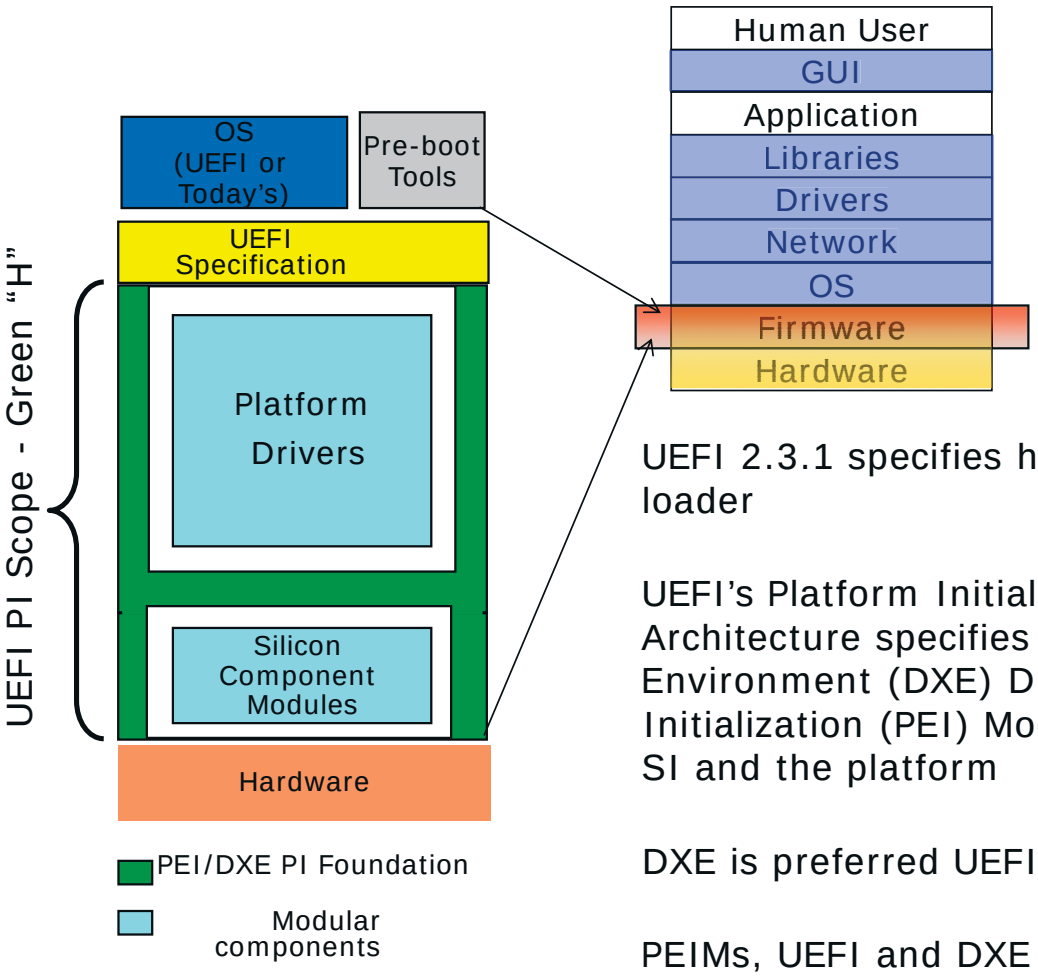
ToorCamp 2012

Secure Boot Ecosystem Challenges

Vincent Zimmer

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

What is UEFI? UEFI Platform Initialization Overview



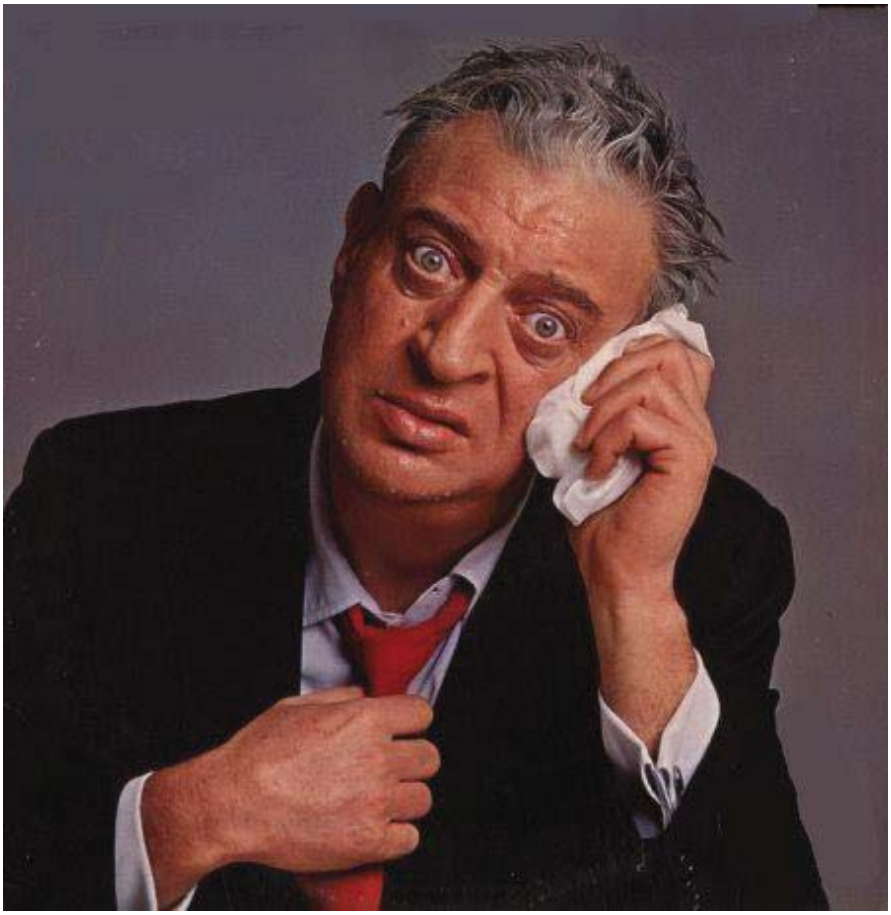
UEFI 2.3.1 specifies how firmware boots OS loader

UEFI's Platform Initialization (PI) 1.2 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 implements networking, Update, other security features

UEFI / PI is a type of BIOS
BIOS- aka. the Rodney Dangerfield of Software



"No respect"

How to build it? UDK2010

Industry Standards Compliance
- UEFI 2.0, UEFI 2.1, UEFI 2.2, UEFI 2.3; PI 1.0, PI 1.1, PI 1.2

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

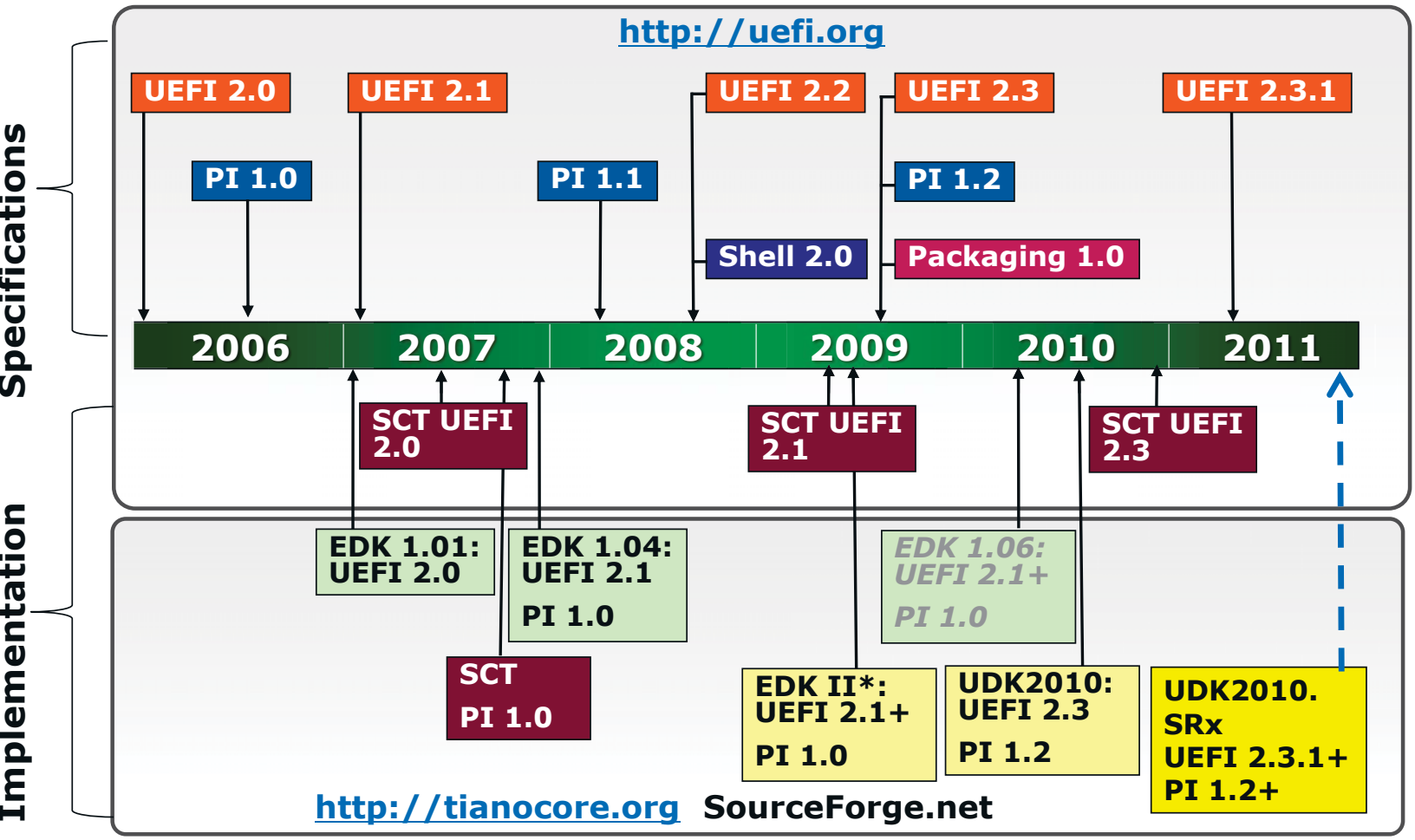
Multiple Development Environments and Tool Chains**
- Windows, Linux, OSX
- VS2003, VS2005, WinDDK, Intel, GCC

