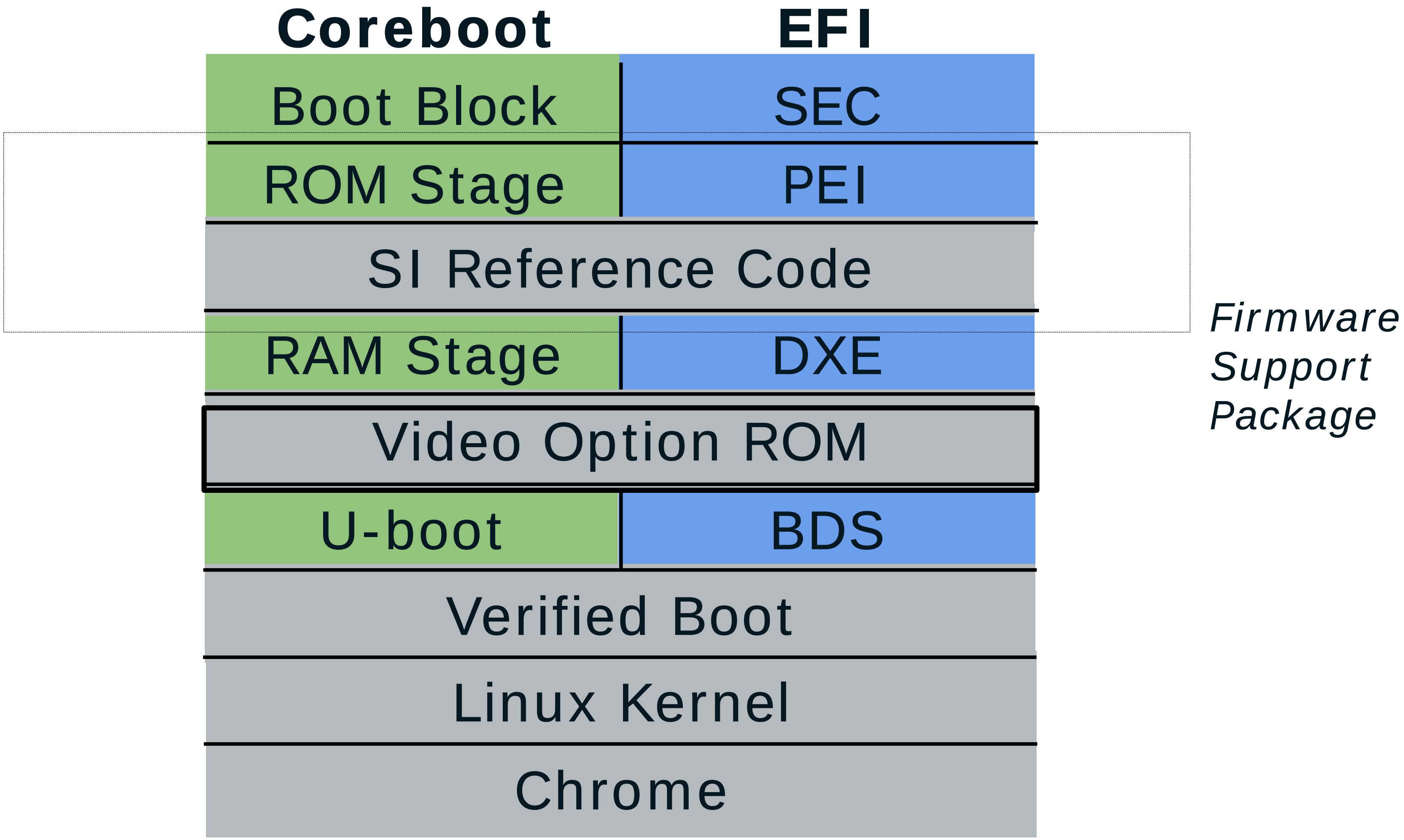
Open

- GPLv2
- Mostly written in C
- Kconfig and modified Kbuild
- High-level organization not too different from EFI
  - Well-defined boot phases
  - Modular CPU, Chipset, Device support
- NOT a bootloader
  - Support for various payloads
  - Payloads can boot Linux, DOS, Windows, etc

Basic Coreboot Boot Flow



Coreboot vs. UEFI



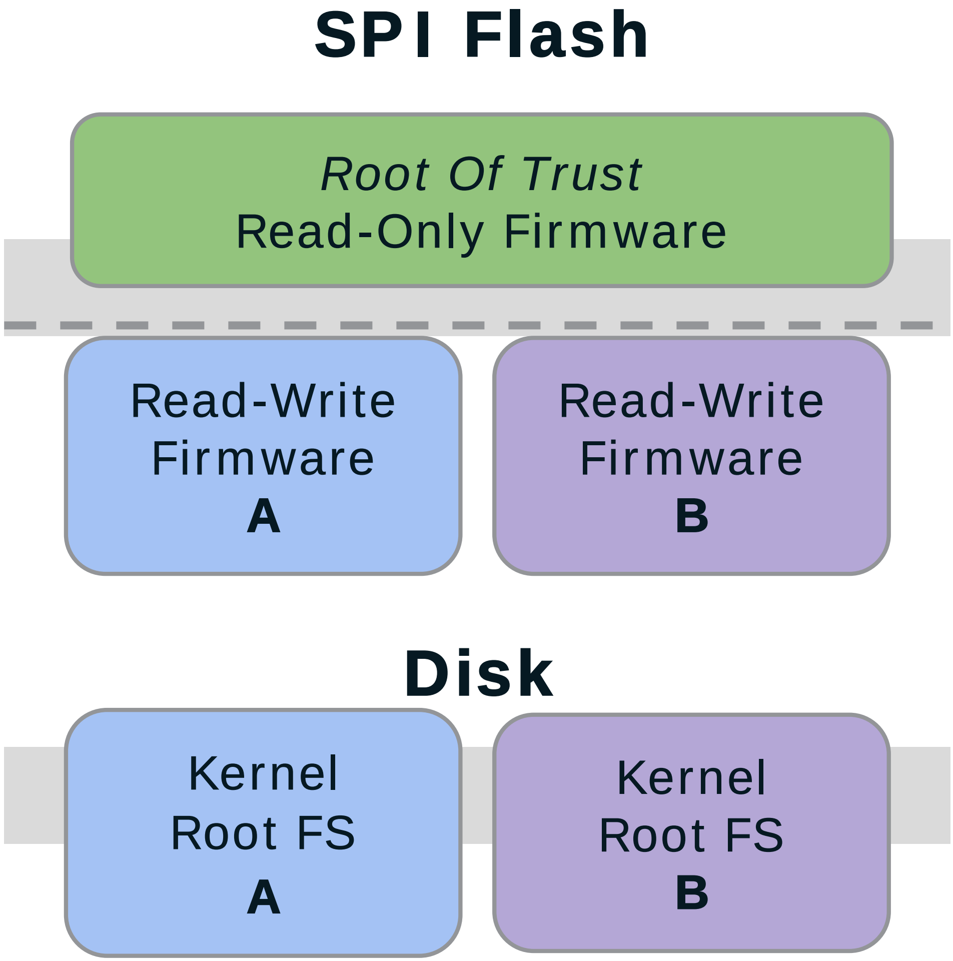
Coreboot	EFI
Boot Block	SEC
ROM Stage	PEI
SI Reference Code	
RAM Stage	DXE
Video Option ROM	
U-boot	BDS
Verified Boot	
Linux Kernel	
Chrome	

Firmware Support Package

Verified Boot - Firmware

- Root Of Trust is in read-only firmware
  - Reset vector must be in RO flash
  - Complicated by SPI Flash Descriptor and ME
- RO firmware can verify signed RW firmware
- Firmware verifies signed kernel from disk
- Reference implementation available
  - chromiumos/platform/vboot_reference.git

