St. Petersburg State University of Aerospace Instrumentation
Department of Information Systems Security

Too Young to be Secure: Analysis of UEFI Threats and Vulnerabilities

Anton Sergeev
Vladimir Bashun
Vector Minchenkov
Alexandr Yakovlev

14 November 2013

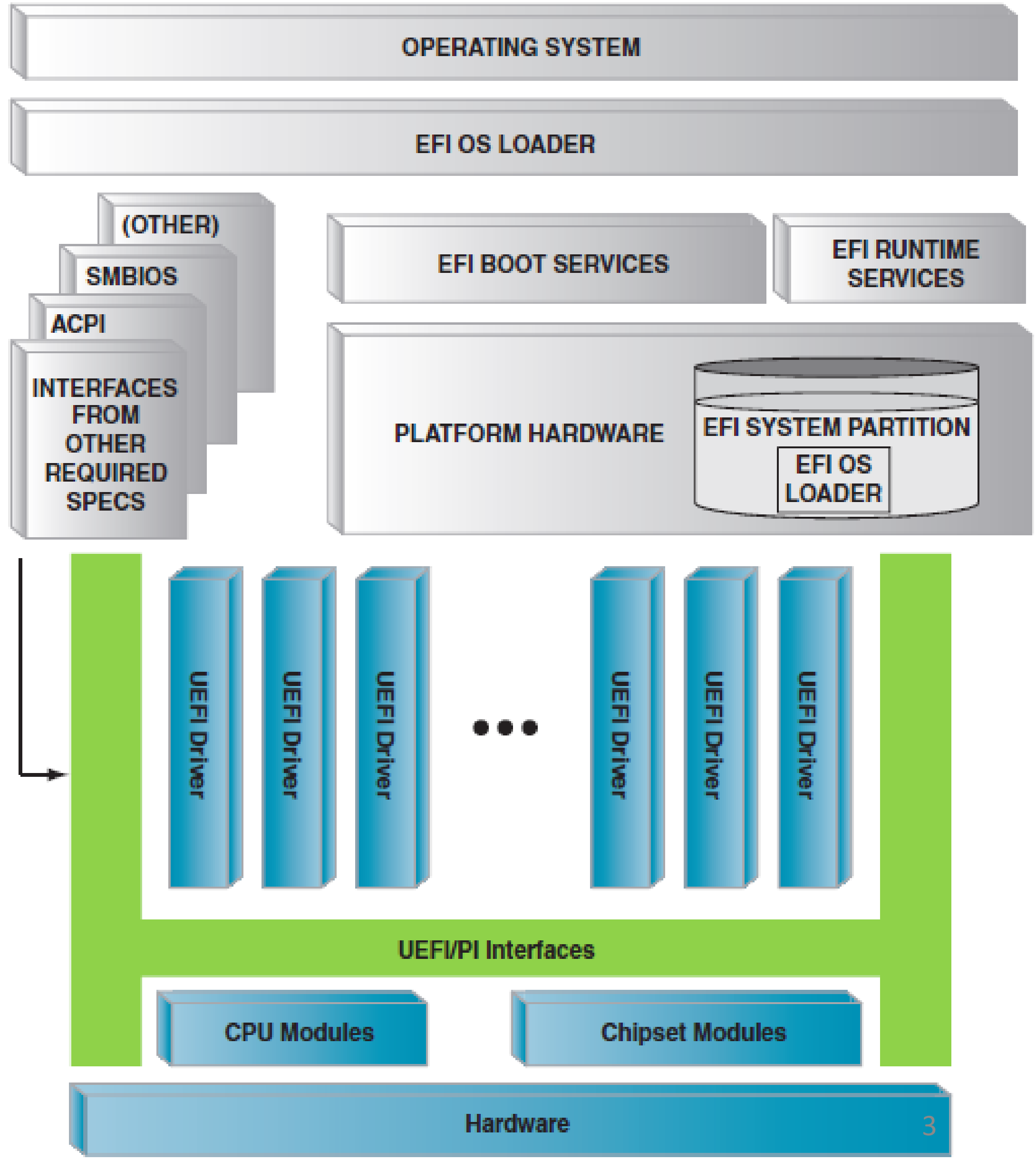
Supported by Infotecs Corp.