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

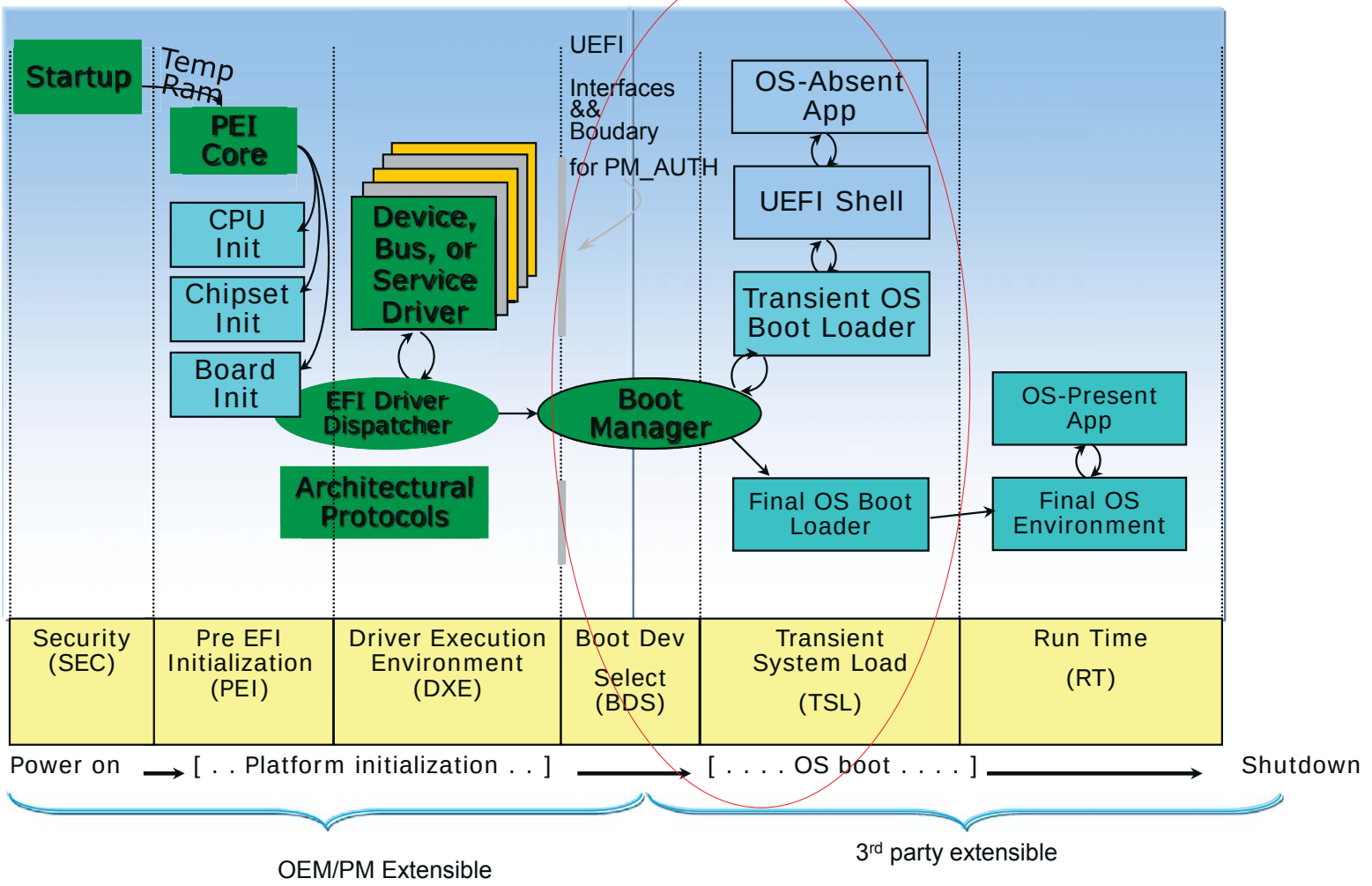
Specification & Tianocore.org Timeline



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

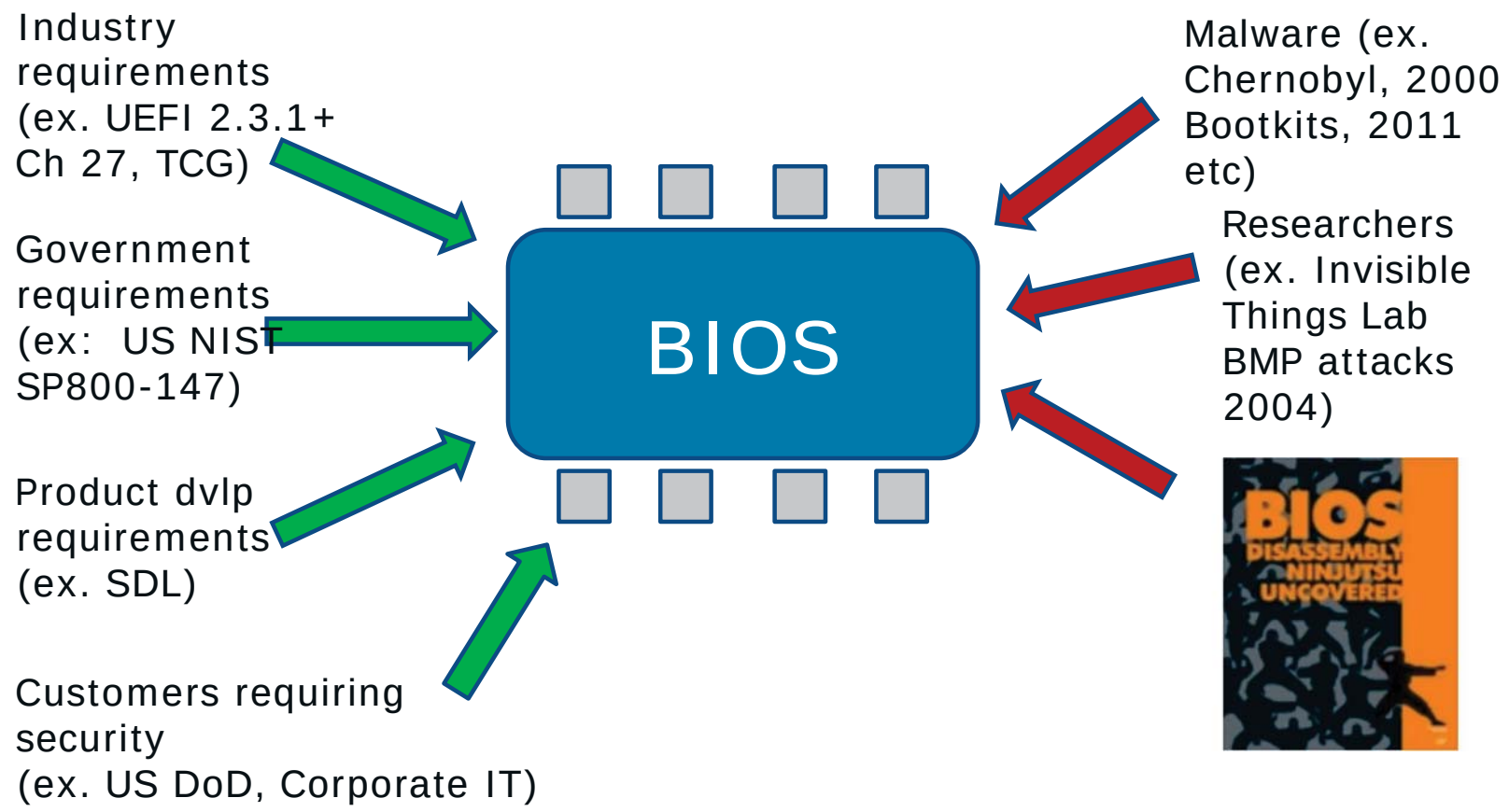
What problem are we solving?