Verified Boot - Overview



SPI Flash

Root Of Trust
Read-Only Firmware

Read-Write
Firmware
A

Read-Write
Firmware
B

Disk

Kernel
Root FS
A

Kernel
Root FS
B

Paths to openness

More platforms, more implementations

Intel FSP to build full platforms w/ Open Source IA firmware ecosystems

www.Coreboot.org

www.Tianocore.org

UEFI community at www.uefidk.com

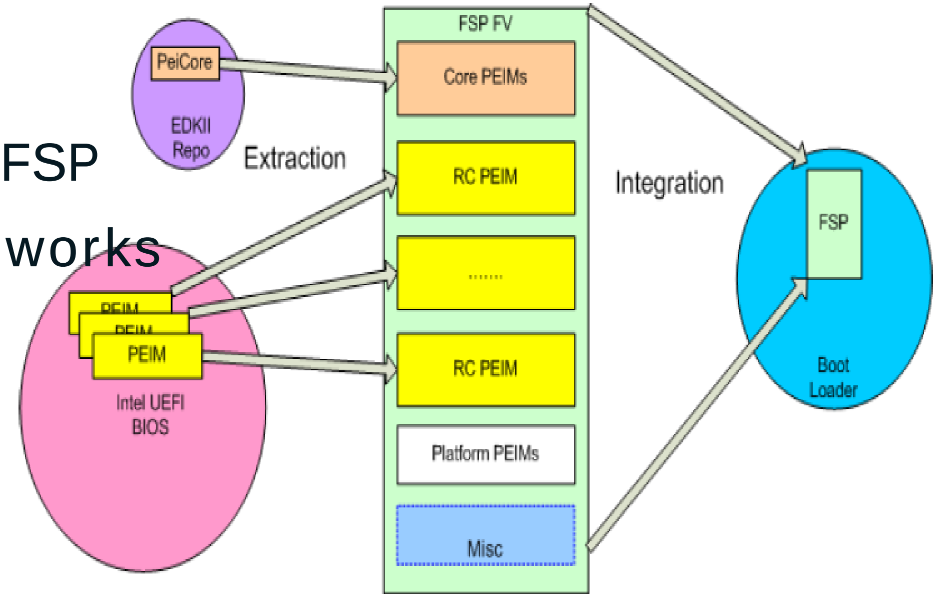
Full platform sources for Intel Quark/Galileo, including feature rich UEFI build (1MByte image) and scaled down “TinyQuark”

Minnowl Atom

Intel® Firmware Support Package (FSP) Overview

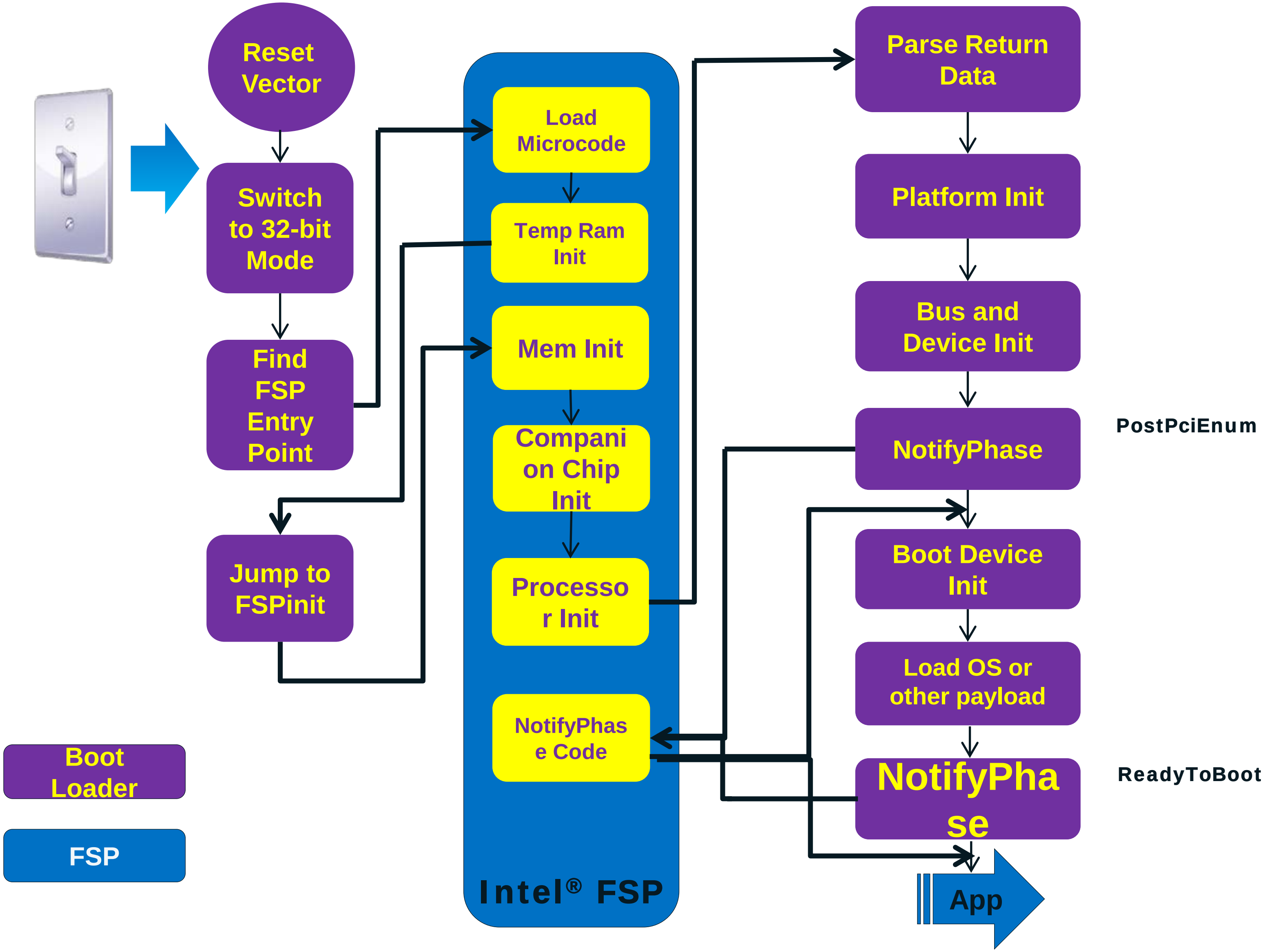
The Intel ® FSP provides processor & chipset initialization in a format that can easily be incorporated into many existing boot loader frameworks without exposing the Intellectual Property (IP) of Intel.

- Distributed as single binary
- Silicon PEIMs packaged into FSP
- Plugs into existing f/w frameworks
- Binary customization



More information at www.intel.com/fsp

Intel® FSP Boot Flow



Reset Vector

Switch to 32-bit Mode

Find FSP Entry Point

Jump to FSPinit

Load Microcode

Temp Ram Init

Mem Init

Companion Chip Init

Processor Init

NotifyPhase Code

Intel® FSP

Parse Return Data

Platform Init

Bus and Device Init

NotifyPhase

PostPciEnum

Boot Device Init

Load OS or other payload

NotifyPhase

ReadyToBoot

App

Boot Loader

FSP

What to build & defend – Rationale for a threat model

“My house is secure” is almost meaningless

- Against a burglar? Against a meteor strike? A thermonuclear device?

“My system is secure” is almost meaningless

- Against what? To what extent?