Goal of the work

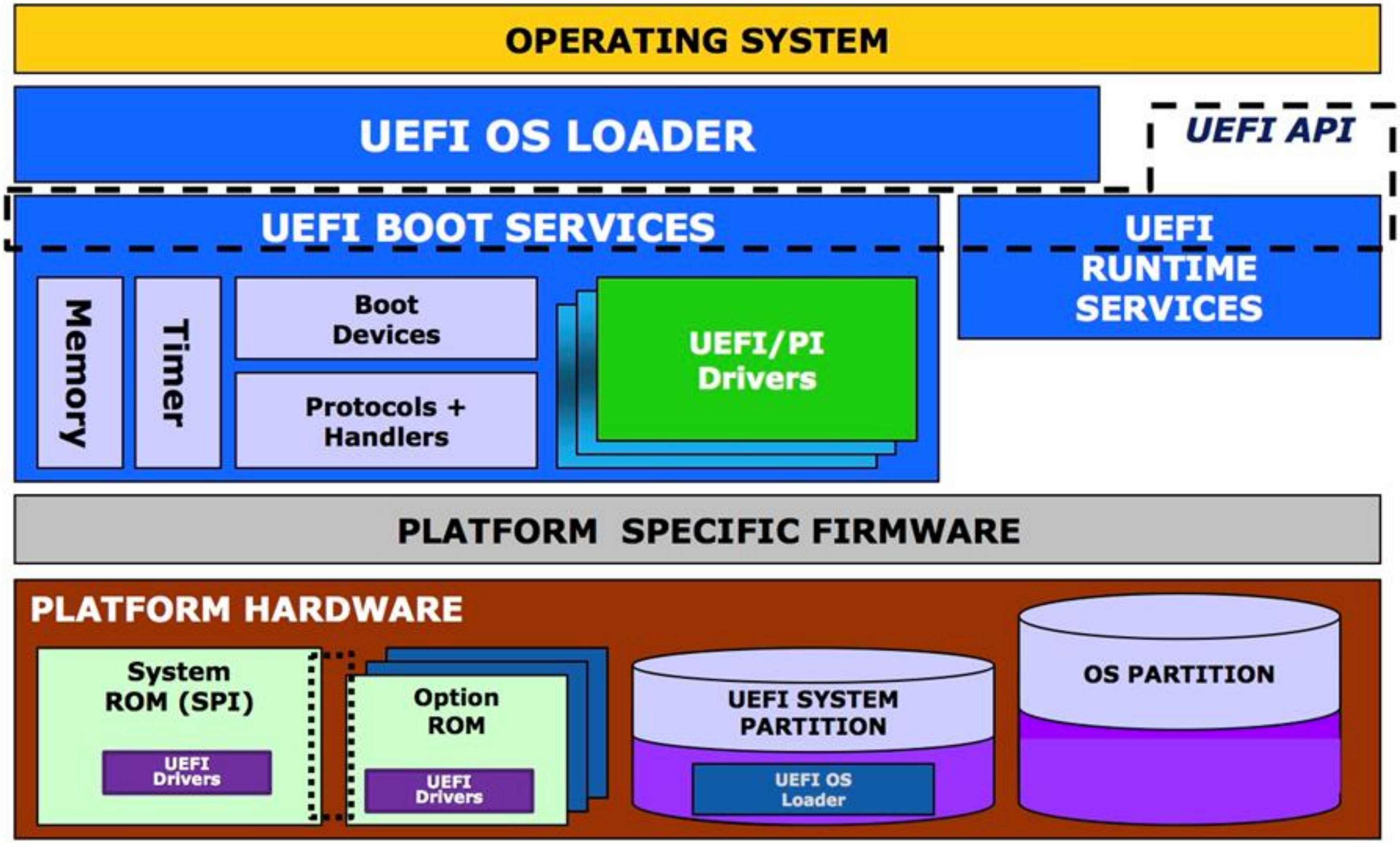
- Analysis of the UEFI Security Issues
  - Review
    - Review of the known vulnerabilities, threats and attacks
  - Research
    - Find new vulnerabilities, threats and attacks
  - Investigate
    - Analyze the points and sources of the security problems
  - Classification
    - Classify the architectural and implementation troubles of UEFI
  - Recommendations
    - how to make this young technology more safe and secure

Unified Extendable Firmware Interface (UEFI)

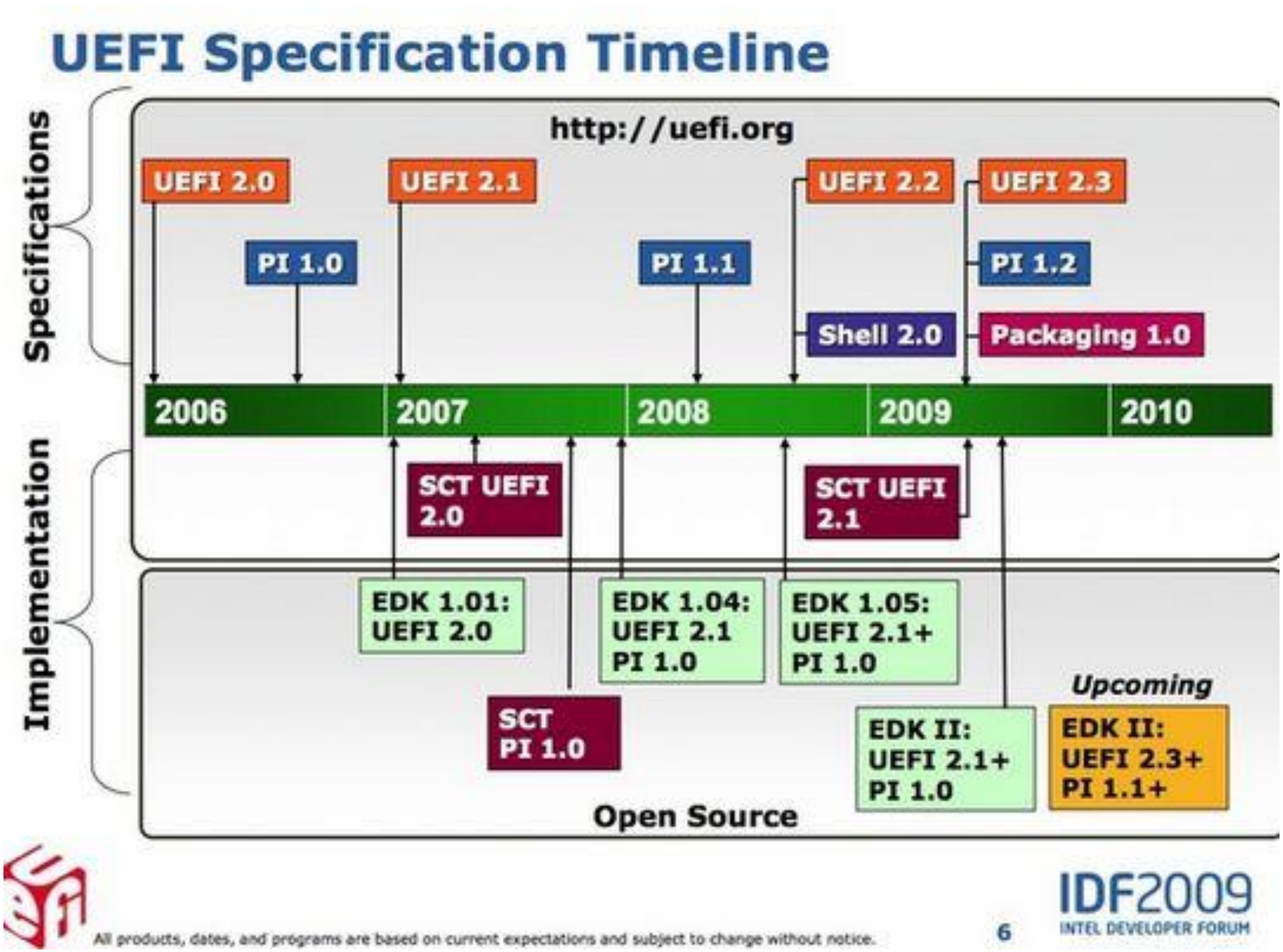
- Specification that defines a new software interface between an operating system and platform firmware
- Meant to replace BIOS
- Developed and promoted by Intel and Microsoft
- UEFI can support remote diagnostics, monitoring, repair and security services for computers even without installed OS