(more importantly, what problems we are not solving w/ this technology)

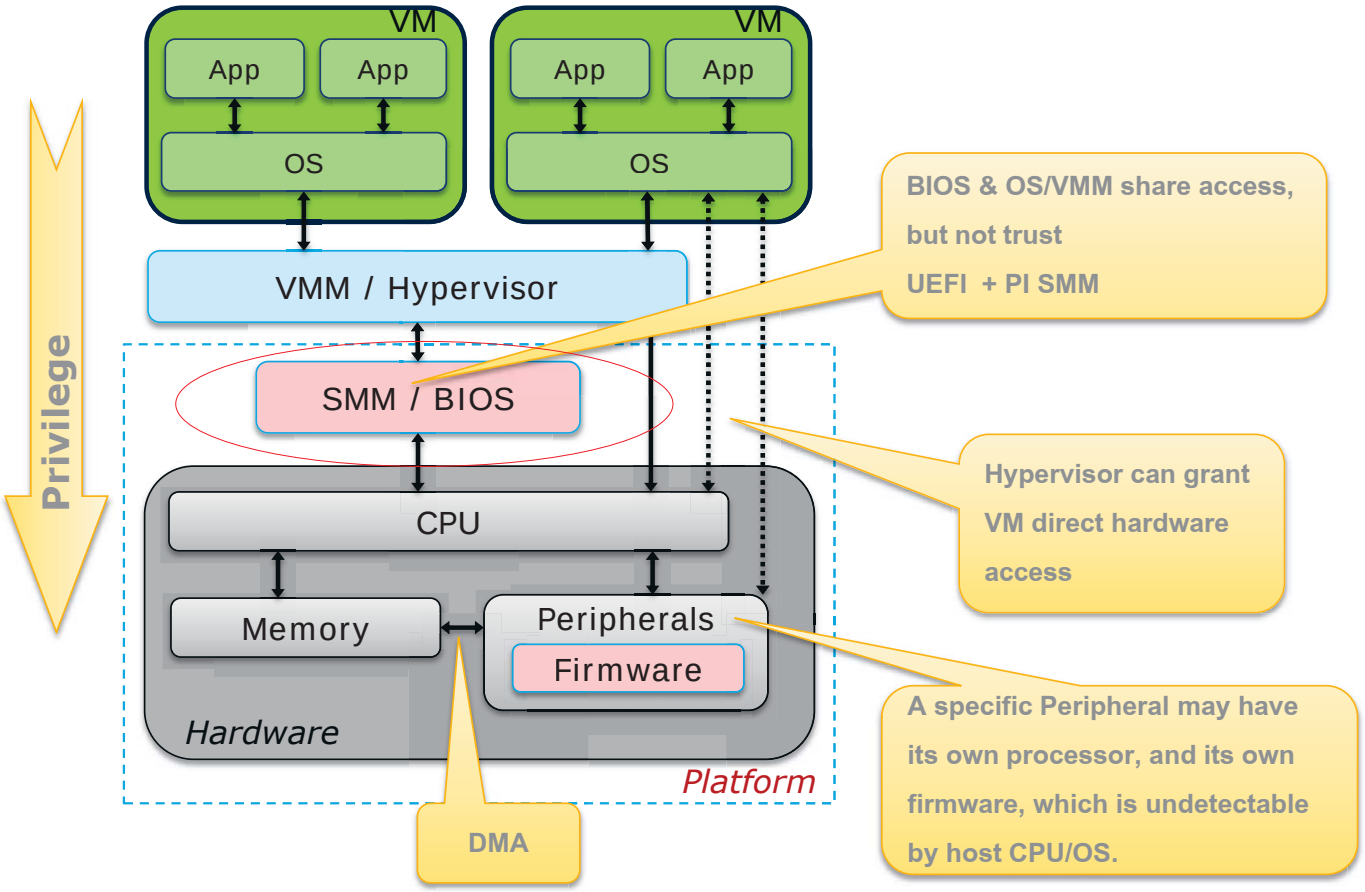


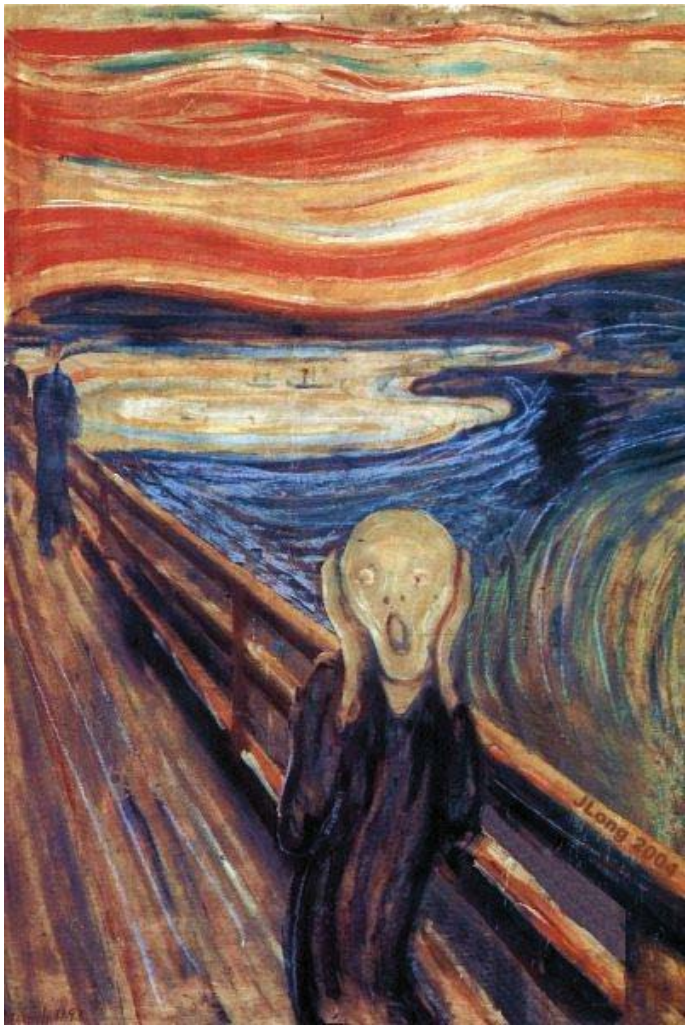
Why

Pressure on BIOS



Where are we (BIOS / UEFI firmware)?




JLong 2004

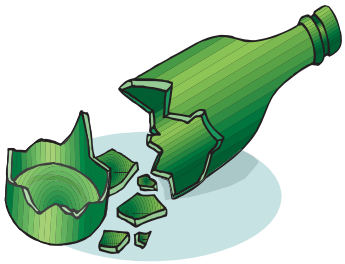
Why

Why use UEFI Secure Boot

Without

Possible corrupted or destroyed data

- BootKit virus – MBR Rootkits
- Network boot attacks e.g. PXESPOILT
- Code Injection Attacks



With

Data integrity

- Trusted boot to OS
- Trusted drivers
- Trusted Applications

a

*

What

What is Security from BIOS Perspective

Secure Boot - UEFI

- Defined a policy for Image loading
- Cryptographically signed
  - Private key at signing server
  - Public key in platform

Measured Boot -Trusted Computing Group (TCG)

- Trusted Platform Module (TPM)
  - Isolated storage and execution for Logging changes, attestation

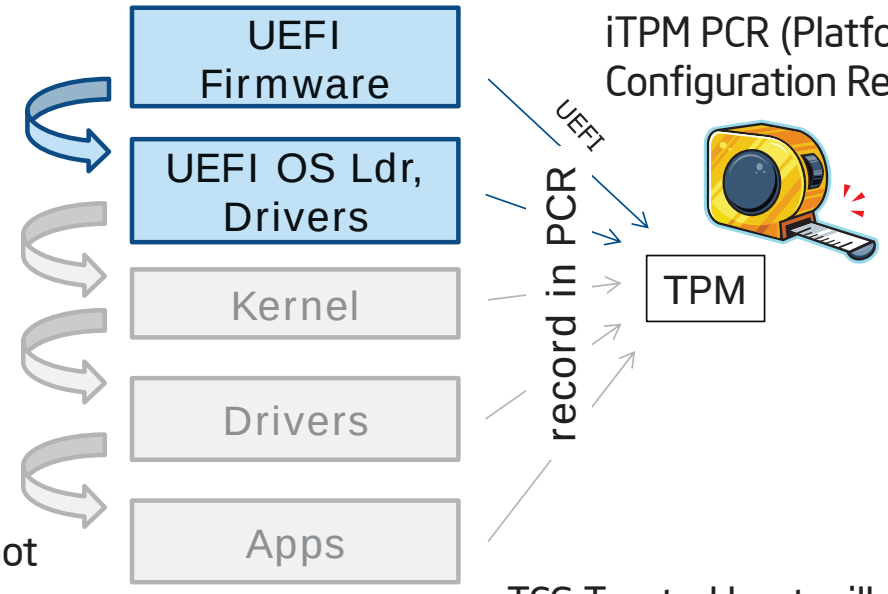
NIST 800-147 -Security Guidelines for System BIOS Implementations

What

UEFI Secure Boot VS TCG Trusted Boot

UEFI authenticate OS loader
(pub key and policy)

Check signature of before loading



UEFI PI will measure OS loader & UEFI drivers into iTPM 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 SW to make security decision using attestation

What

NIST Implementation Requirements

Make sure UEFI PI code is protected

The NIST BIOS Protection Guidelines break down to three basic requirements...