Threat modeling is a process to define the goals and constraints of a (software) security solution

- Translate user requirements to security requirements

We use threat modeling for firmware codebases

- We believe the process and findings are applicable to driver implementations as well as UEFI implementations in general

Defining, using a threat model

A Threat Model (TM) defines the security assertions and constraints for a product

- Assets: What we're protecting
- Threats: What we're protecting it against
- Mitigations: How we're protecting our Assets

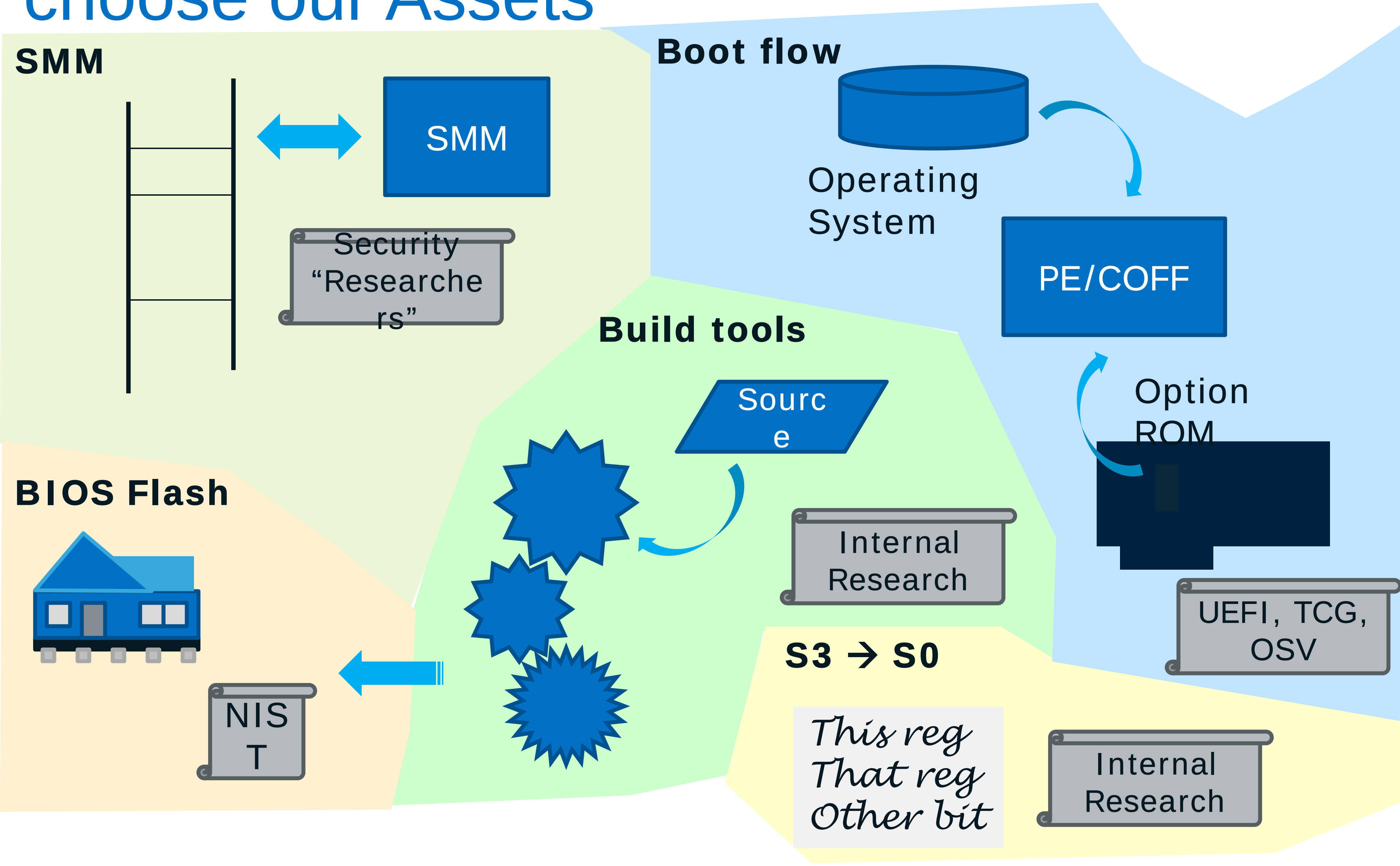
Use TM to narrow subsequent mitigation efforts

- Don't secure review, fuzz test all interfaces
- Select the ones that are critical

TM is part science, part art, part experience, part nuance, part preference

- Few big assets vs lots of focused assets

We don’t always get to choose our Assets



SMM

SMM

Security “Researchers”