Unified Extendable Firmware Interface (UEFI)



UEFI Development Roadmap



Advantages over BIOS

- Modular design
- Support of multiple boot devices
- Flexible pre-OS environment, including network capability
- CPU-independent architecture
- CPU-independent drivers
- OS-independent boot and runtime services
- Security and validation of the OS loader

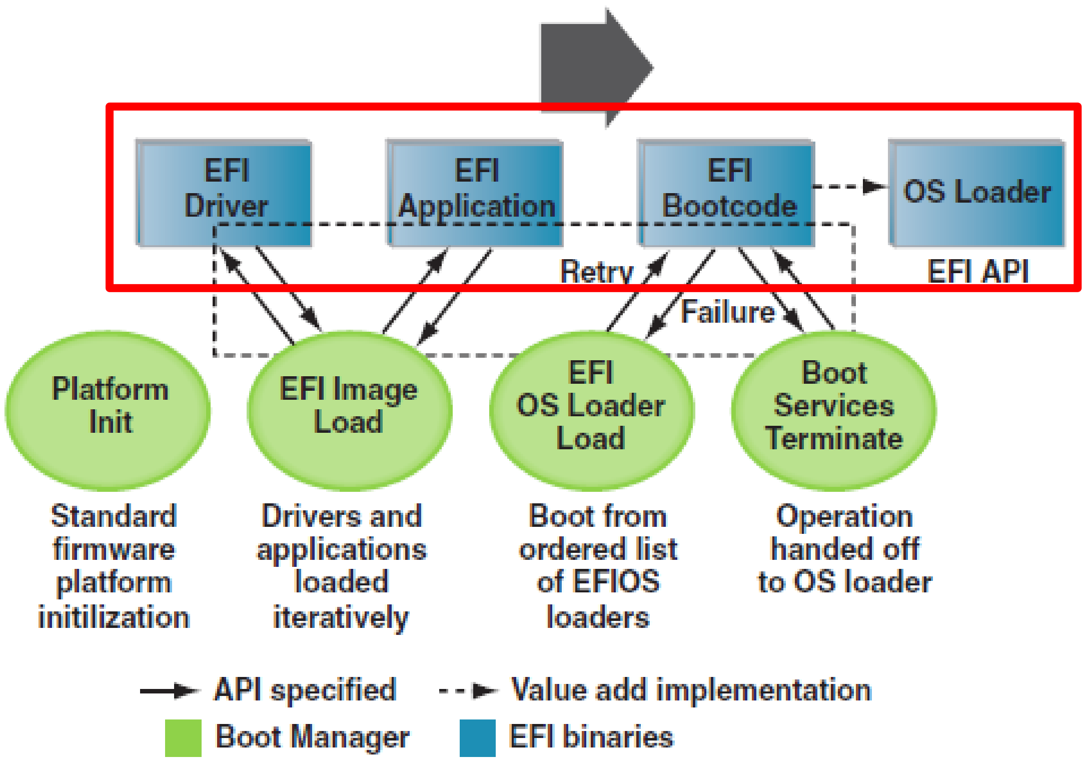
Security in UEFI

- Secure boot (UEFI Spec)
  - Secure Boot
  - Verification of the loaded modules
- Measured Boot
  - TPM
  - Logging

Security in UEFI

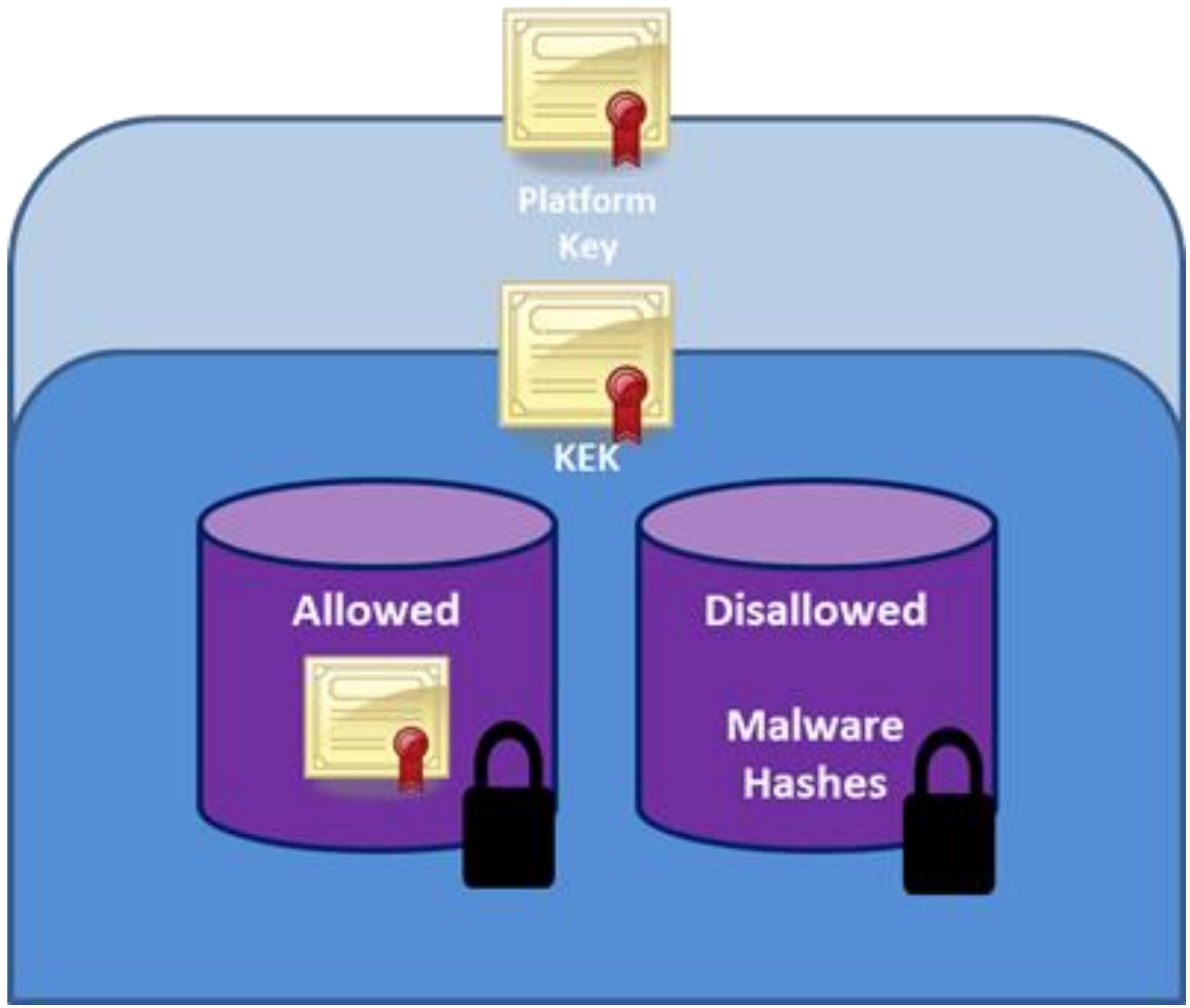
- Secure boot (UEFI Spec)
  - Secure Boot allows to load the signed drivers/apps only
  - Verification of the loaded modules allows to check the integrity of key system components
- Measured Boot
  - TPM = Trusted Platform Module, a secure crypto processor that can store cryptographic keys/hashes
  - Audit and Logging helps to save all the information about all the processes