1. The BIOS must be protected
2. BIOS updates must be signed
3. BIOS protection cannot be bypassed

What

UEFI Secure Boot Goals

Local verification. Complements measured boot

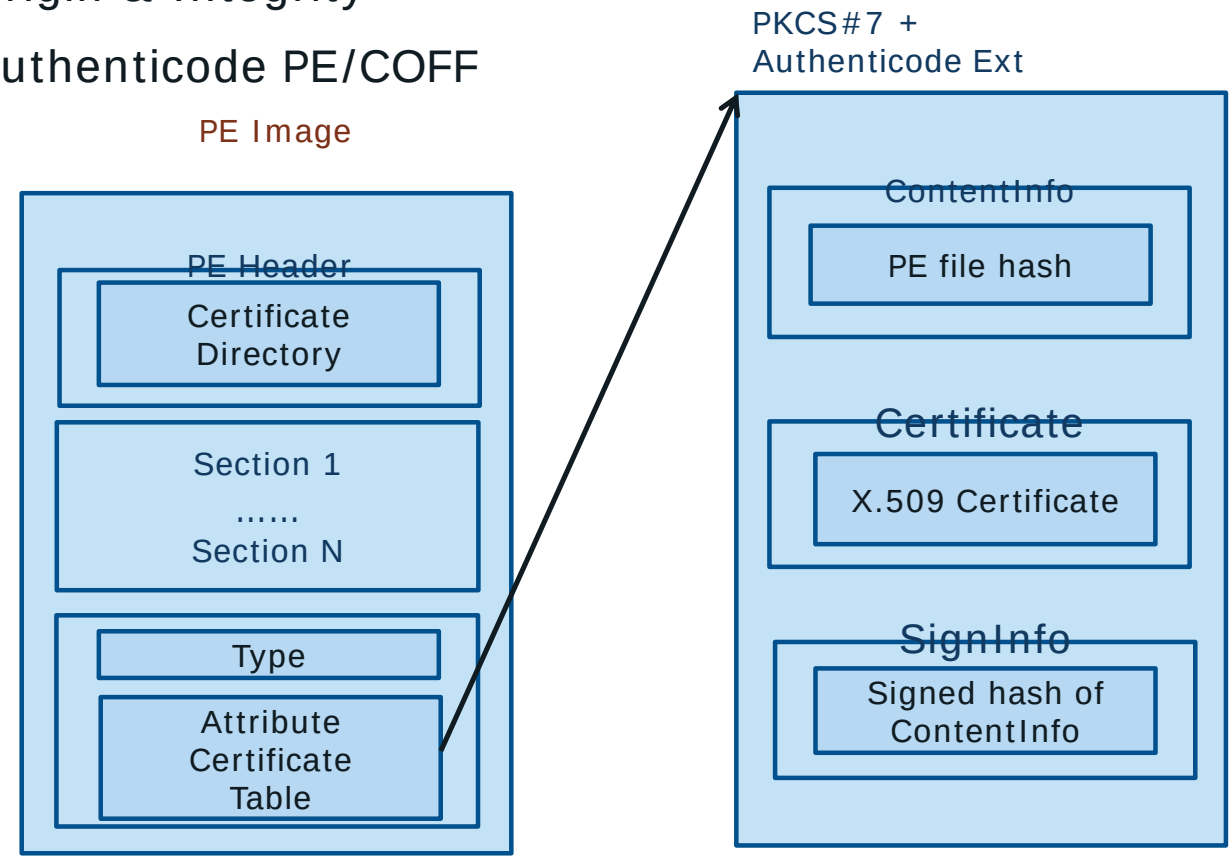
Allow the platform owner to check the integrity and security of a given UEFI image ensuring that the image is only loaded in an approved manner.

Allow the platform owner to manage the platform's security policy as defined by the UEFI Secure Boot authenticated variables

UEFI Image (driver & application/OS loader) Signing

Why? – Origin & Integrity

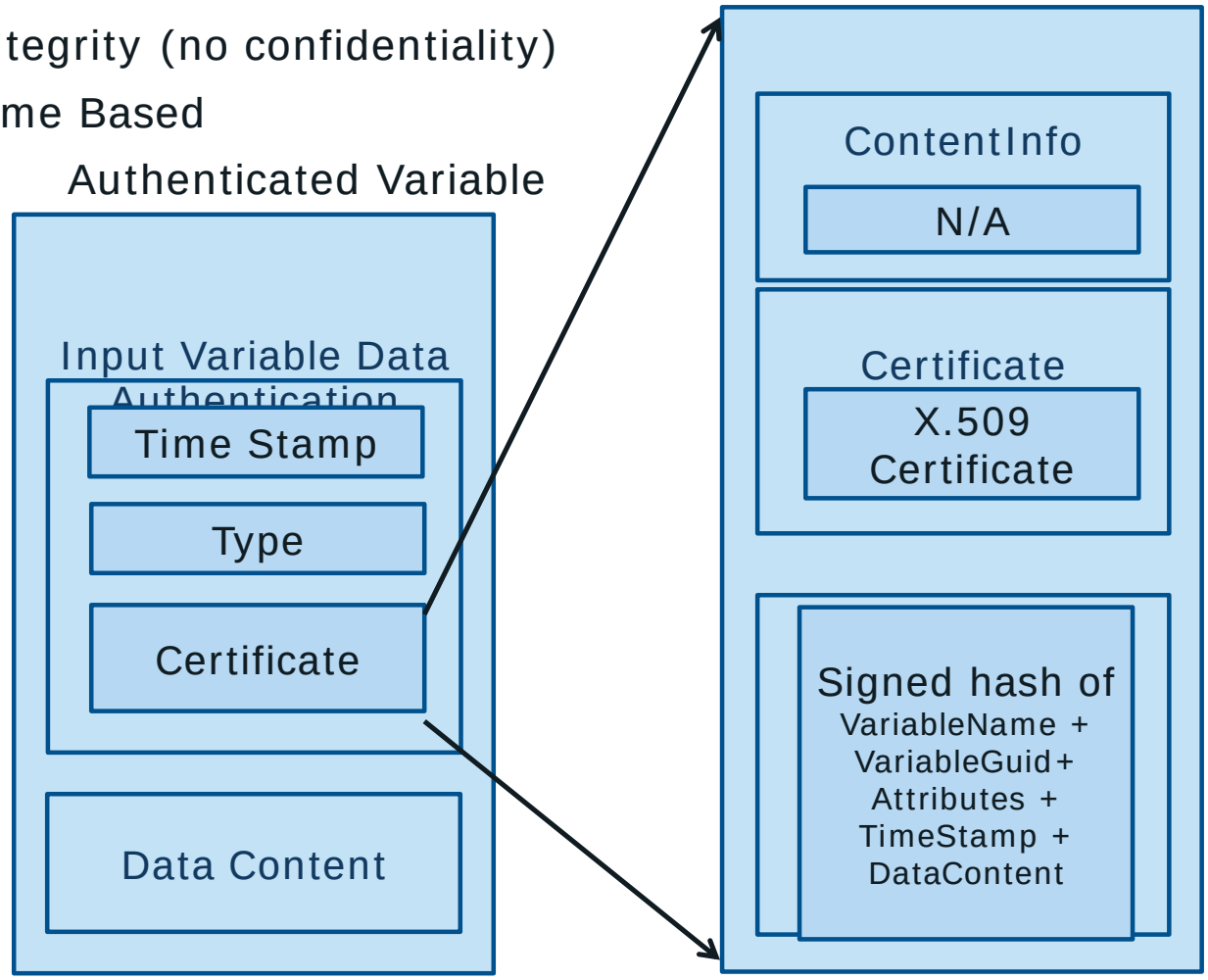
How? – Authenticode PE/COFF



UEFI Authenticated Variable

Why? – Integrity (no confidentiality)