Boot flow

Operating System

PE/COFF

Option ROM

UEFI, TCG, OSV

Build tools

Source

Internal Research

BIOS Flash

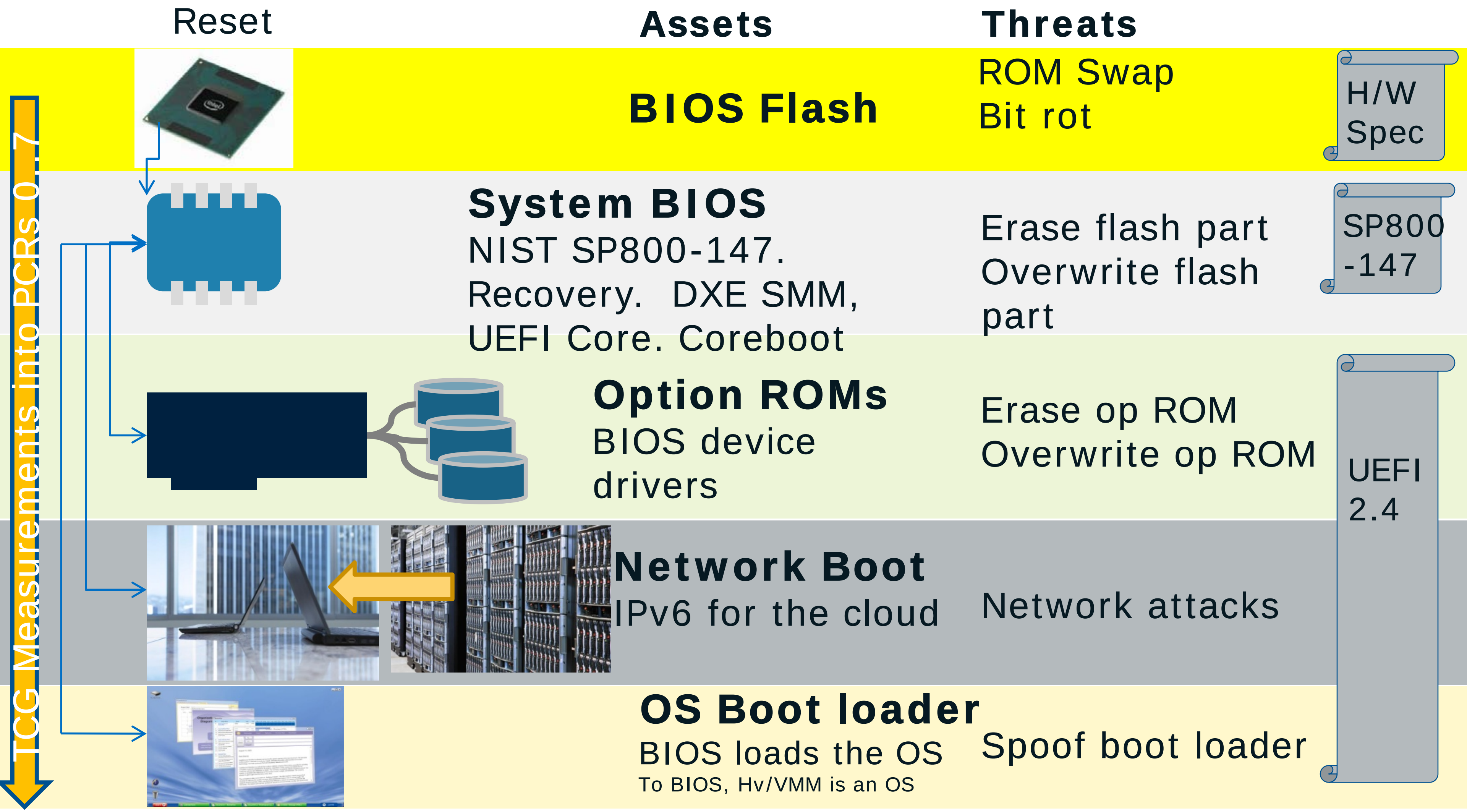
NIST

S3 → S0

This reg
That reg
Other bit

Internal Research

Technologies – putting it together

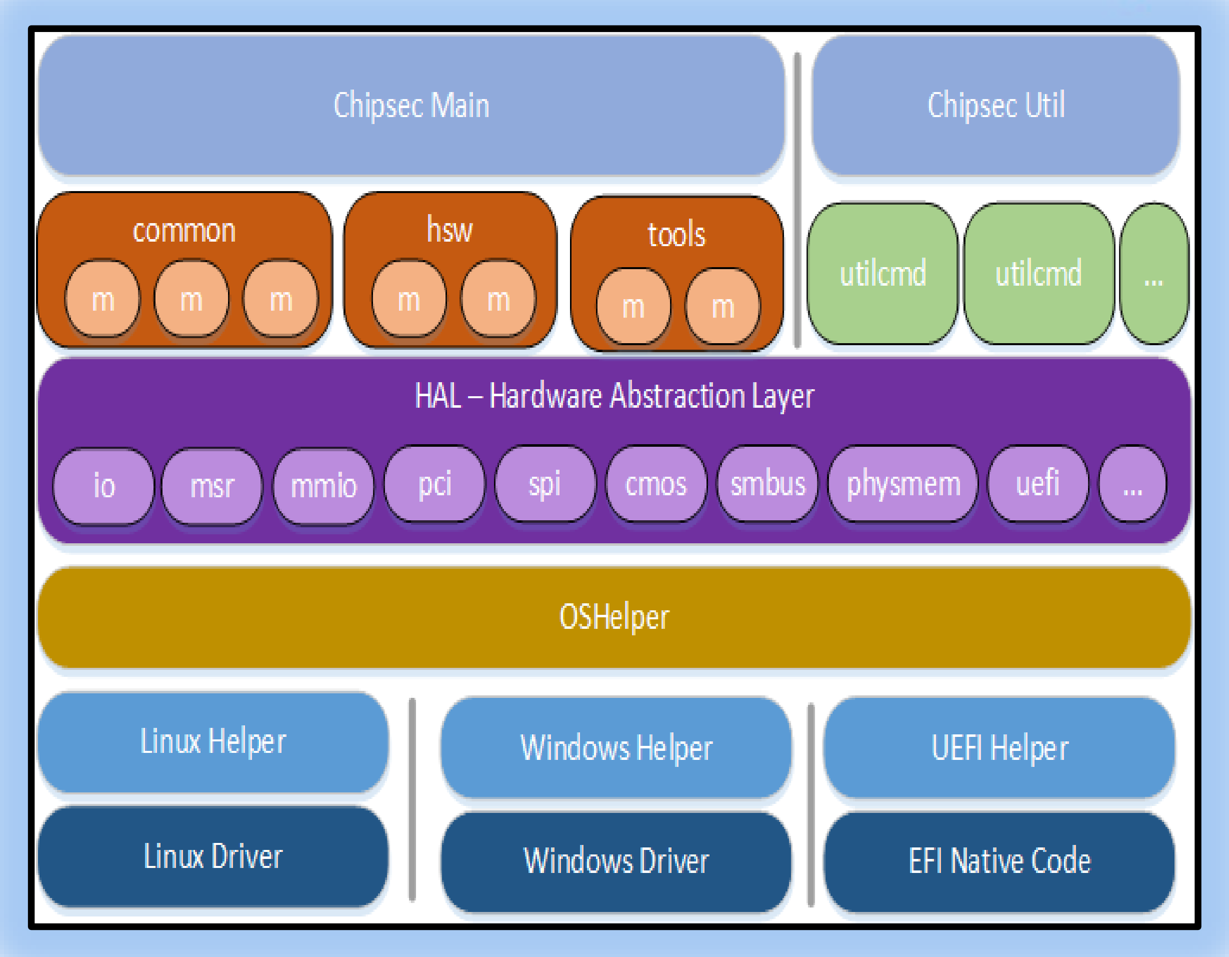


Assets	Threats	
BIOS Flash	ROM Swap Bit rot	H/W Spec
System BIOS NIST SP800-147. Recovery. DXE SMM, UEFI Core. Coreboot	Erase flash part Overwrite flash part	SP800-147
Option ROMs BIOS device drivers	Erase op ROM Overwrite op ROM	UEFI 2.4
Network Boot IPv6 for the cloud	Network attacks	
OS Boot loader BIOS loads the OS To BIOS, Hv/VMM is an OS	Spoof boot loader	

Different colors for different vendors ⬜ 🟩 ⬛ 🟨 🟨

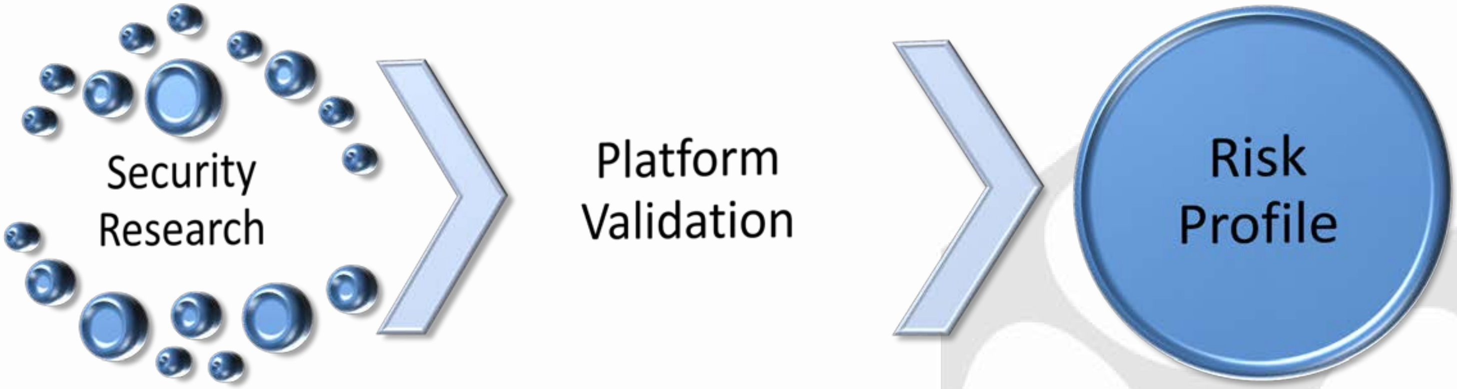
chipsec - Platform Security Assessment Framework

A single test designed to run in multiple environments



https://github.com/chipsec/chipsec

How do we raise the bar?