UEFI Secure Boot

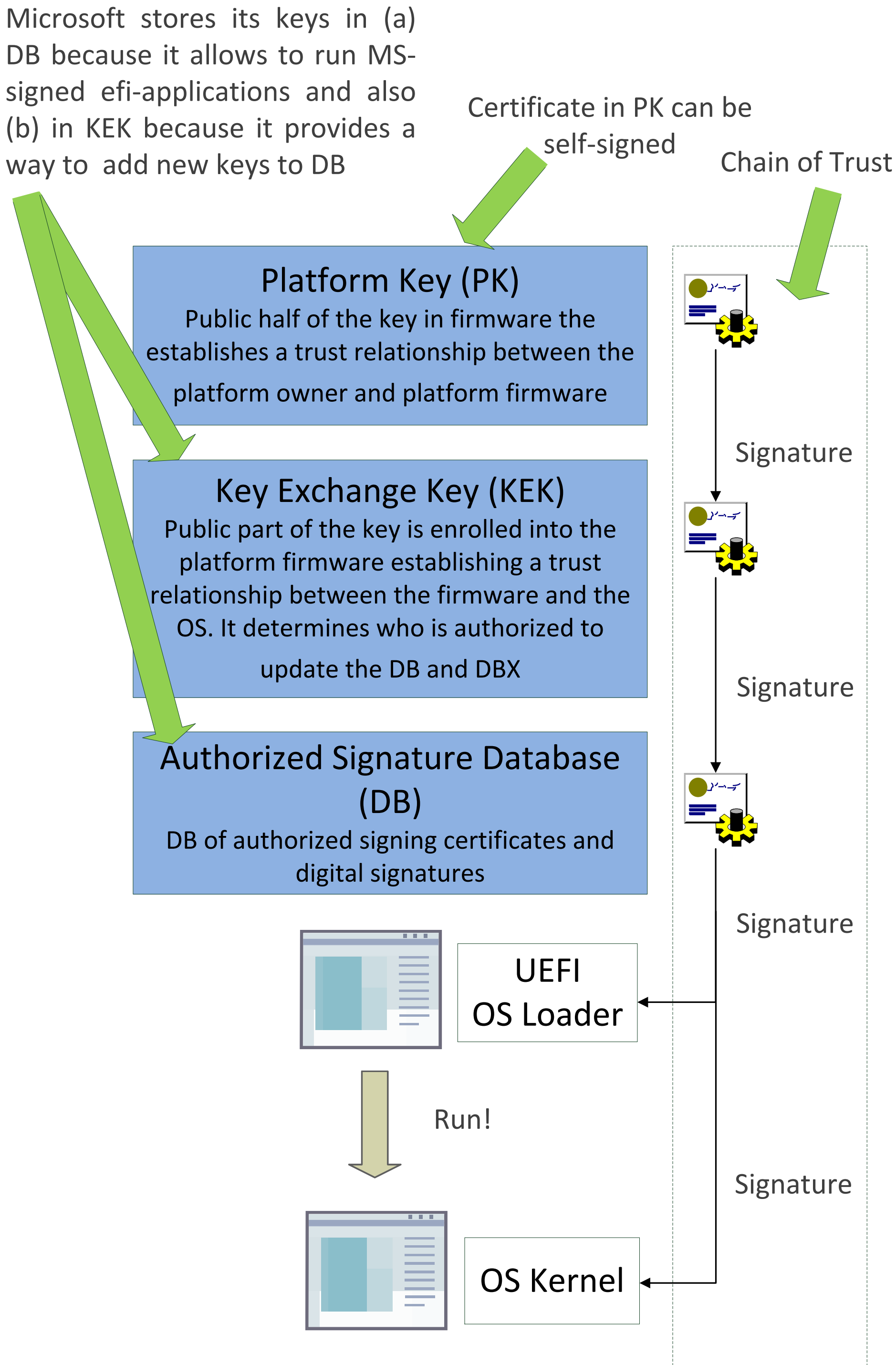


Secures the boot process by **preventing the loading of drivers or OS loaders that are not signed** with an acceptable digital signature