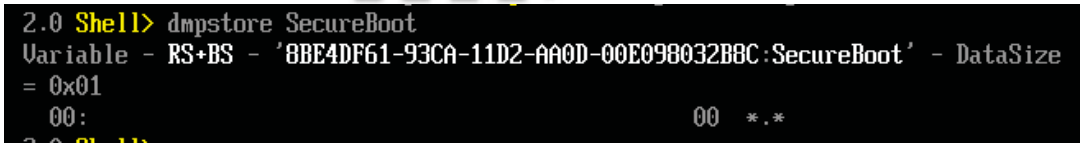
How? – Time Based



Authenticated Variables

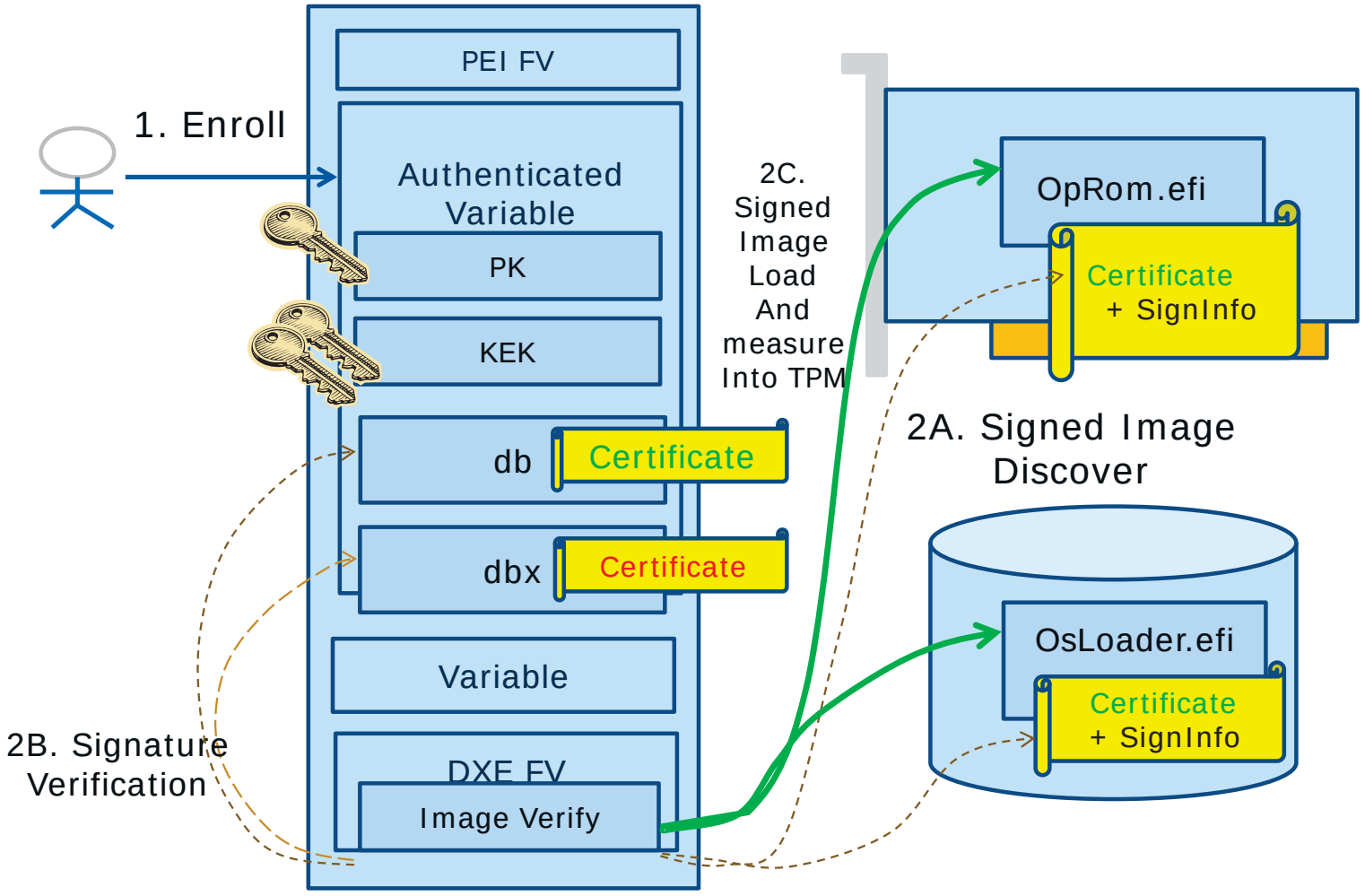
Secure Boot's Authenticated Variables

Key/ DB Name	Variable	Details
PkPub	PK	OEM and Platform FW- format is RSA-2048
Key Exchange Key	KEK	Platform FW and OS - format is RSA-2048
Authorized Signature DB	DB	Authorized Signing certificates - white list
Forbidden Signature DB	DBX	Unauthorized Signing certificates - Black list
Setup Mode	SetupMode	NULL - Secure Boot not supported 0 - PK is enrolled - in user mode User mode requires authentication 1 – Platform is in Setup mode – no PK enrolled
Secure Boot	SecureBoot	1-Platform in Secure boot mode

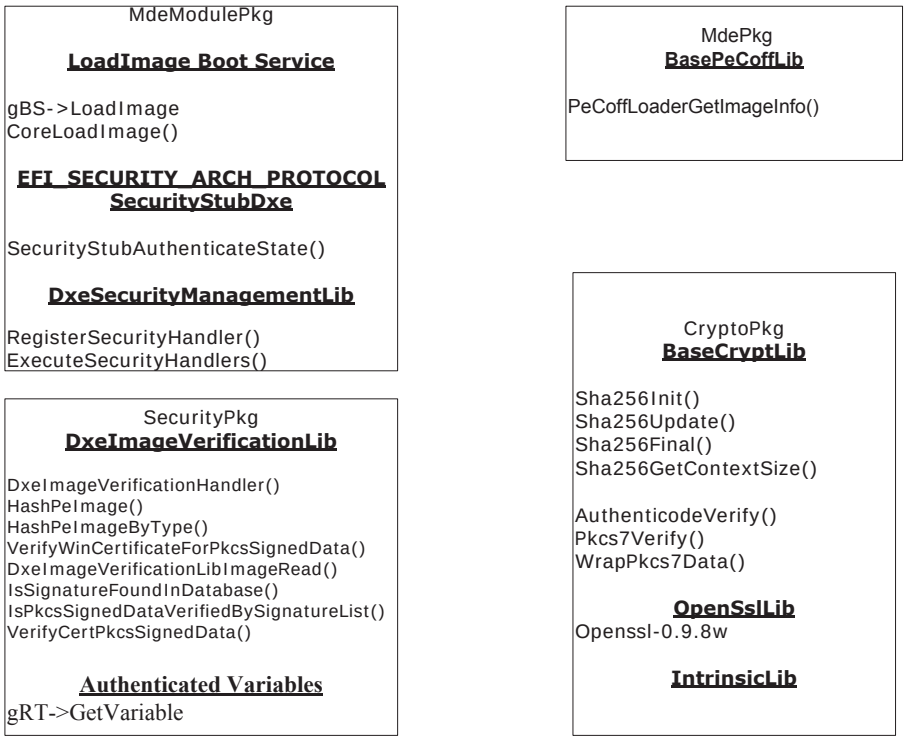


Authorization Flow

UEFI Secure Boot Flow

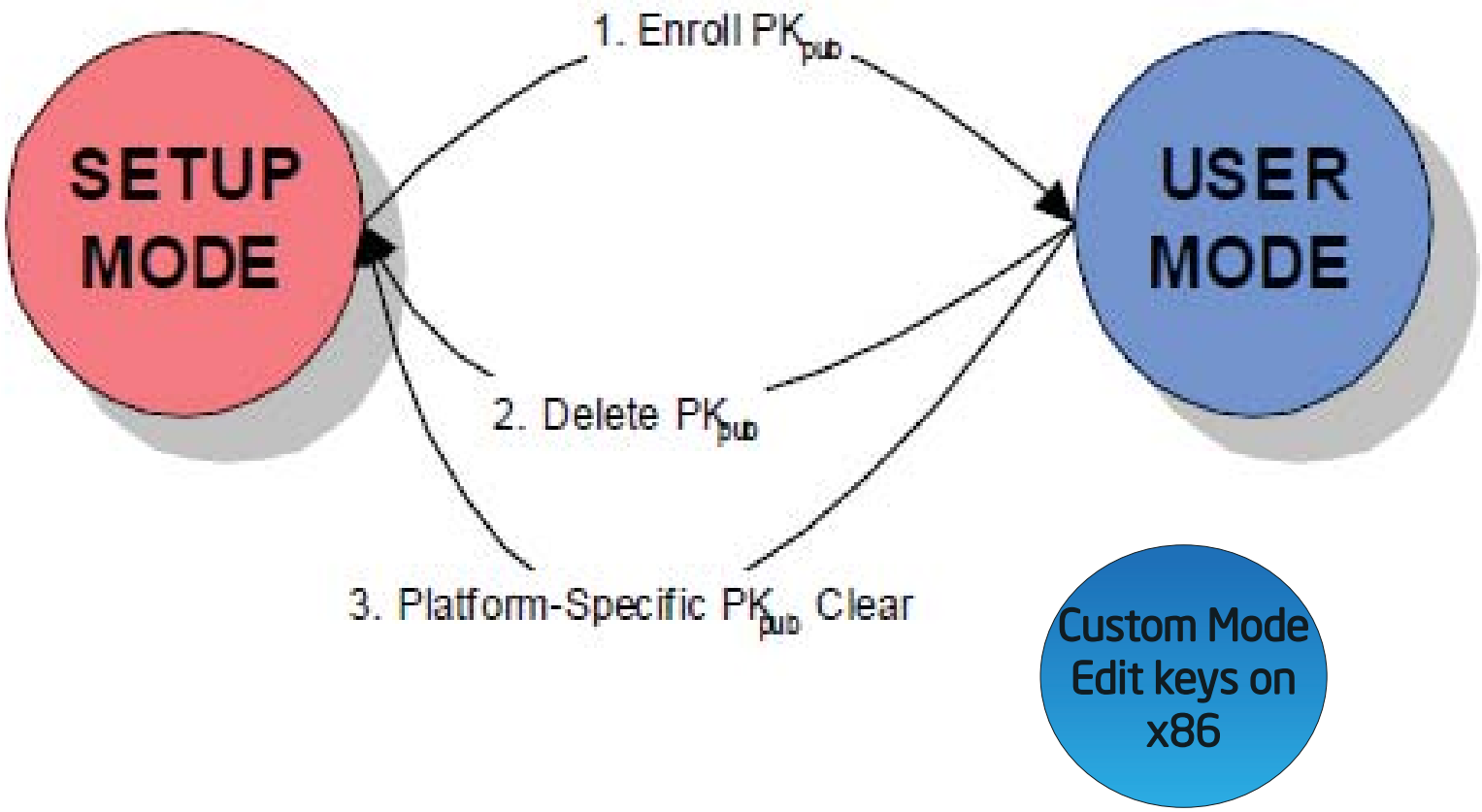


Relevant open source software packages/routines for Authorization flow



See *Rosenbaum, Zimmer, "A Tour Beyond BIOS into UEFI Secure Boot,"* for more details