Security Research

Platform Validation

Risk Profile

New Attacks

Test Modules for OEMs/IBVs

End-user Risk

Empowering End-Users to Make a Risk Decision

Known Threats and CHIPSEC modules

Issue	CHIPSEC Module	Public Details
SMRAM Locking	common.smm	CanSecWest 2006
BIOS Keyboard Buffer Sanitization	common.bios_kbrd_buffer	DEFCON 16 2008
SMRR Configuration	common.smrr	ITL 2009 CanSecWest 2009
BIOS Protection	common.bios_wp	BlackHat USA 2009 CanSecWest 2013 Black Hat 2013 NoSuchCon 2013 Flashrom
SPI Controller Locking	common.spi_lock	Flashrom Copernicus
BIOS Interface Locking	common.bios_ts	PoC 2007
Access Control for Secure Boot Keys	common.secureboot.keys	CanSecWest 2014
Access Control for Secure Boot Variables	common.secureboot.variables	HITB 2014

Example: BIOS Write Protection

Black Hat USA 2013 “BIOS Security” – MITRE (Kovah, Butterworth, Kallenberg)
NoSuchCon 2013 “BIOS Chronomancy: Fixing the Static Root of Trust for Measurement” – MITRE (Kovah, Butterworth, Kallenberg)

Is BIOS correctly protected?

common.bios_wp

```
[+] imported chipsec.modules.common.bios_wp
[x][ =======================================================================
[x][ Module: BIOS Region Write Protection
[x][ =======================================================================
BIOS Control (BDF 0:31:0 + 0xDC) = 0x2A
[05]    SMM_BWP = 1 (SMM BIOS Write Protection)
[04]    TSS     = 0 (Top Swap Status)
[01]    BLE     = 1 (BIOS Lock Enable)
[00]    BIOSWE  = 0 (BIOS Write Enable)

[+] BIOS region write protection is enabled (writes restricted to SMM)

[*] BIOS Region: Base = 0x00500000, Limit = 0x00FFFFFF
SPI Protected Ranges
------------------------------------------------------------
PRx (offset) | Value    | Base     | Limit    | WP? | RP?
------------------------------------------------------------
PR0 (74)     | 00000000 | 00000000 | 00000000 | 0   | 0
PR1 (78)     | 8FFF0F40 | 00F40000 | 00FFF000 | 1   | 0
PR2 (7C)     | 8EDF0EB1 | 00EB1000 | 00EDF000 | 1   | 0
PR3 (80)     | 8EB00EB0 | 00EB0000 | 00EB0000 | 1   | 0
PR4 (84)     | 8EAF0C00 | 00C00000 | 00EAF000 | 1   | 0

[!] SPI protected ranges write-protect parts of BIOS region (other parts of BIOS can be
modified)

[+] PASSED: BIOS is write protected
```

Direct HW Access for Manual Testing

```
Examples:
      chipsec_util msr 0x200
      chipsec_util mem 0x0 0x41E 0x20
      chipsec_util pci enumerate
      chipsec_util pci 0x0 0x1F 0x0 0xDC byte
      chipsec_util io 0x61 byte
      chipsec_util mmcfg 0 0x1F 0 0xDC 1 0x1
      chipsec_util cmos dump
      chipsec_util ucode id
      chipsec_util smi 0x01 0xFF
      chipsec_util idt 0
      chipsec_util cpuid 1
      chipsec_util spi read 0x700000 0x100000
bios.bin
      chipsec_util decode spi.bin
      chipsec_util uefi var-list
      ..
```

Forensics

Live system firmware analysis

```
    chipsec_util spi info
    chipsec_util spi dump rom.bin
    chipsec_util spi read 0x700000 0x100000 bios.bin
    chipsec_util uefi var-list
    chipsec_util uefi var-read db D719B2CB-3D3A-4596-A3BC-DAD00E67656F db.bin
```

Offline system firmware analysis

```
    chipsec_util uefi keys PK.bin
    chipsec_util uefi nvram vss bios.bin
    chipsec_util uefi decode rom.bin
    chipsec_util decode rom.bin
```

Moving Forward

Test tools complement the SCT, but **the community can do more!**

Changing our development philosophy?

- "Testing shows the presence, not the absence of bugs" (*Dijkstra,1970*)
- Better Living Through Tools? (*Zimmer, 2013*)

Getting code coverage closer to 100%?

- Internal Intel effort using DDT with EDK II
- Moving to KLEE (open source)

"Infrastructure for automatic code checking" (coreboot)

- Automated system including KLEE, Splint, Frama-C

Summary

- Threats of firmware attacks & UEFI extensibility are real
- Address w/ open standards and open source
- Secure boot is here
- Platforms under attack
- More focus on implementation, less on feature
- Continue to open, open, open

For more information - UEFI Secure Boot

Intel Technology Journal, Volume 15, Issue 1, 2011, UEFI Today: Bootstrapping the Continuum, UEFI Networking and Pre-OS Security, page 80 at ITJ Secure Boot

Rosenbaum, Zimmer, "A Tour Beyond BIOS into UEFI Secure Boot," Intel Corporation, July 2012 http://sourceforge.net/projects/edk2/files/General%20Documentation/A_Tour_Beyond_BIOS_into_UEFI_Secure_Boot_White_Paper.pdf/download

UEFI 2.3.1 specification: Sections 7.2 (Variable Services) and Sections 27.2 through 27.8 (Secure Boot) of the at www.uefi.org

Beyond BIOS: Developing with the Unified Extensible Firmware Interface, 2nd Edition, Zimmer, et al, ISBN 13 978-1-934053-29-4, Chapter 10 – Platform Security and Trust, http://www.intel.com/intelpress

“Hardening the Attack Surfaces,” MSFT 2012 UEFI Plugfest http://www.uefi.org/learning_center/UEFI_Plugfest_2012Q1_Microsoft_AttackSurface.pdf

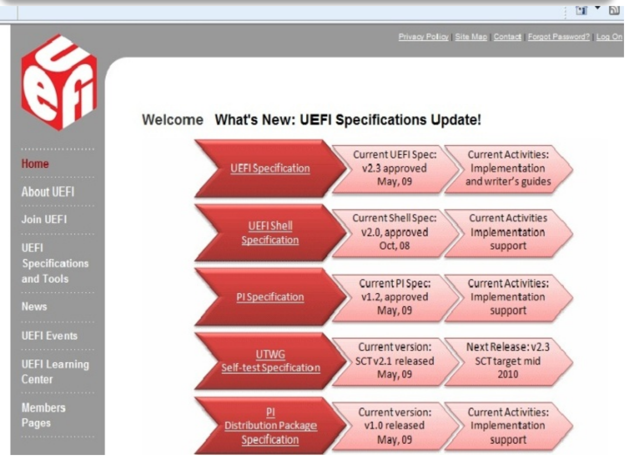
“Building hardware-based security with a TPM” MSFT BUILD http://channel9.msdn.com/Events/BUILD/BUILD2011/HW-462T

Lin, Oswald, Zimmer, “UEFI Secure Boot in Linux,” Intel Developer Forum, San Francisco, September 11, 2013 https://intel.activeevents.com/sf13/connect/fileDownload/session/A25811835C1B6573651FC73FB20D0F6C/SF13_STTS002_100.pdf

A Tale of One Software Bypass of Windows 8 Secure Boot by Andrew Furtak, Oleksandr Bazhaniuk and Yuriy Bulygin, Blackhat 2013

UEFI Industry Resources

UEFI Forum



Privacy Policy | Site Map | Contact | Forgot Password? | Log On

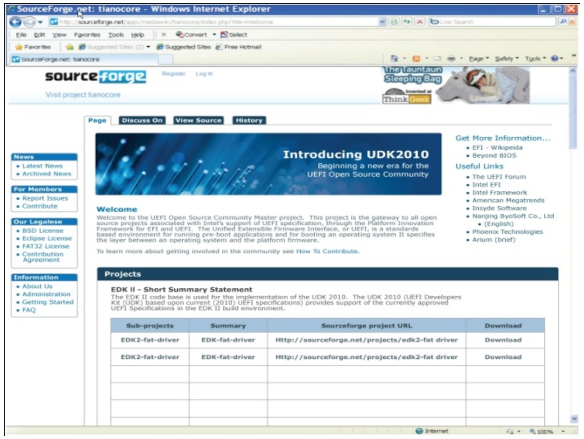
- Home
- About UEFI
- Join UEFI
- UEFI Specifications and Tools
- News
- UEFI Events
- UEFI Learning Center
- Members Pages