UEFI Secure Boot: Trusted databases of certificates, keys and hashes



UEFI Secure Boot



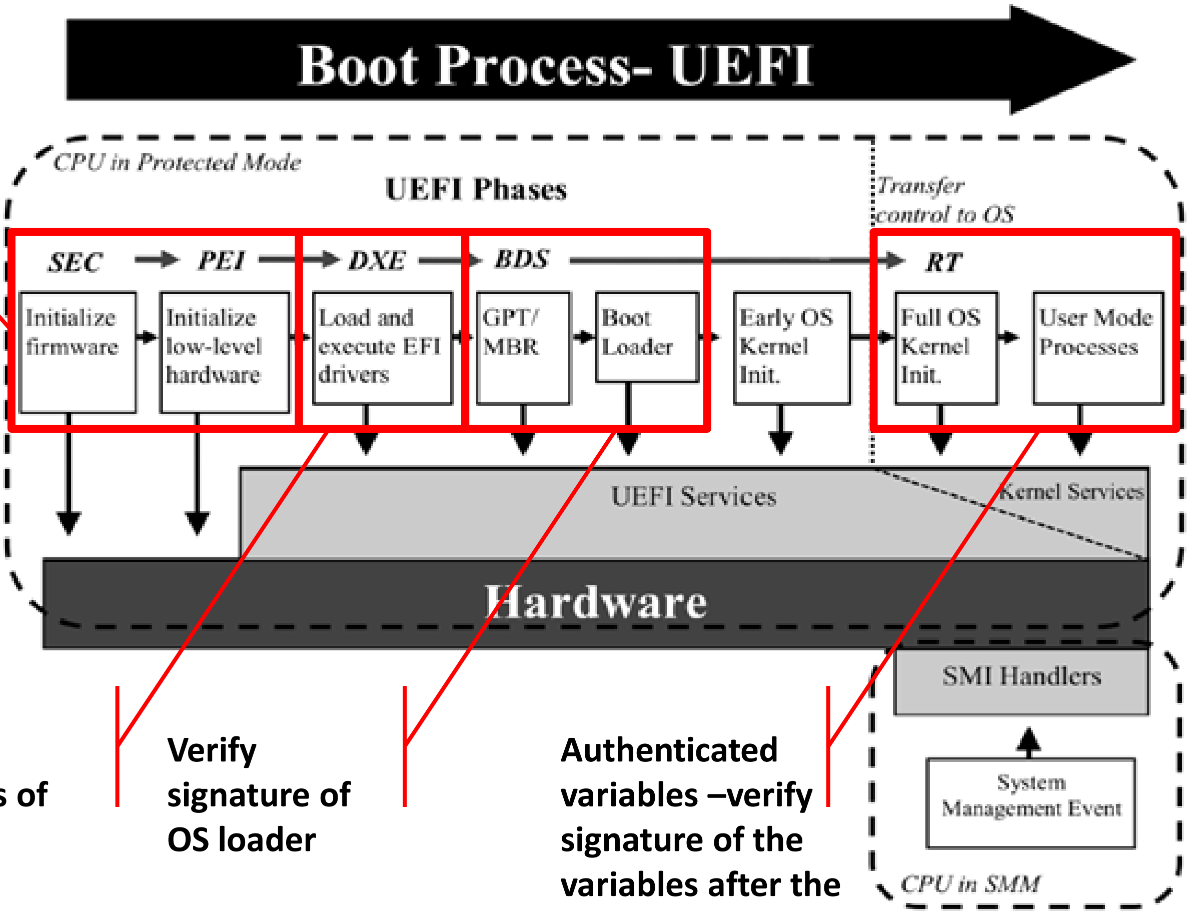
True "Secure Boot"

=

Verification of all the Chain of Trust

from the firmware to OS loader

Loading with Secure Boot



Verify firmware's signature

Verify signatures of drivers

Verify signature of OS loader

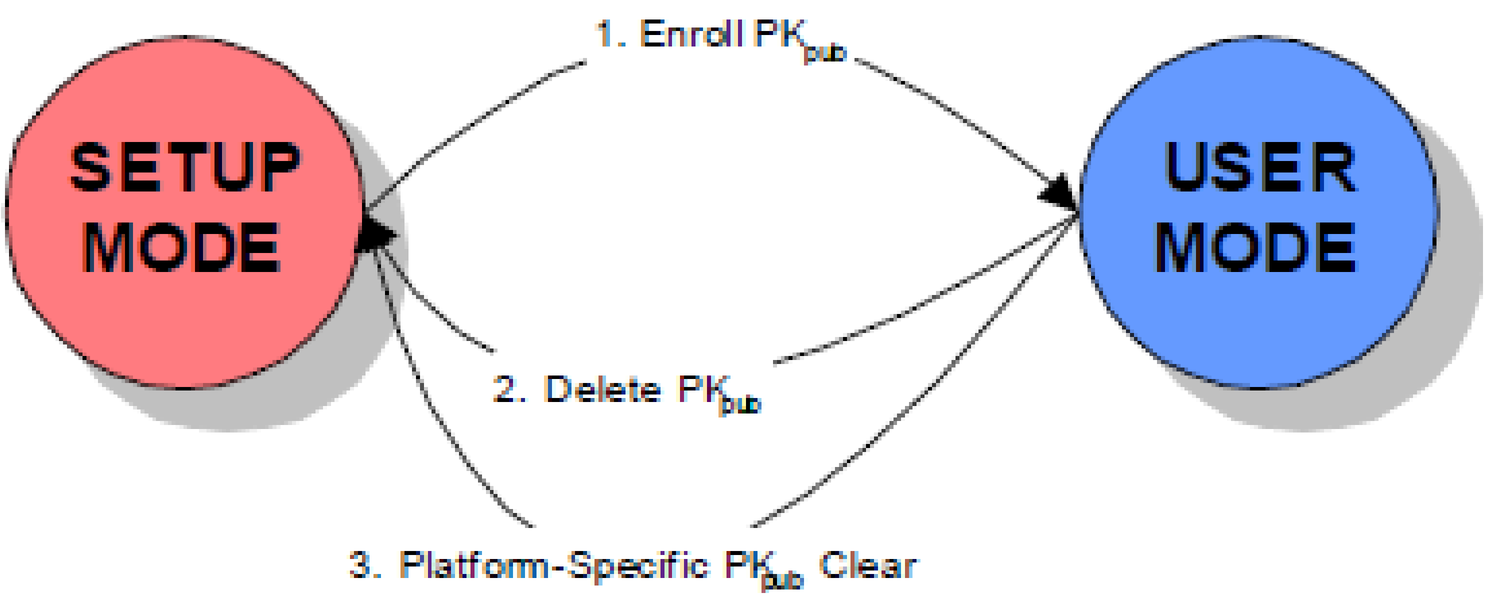
Authenticated variables –verify signature of the variables after the OS loading

True "Secure Boot" =

Verification of all the Chain of Trust

from the firmware to OS loader

UEFI Secure Boot: Initialization



UEFI: key upper-level problems

- Not all operation systems support Secure Boot
- Difficulty of managing keys
  - Problems with Secure Boot under dual-boot loading
  - Problems of HW platform initialization for turning in Secure Boot
- Secure Boot can be turned off by user
- When using virtualization guest OSs are not directly controlled by the secure boot

Testbed

HW+FW+SW system for practical experiments with UEFI

Configuration:

- 1 PC with emulated UEFI:
  - VMM KVM (based on Ubuntu 13.04), emulating UEFI
    - SW packet OVMF for emulation UEFI Secure Boot under KVM
    - 2 guest OSs: Ubuntu 12.10 and Fedora 19
- 1 PC with dual BIOS (2 independent firmware ROM)

Typical Tools

- OVMF
  - UEFI support for VMMs
- Sbsigntools
  - PKI Lib for managing keys and certificates
- Efitools
  - Key management tool for working with certificates (PK, KEK, db, dbx, MOK) and hashes (Hash in MokList).
- MokManager
  - Key management tool for working with certificates and hashes

VMMs and UEFI support

Hypervisor	Support of UEFI	Support of Secure Boot for VMM itself	Full Emulation of UEFI for guest OSs	Support of Secure Boot for guest Oss (Emulation or Path through)
Microsoft Hyper-V	+	+	+	+
VMware vSphere ESXi	+	-	+	-
KVM	+	+	+	+
Xen	+	-	+	+
Red Hat RHEV 3	-	-	-	-
OracleVM	-	-	-	-

Linux UEFI loaders

Full-featured loaders:

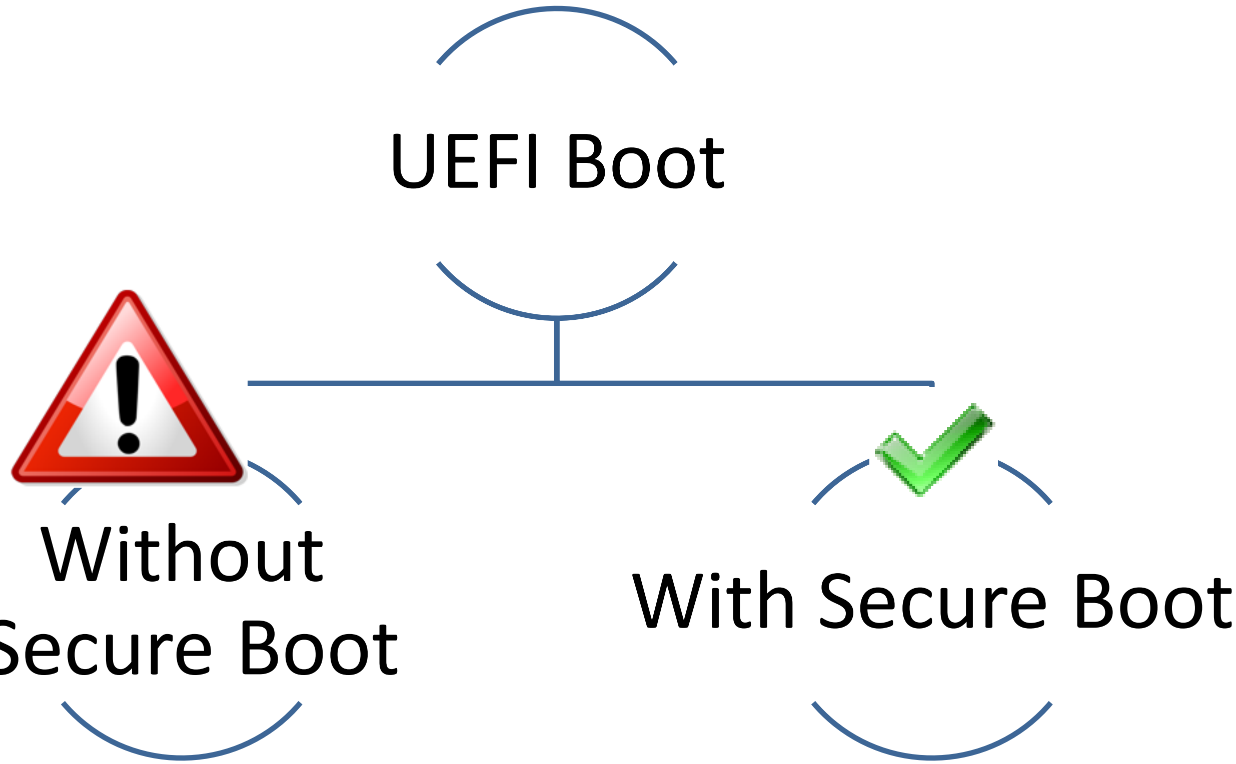
- GRUB
- ELILO
- EFIStub

Lightweight pre-loaders which load full-featured loaders:

- rEFInd
- EFILINUX
- Gummiboot
- PreLoader(Linux Foundation)
- Shim

Signed by Microsoft

UEFI Boot Process



Totally Unsecure, much more unsafe than old BIOS. See details in our paper.

The main UEFI loading mode and the most interesting case. **We'll present some results NOW**