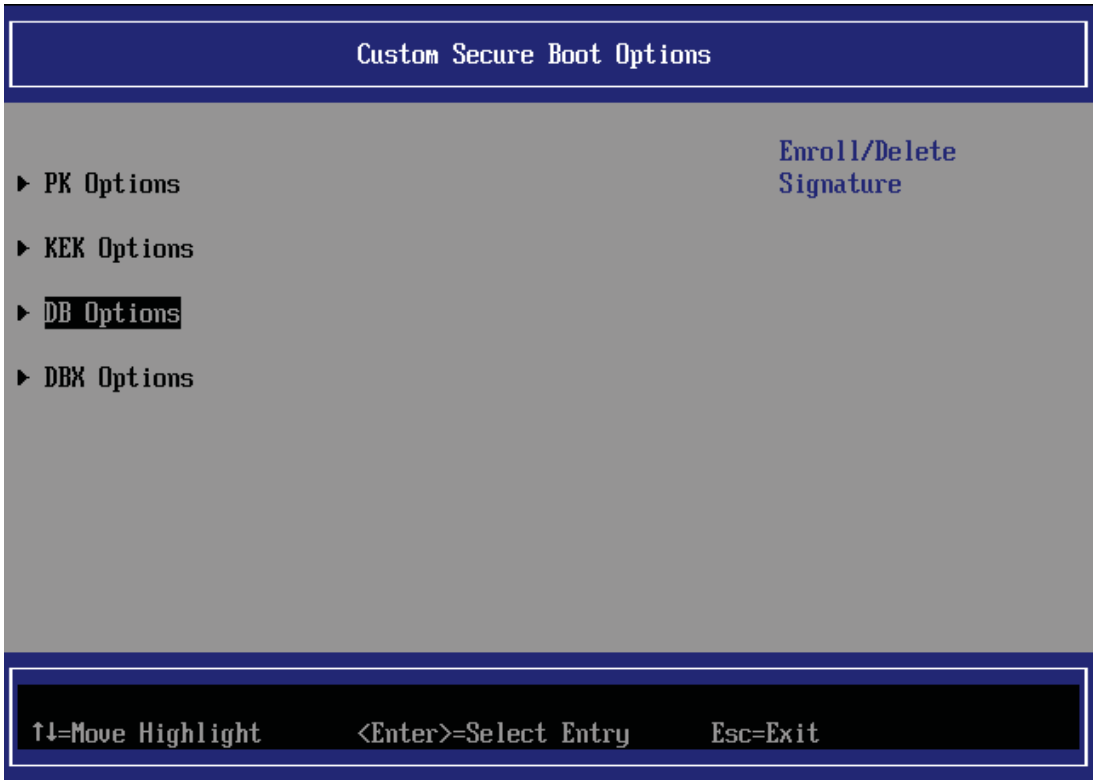
Put them altogether: UEFI Secure Boot



Enable Secure Boot

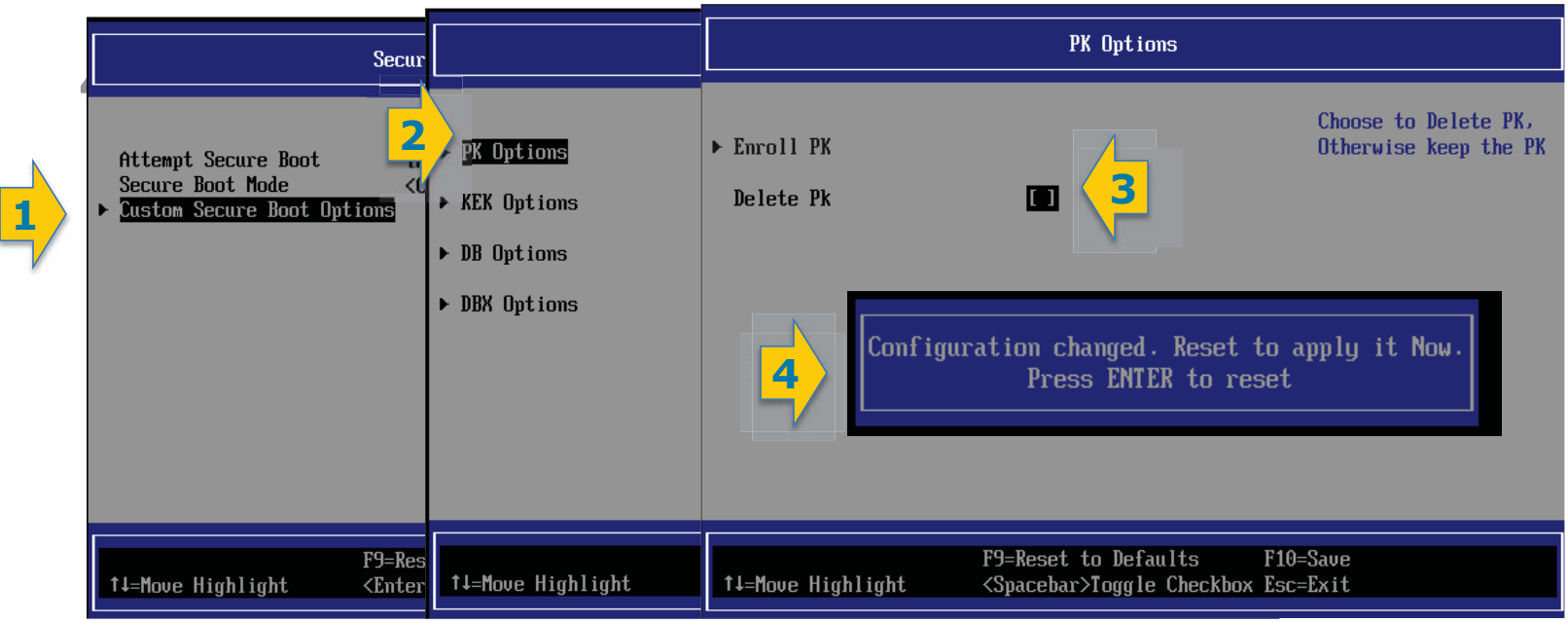
End user controls -Custom Secure Boot Options

Enrolling DB and/or DBX for physically present user

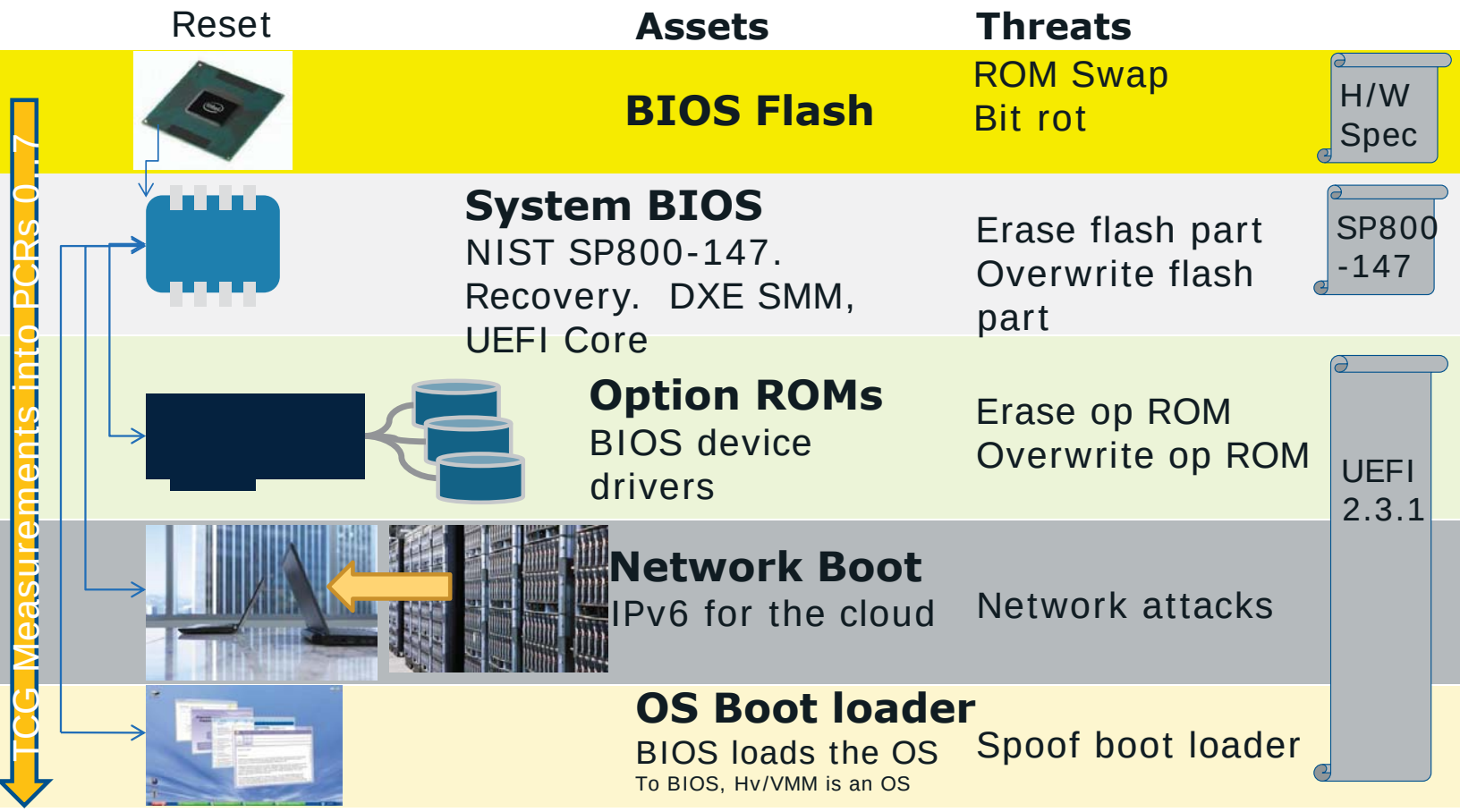


Disable Secure Boot

1. Select Custom Secure Boot Options
2. Select PK Options
3. Delete Pk (space bar)



Technologies – putting it together



Different colors for different vendors

Challenges

- Multi-OS support, GPL3 & Open source
- Firmware size – open source & crypto libs
- Speed impacts
- Consistency w/ other ‘security’ technologies in platform
- Robustness
  - Coding practice
  - Protected updates
  - Recovery
- Validation
  - Negative testing
  - Fuzzing
- Interoperability of different implementations

Summary

- Threats of UEFI extensibility are real
- Address w/ open standards and open source
- Secure boot is coming w/ next OS wave (and like longevity of any shrinkwrap OS release, will continue for 10 yrs)
- Challenges in ecosystem enabling

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
http://www.intel.com/technology/itj/2011/v15i1/pdfs/Intel-Technology-Journal-Volume-15-Issue-1-2011.pdf

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

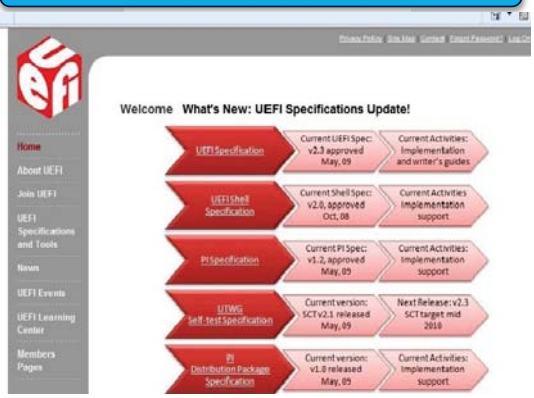
Beyond BIOS: Developing with the Unified Extensible Firmware Interface, 2nd Edition, Zimmer, et al, ISBN 13 978-1-934053-29-4, Chapter 10 – Platform Security and Trust,
http://www.intel.com/intelpress