Welcome **What's New: UEFI Specifications Update!**

UEFI Specification	Current UEFI Spec: v2.3 approved May, 09	Current Activities: Implementation and writer's guides
UEFI Shell Specification	Current Shell Spec: v2.0, approved Oct, 08	Current Activities Implementation support
PI Specification	Current PI Spec: v1.2, approved May, 09	Current Activities: Implementation support
UTWG Self-test Specification	Current version: SCT v2.1 released May, 09	Next Release: v2.3 SCT target mid 2010
PI Distribution Package Specification	Current version: v1.0 released May, 09	Current Activities: Implementation support

www.uefi.org

UEFI Open Source



SourceForge.net: tianocore - Windows Internet Explorer

http://sourceforge.net/apps/mediawiki/tianocore/index.php?title=Welcome

File Edit View Favorites Tools Help
Convert Select
Favorites Suggested Sites (2) Suggested Sites Free Hotmail
SourceForge.net: tianocore
Page Safety Tools
sourceforge
Register Log In
Visit project tianocore
The Tauntaun Sleeping Bag
Invented at ThinkGeek
Page
Discuss On
View Source
History

News
- Latest News
- Archived News

For Members
- Report Issues
- Contribute

Our Legalese
- BSD License
- Eclipse License
- FAT32 License
- Contribution Agreement

Information
- About Us
- Administration
- Getting Started
- FAQ

Introducing UDK2010
Beginning a new era for the UEFI Open Source Community

Welcome

Welcome to the UEFI Open Source Community Master project. This project is the gateway to all open source projects associated with Intel's support of UEFI specification, through the Platform Innovation Framework for EFI and UEFI. The Unified Extensible Firmware Interface, or UEFI, is a standards based environment for running pre-boot applications and for booting an operating system It specifies the layer between an operating system and the platform firmware.

To learn more about getting involved in the community see How To Contribute.

Get More Information...
- EFI - Wikipeida
- Beyond BIOS

Useful Links
- The UEFI Forum
- Intel EFI
- Intel Framework
- American Megatrends
- Insyde Software
- Nanjing ByoSoft Co., Ltd
  - (English)
- Phoenix Technologies
- Arium (brief)

Projects

EDK II - Short Summary Statement

The EDK II code base is used for the implementation of the UDK 2010. The UDK 2010 (UEFI Developers Kit (UDK) based upon current (2010) UEFI specifications) provides support of the currently approved UEFI Specifications in the EDK II build environment.

Sub-projects	Summary	Sourceforge project URL	Download
EDK2-fat-driver	EDK-fat-driver	Http://sourceforge.net/projects/edk2-fat driver	Download
EDK2-fat-driver	EDK-fat-driver	Http://sourceforge.net/projects/edk2-fat driver	Download

www.tianocore.org

Intel UEFI Resources

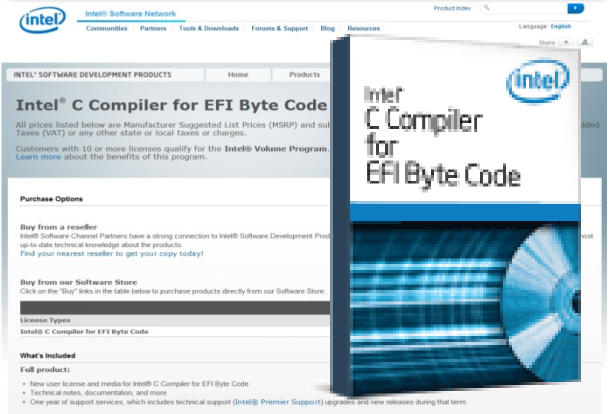
Intel® UDK Community Resource Center

Welcome to Intel® UDK Community Resource Center

Learn more about UEFI ›

www.intel.com/UDK

Intel EBC Compiler



Intel® Software Network
Product Index
Communities | Partners | Tools & Downloads | Forums & Support | Blog | Resources
Language: English
Share
INTEL® SOFTWARE DEVELOPMENT PRODUCTS
Home
Products
Intel C Compiler for EFI Byte Code

Intel® C Compiler for EFI Byte Code

All prices listed below are Manufacturer Suggested List Prices (MSRP) and sub… …dded Taxes (VAT) or any other state or local taxes or charges.

Customers with 10 or more licenses qualify for the **Intel® Volume Program.** Learn more about the benefits of this program.

Purchase Options

Buy from a reseller
Intel® Software Channel Partners have a strong connection to Intel® Software Development Prod… …ost up-to-date technical knowledge about the products.
Find your nearest reseller to get your copy today!

Buy from our Software Store
Click on the "Buy" links in the table below to purchase products directly from our Software Store.

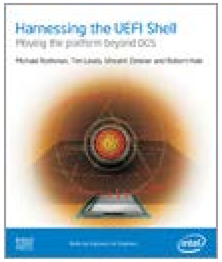
License Types
Intel® C Compiler for EFI Byte Code

What's Included

Full product:
- New user license and media for Intel® C Compiler for EFI Byte Code
- Technical notes, documentation, and more
- One year of support services, which includes technical support (Intel® Premier Support) upgrades and new releases during that term

http://software.intel.com/en-us/articles/intel-c-compiler-for-efi-byte-code-purchase/

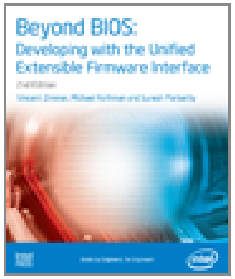
UEFI Books/ Collateral



Harnessing the UEFI Shell

Moving the platform beyond DOS

Michael Rothman, Tim Lewis, Vincent Zimmer and Robert Hale

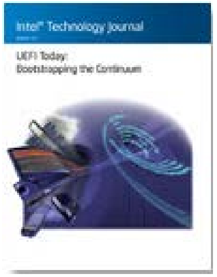


Beyond BIOS:

Developing with the Unified Extensible Firmware Interface

2nd Edition

Vincent Zimmer, Michael Rothman and Suresh Marisetty



Intel® Technology Journal

UEFI Today: Bootstrapping the Continuum

www.intel.com/intelpress

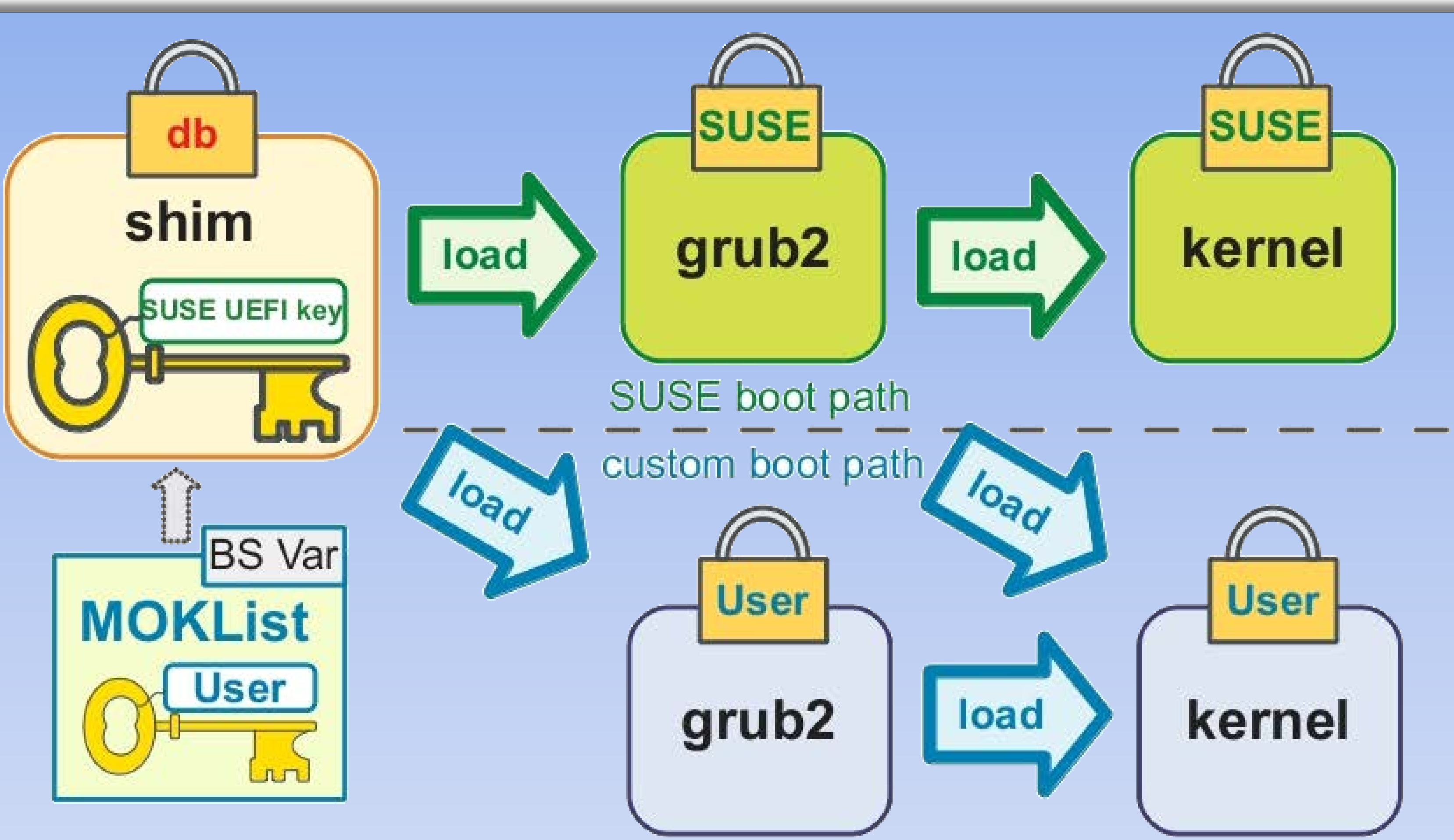
http://www.intel.com/technology/itj/2011/v15i1/index.htm