UEFI Secure Boot: Attacks

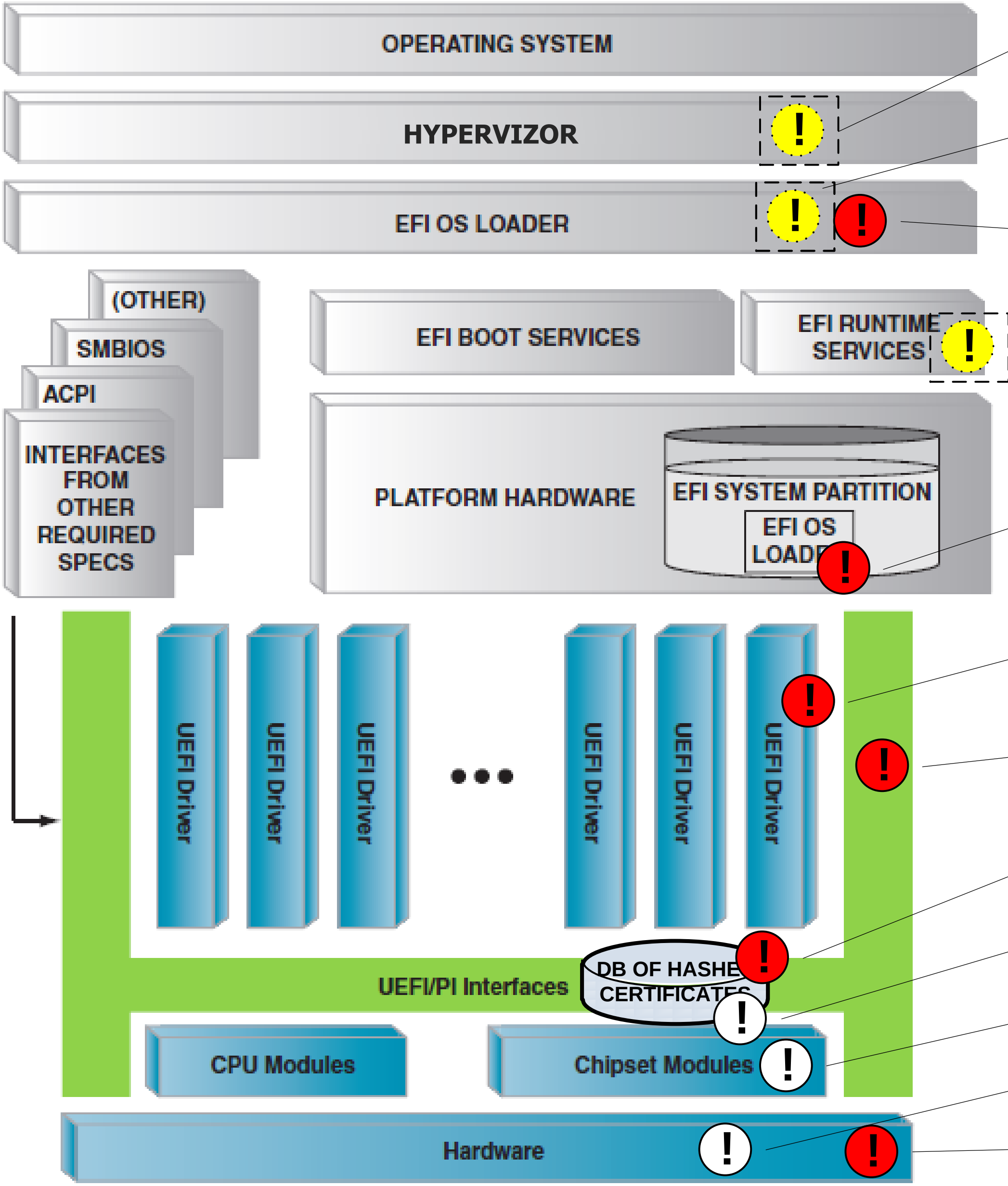
- The main goal of attack on secure boot is to avoid verification somehow and execute unsigned code.
- If this goal is achieved, the system works in “no secure boot mode” and becomes unprotected for even simple attacks

UEFI Secure Boot: Attacks 1/2

- **Disable Secure Boot (Illegally turn it off)**
  - Delete or corrupt PK EFI variable in NVRAM.
  - Change state of SetupMode and SecureBootEnable variables, stored in NVRAM.
- **Violate the integrity of Secure Boot**
  - For example, patch DxeImageVerificationLib library to change the verification policies.

UEFI Secure Boot: Attacks 2/2

- **Execute code, signed by invalid keys**
  - Add invalid certificate or hash to a db variable, stored in NVRAM.
  - Now all images signed by that key shall pass verification
- **Execute code without signature verification**
  - Inject malware code to platform firmware or Option ROM
  - Execute malware code in compatibility with Legacy BIOS mode.



Threats from malware hypervisor created on top of legal signed version of Linux, Or injected using malware boot loader

Threats from running malware boot loader, signed by legal MOK-key, if this key was stolen

Threats from malware boot loaders. If malware boot loader hash is injected to trusted database (db). E.g. Linux Foundation loader allows adding hashes to trusted db and thus load unsigned kernels

Attacks on Runtime serivces. Bugs in implementation of Authenticated Write Accesds may be used to unauthorized keys update. UpdateCapsule() can be used for attack on firmware update.

Threats of bloken chain of trust. It is not known it chain of signature verification ends up with verification of OS kernel. Boot loader call eash other in order at system start up (Shim, Grub2), But it can't be guaranteed that digital signature of kernel itself is also verified.

Execution of malware DXE driver, If it's signature is verified after injection of illegal key to SecureBoot database, or if platform driver verification policy is ALWAYS_EXECUTE for this driver

Threats in UEFI firmware update mechanisms: Bugs in secure firmware update implementation may result in unathorized UEFI BIOS update

Threats from injection of illegal certificate or hash to db
Once illegal hash or certificate is placed to db, it is possible to execute malware driver or Boot Loader

Threats from hash collisions. E.g. findinf collisions for hashes in white list.

Threat from attacks on SMM Mode of processor

Attacks on TPM module (cold boot attack)

Threats from direct write to SPI flash: If SPI flash configuration is not locked, malware can be written directly to SPI flash