“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

UEFI Industry Resources

UEFI Forum



www.uefi.org

UEFI Open Source

www.tianocore.org

Intel UEFI Resources

www.intel.com/UDK

Intel EBC Compiler

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

UEFI Books/ Collateral

www.intel.com/intelpress

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

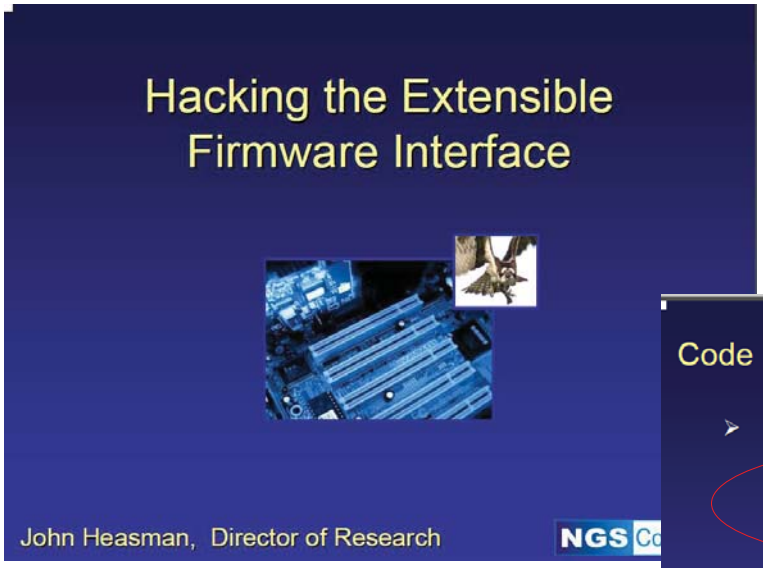
ToorCamp

Thank You

Contact:
vincent.zimmer@gmail.com

Backup

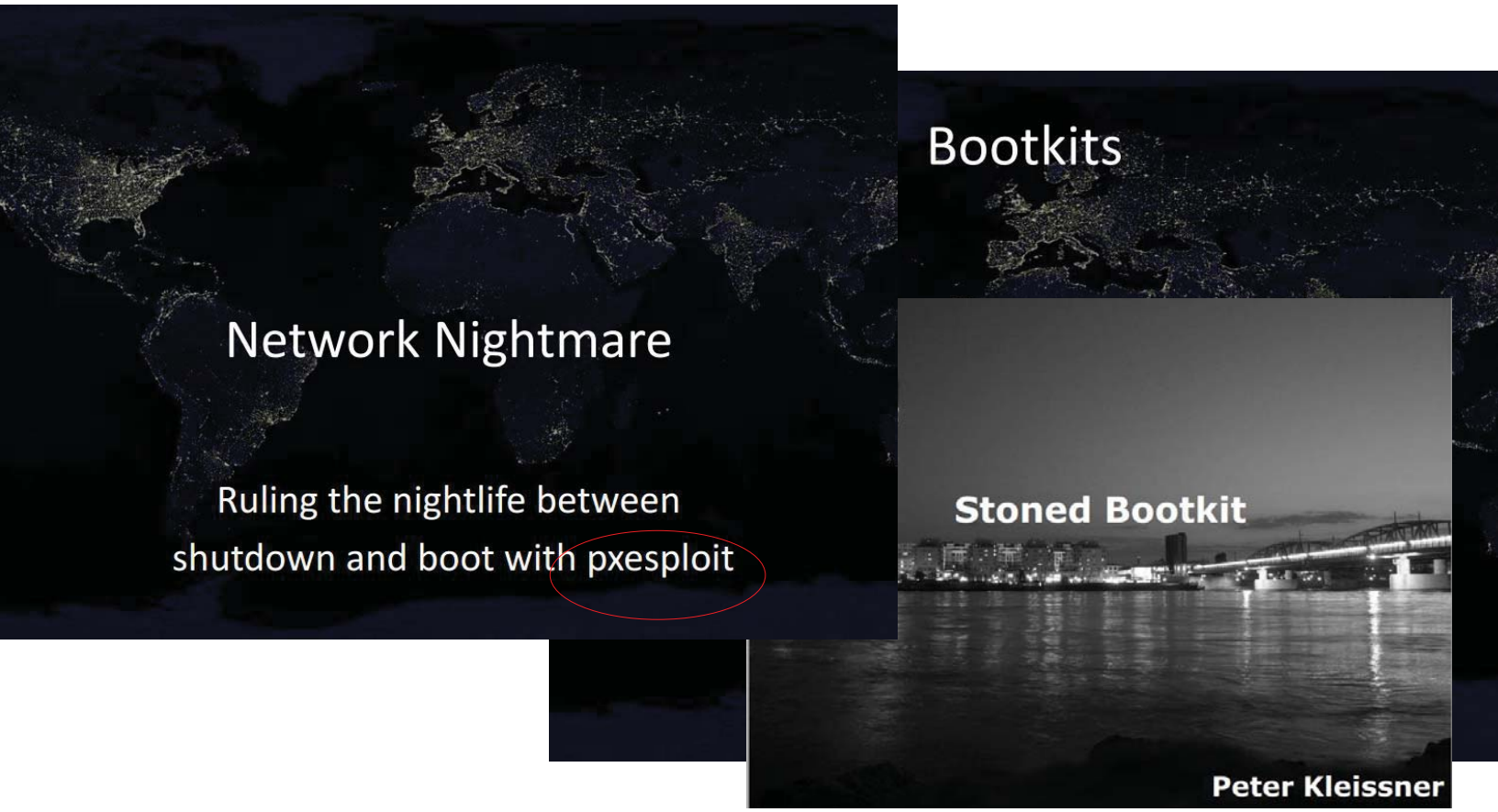
History of attacks - 2007 - Blackhat Las Vegas



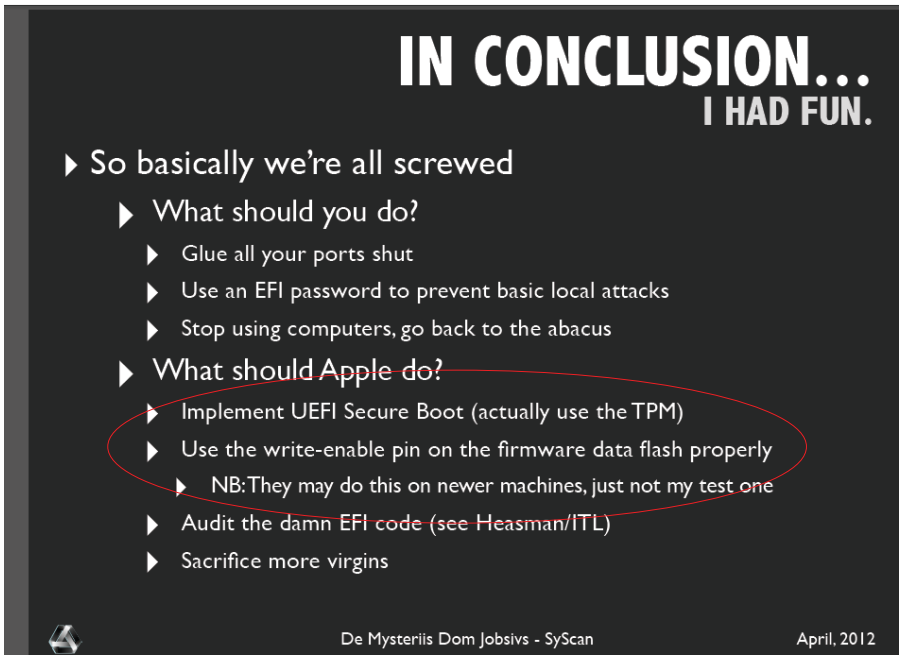
Code Injection Attacks

- Important when firmware verifies digital signatures
  - Depends on implementation flaw in driver
  - e.g. stack overflow, heap overflow
  - or incorrect signature verification
- Plenty of targets:
  - File system drivers (e.g. FAT32, HFS+)
  - PE parsing code
  - Crypto code (Data in certs, ASN.1 decoding)
  - Network interaction (PXE)