ToorCamp

Thank You

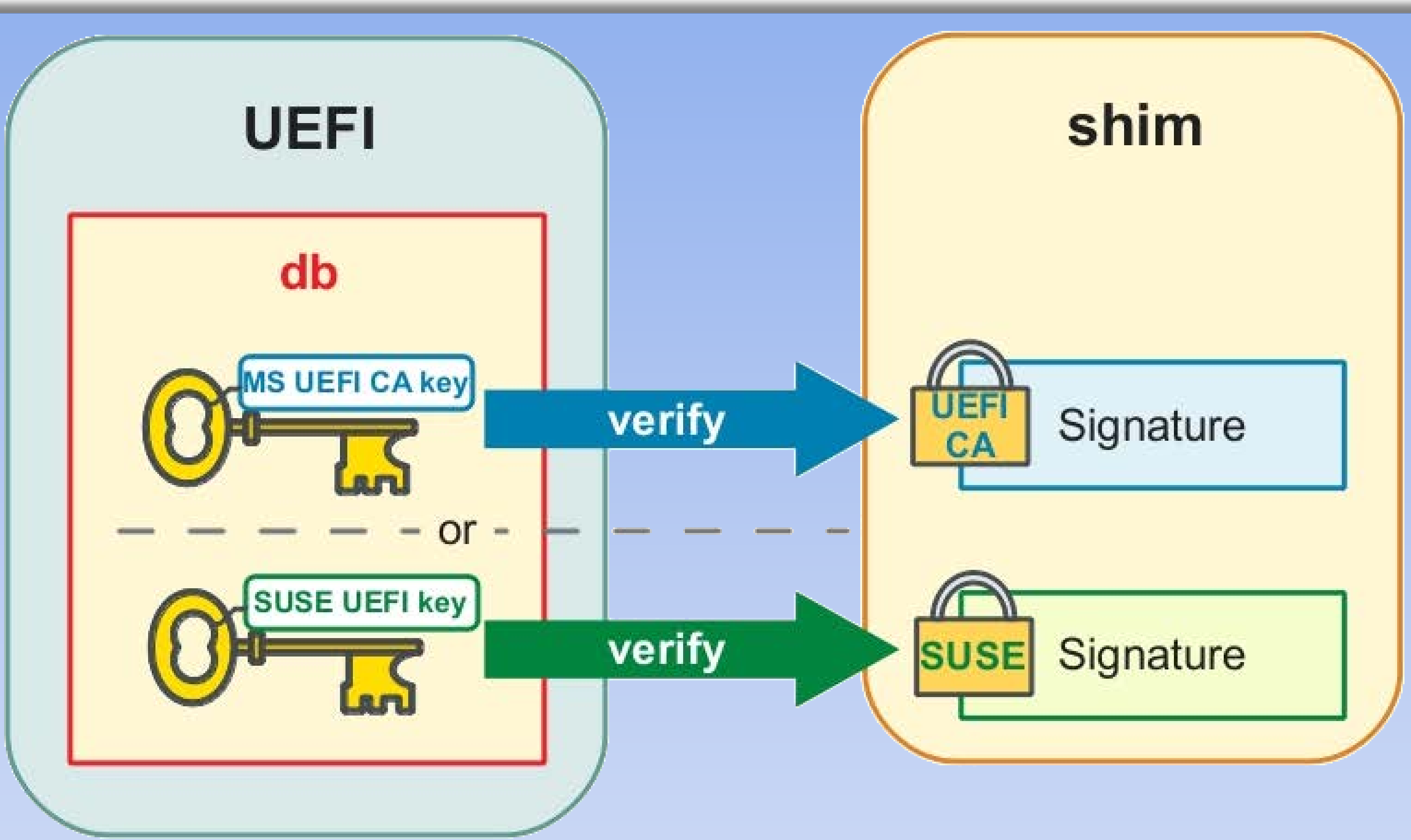
Contact:
vincent.zimmer@gmail.com
@vincentzimmer

BACKUP



Load the UEFI image as long as it is trusted

Linux Update – Multiple OS Boot with MOK



Either the UEFI CA key or SUSE key will let the shim boot with UEFI secure boot

Multi-Signature Support for Shim

RandomNumberGenerator

UEFI driver implementing the EFI_RNG_PROTOCOL from the UEFI2.4 specification

TCG

PEI Modules & DXE drivers implementing Trusted Computing Group measured boot

EFI_TCG_PROTOCOL and EFI_TREE_PROTOCOL from the TCG and Microsoft MSDN websites, respectively

UserIdentification

DXE drivers that support multi-factor user authentication

Chapter 31 of the UEFI 2.4 specification

Library

DxeVerificationLib for “UEFI Secure Boot”, chapter 27.2 of the UEFI 2.4 specification + other support libs

VariableAuthenticated

SMM and runtime DXE authenticated variable driver, chapter 7 of the UEFI2.4 specification

https://svn.code.sf.net/p/edk2/code/trunk/edk2/SecurityPkg

UDK2014 SecurityPkg

Flash**

NIST SP800-147 says

- Lock code flash except for update before Exit Mfg Auth
- Signed update (>= RSA2048, SHA256)
- High quality signing servers
- Without back doors (“non-bypassability”)

Threats

- PDOS – Permanent Denial of Service
  - System into inefficient room heater
- Elevation of privilege
  - Owning the system at boot is an advantage to a virus

Known attacks

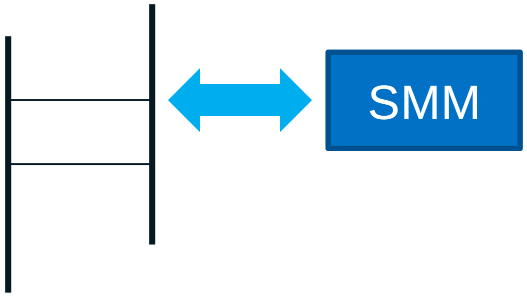
- CIH / Chernobyl 1999-2000
- Mebroni 2010