Treats danger level: High

Low danger Moderaty

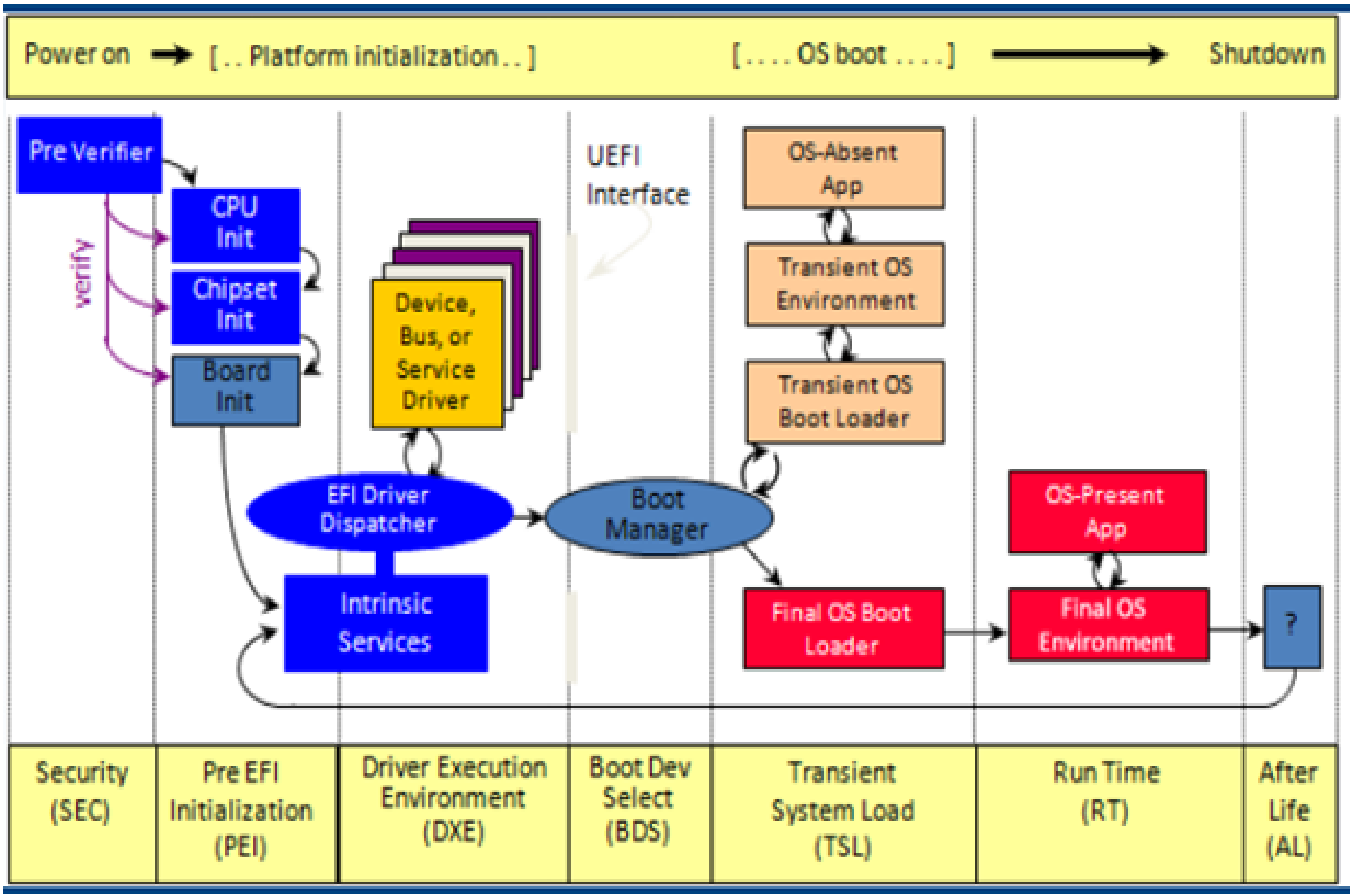
Future Work

- Implement and check new attacks on UEFI Secure Boot
- Make Demo
- Publish results

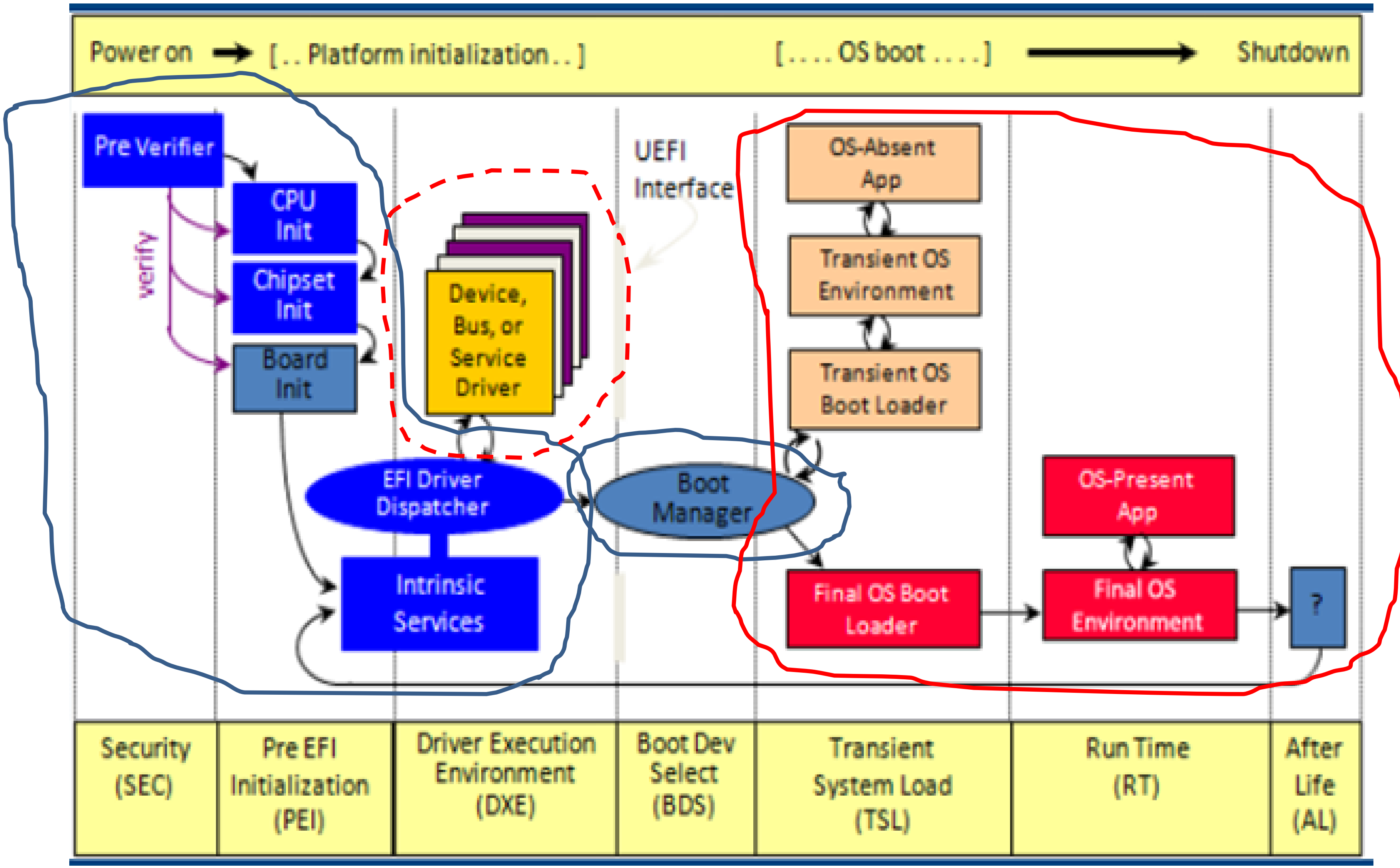
Thank you!

- Questions?

Platform Initialization



Platform Initialization



Loading without Secure Boot