Defcon 19 – Bootkits and network boot attacks



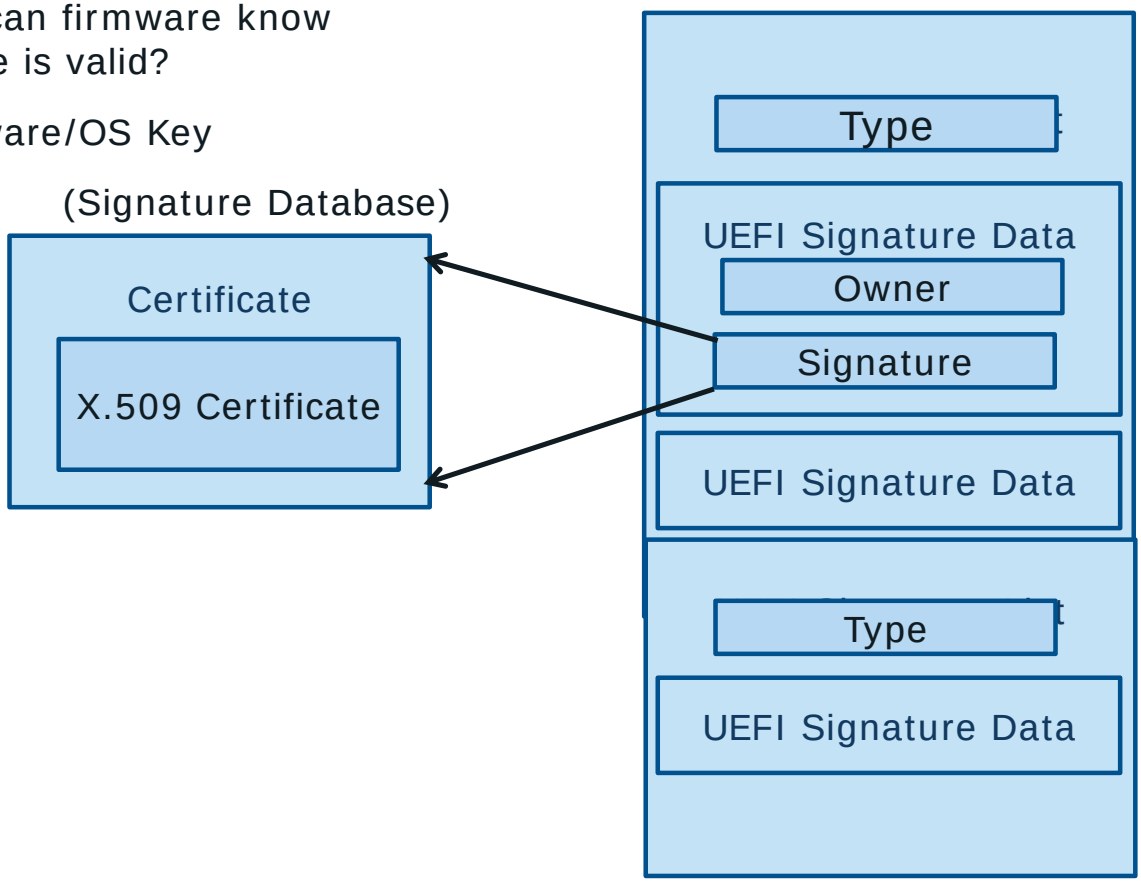
SYSCAN Singapore – April 2012



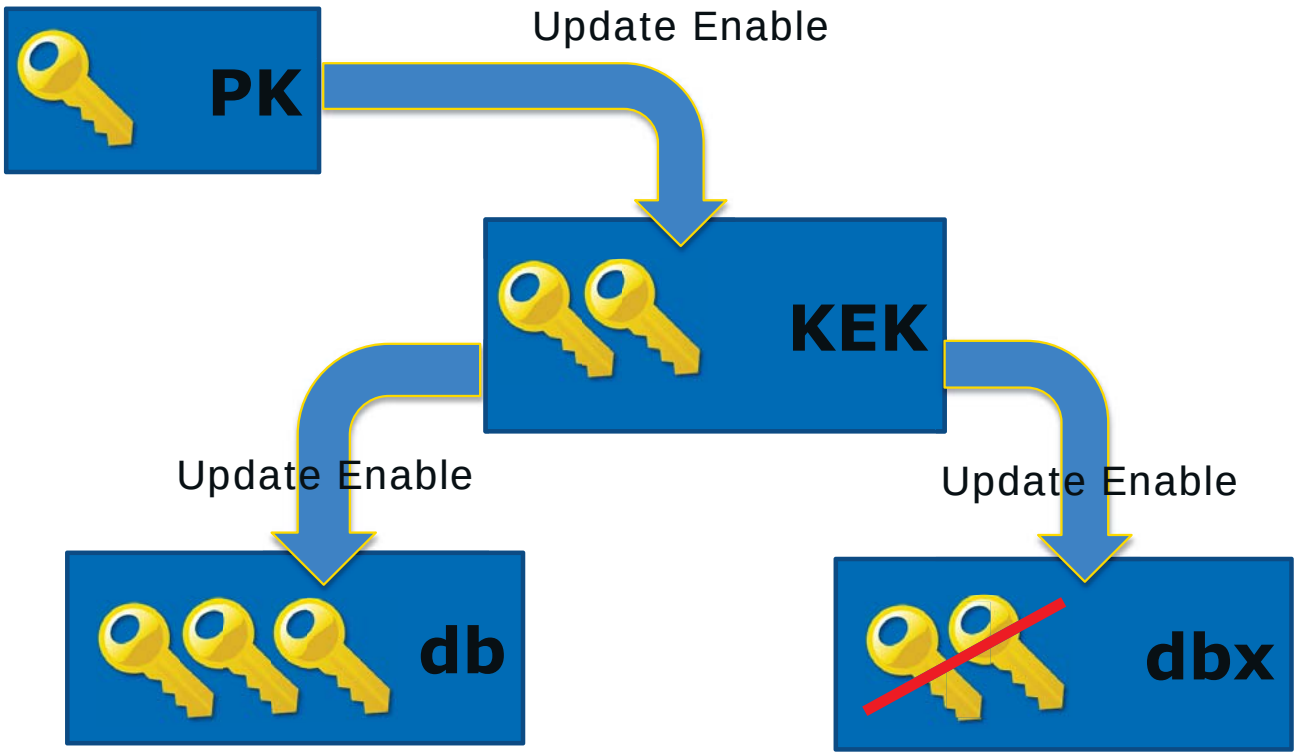
Firmware/OS Key

Why? – How can firmware know if certificate is valid?

How? – Firmware/OS Key



UEFI Secure Boot Database Review



Who "Owns" The System Security Keys?

PK – Key pair is created by Platform Manufacturer

Typically one PK pair used for a model or model Line

KEK – Key supplied by OS Partner,

Optional: Include 2nd key created by OEM

db – OS vendor supplies Key,

CA supplies Key,

Optional: OEM App Signing Key

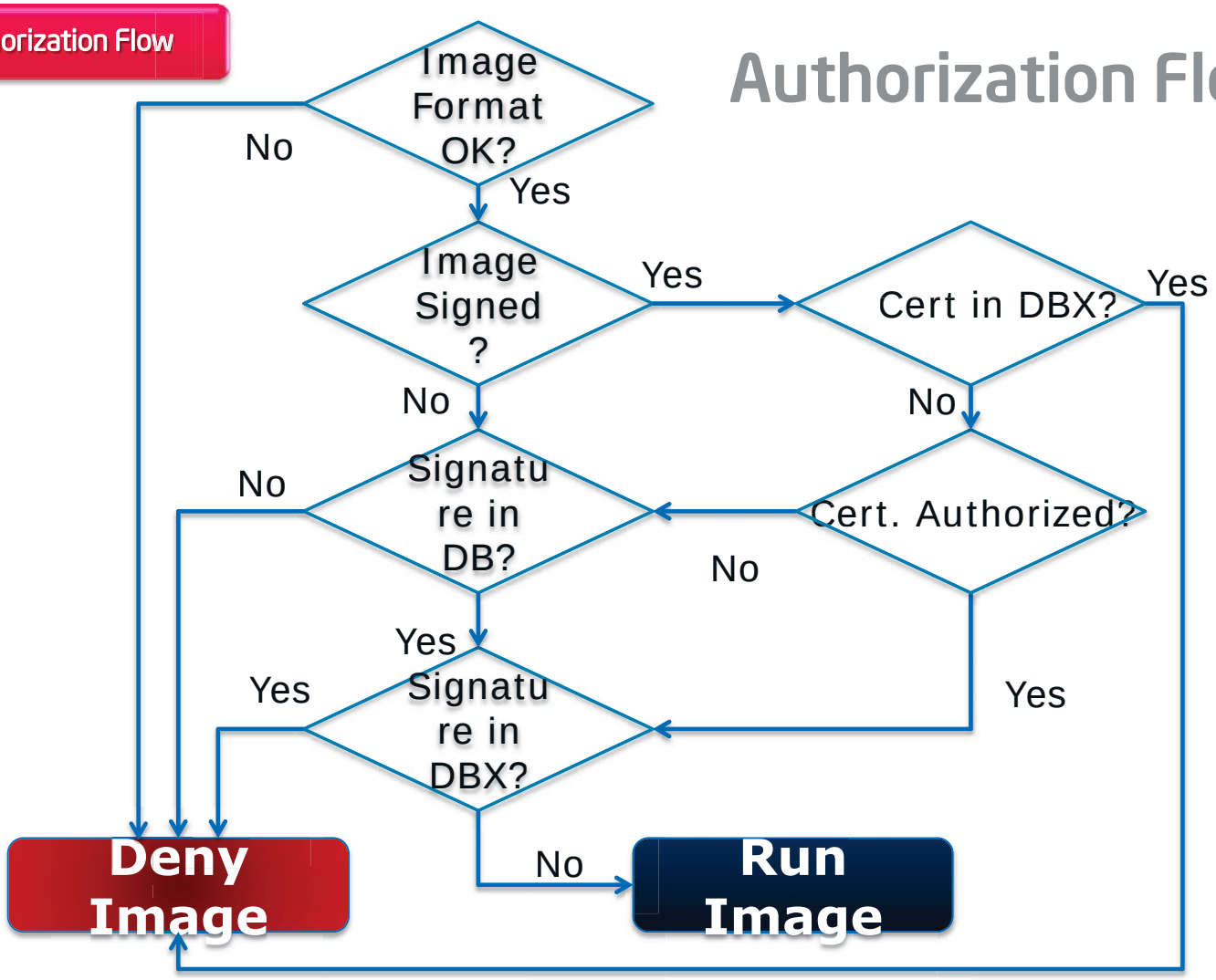
dbx – list of revoked keys

- Signing authority issues revoked keys

Signature Tests using db Keys Block Rogue S/W!

Authorization Flow

Authorization Flow



See *Rosenbaum, Zimmer, "A Tour Beyond BIOS into UEFI Secure Boot,"* for more details