Mitigations include

- Reexamining flash protection methods – use the best even if its new
- Using advanced techniques to locate and remove (un)intentional backdoors

** or tomorrow’s equivalent NV storage

SMM



SMM is valuable because

- It's invisible to Anti Virus, etc
- SMM sees all of system RAM
- Not too different from PCI adapter device firmware

Threats

- Elevation
  - View secrets or own the system by subverting RAM

Known attacks

- See e.g Duflot

Mitigations include

- Validate "external" / "untrusted" input
- Remove calls from inside SMM to outside SMM

Resume from S3

This reg
That reg
Other bit

ACPI says that we return the system to the S5→S0 configuration at S3→S0

- Must protect the data structures we record the cold boot config in

Threats

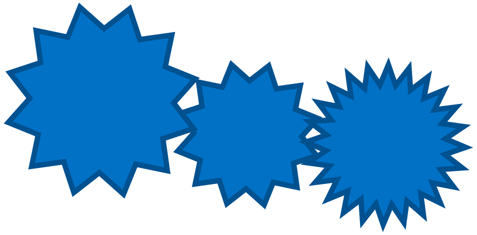
- Changing data structures could cause security settings to be incorrectly configured leaving S3
- Reopen the other assets’ mitigated threats

No known attacks

Mitigations include

- Store data in SMM -or-
- Store hash of data structures and refuse to resume if the hashes don’t compare

Tool chain



Tools create the resulting firmware

- Rely on third party tools and home grown tools
- Incorrect or attacked tools leave vulnerabilities

Threats

- Disabled signing, for example

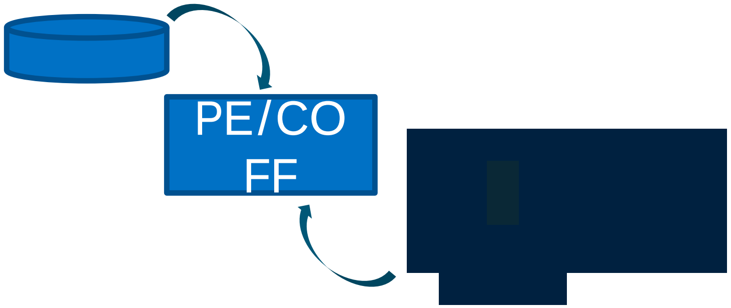
Known attacks

- See e.g. *Reflections on Trust*, Ken Thompson**

Mitigation

- Difficult: For most tools, provided as source code
- Review for correct implementation
- Use static, dynamic code analysis tools
  - PyLint for Python, for example

** CACM, Vol 27, No 8, Aug, 1984, pp. 761-763



Boot flow

Secure boot

- Authenticated variables
- Based on the fundamental Crypto being correct
- Correct location for config data

Threats

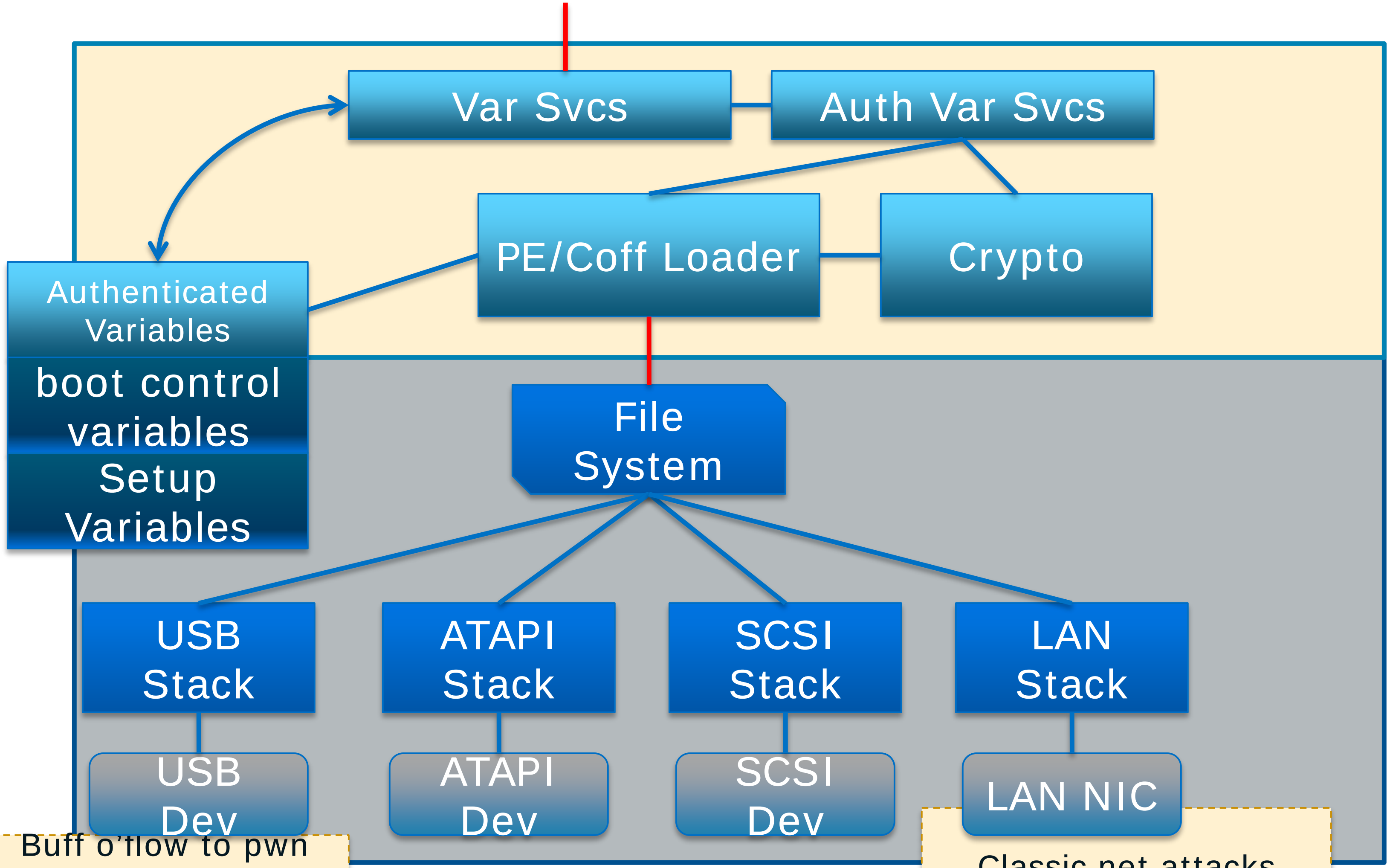
- Run unauthorized op roms, boot loaders
- PDOS systems with bad config variables

Known attacks

Mitigations include

- Sanity check config vars before use, use defaults
- Reviews, fuzz checking, third party reviews, etc.

TM to Modules: Boot flow



Buff o'flow to pwn sys.
Sanity check config pkts.

Classic net attacks
Fuzz test packets

Assets or not?

Variable content sanity checking?

- If you randomly fill in your Setup variables, will your system still boot?
- Fit in as a part of boot flow

ACPI? We create it but don't protect it

TPM support? We fill in the PCRs but don't use them (today)

Quality ≠ Security

Analyze and Mark external Interfaces where input can be attacker controlled data, comment headers

```
/**
  Install child handles if the Handle supports GPT partition structure.

  Caution: This function may receive untrusted input.
  The GPT partition table is external input, so this routine
  will do basic validation for GPT partition table before install
  child handle for each GPT partition.

  @param[in]  This       Calling context.
  @param[in]  Handle     Parent Handle.
  @param[in]  DevicePath Parent Device Path.

**/
EFI_STATUS
PartitionInstallGptChildHandl
```

UDK2010 example:
http://edk2.svn.sourceforge.net/svnroot/edk2/trunk/edk2/MdeModulePkg/Universal/Disk/PartitionDxe/Gpt.c