tianocore

UEFI & EDK II Training

UEFI Aware Operating System

tianocore.org

LESSON OBJECTIVE

- Explain How the OS and UEFI Work together
- Explain the UEFI Requirements for UEFI aware OS
- Explain How Secure Boot Fits with UEFI

UEFI AWARE OS REQUIREMENTS

Common Requirements

tianocore

UEFI OPERATING SYSTEMS

Windows Server 2008 R2

Windows 10

Windows 7

Windows Storage Server 2008 R2 Enterprise

Microsoft

CANONICAL

yocto PROJECT

ubuntu

debian

SUSE

Clear Linux* Project

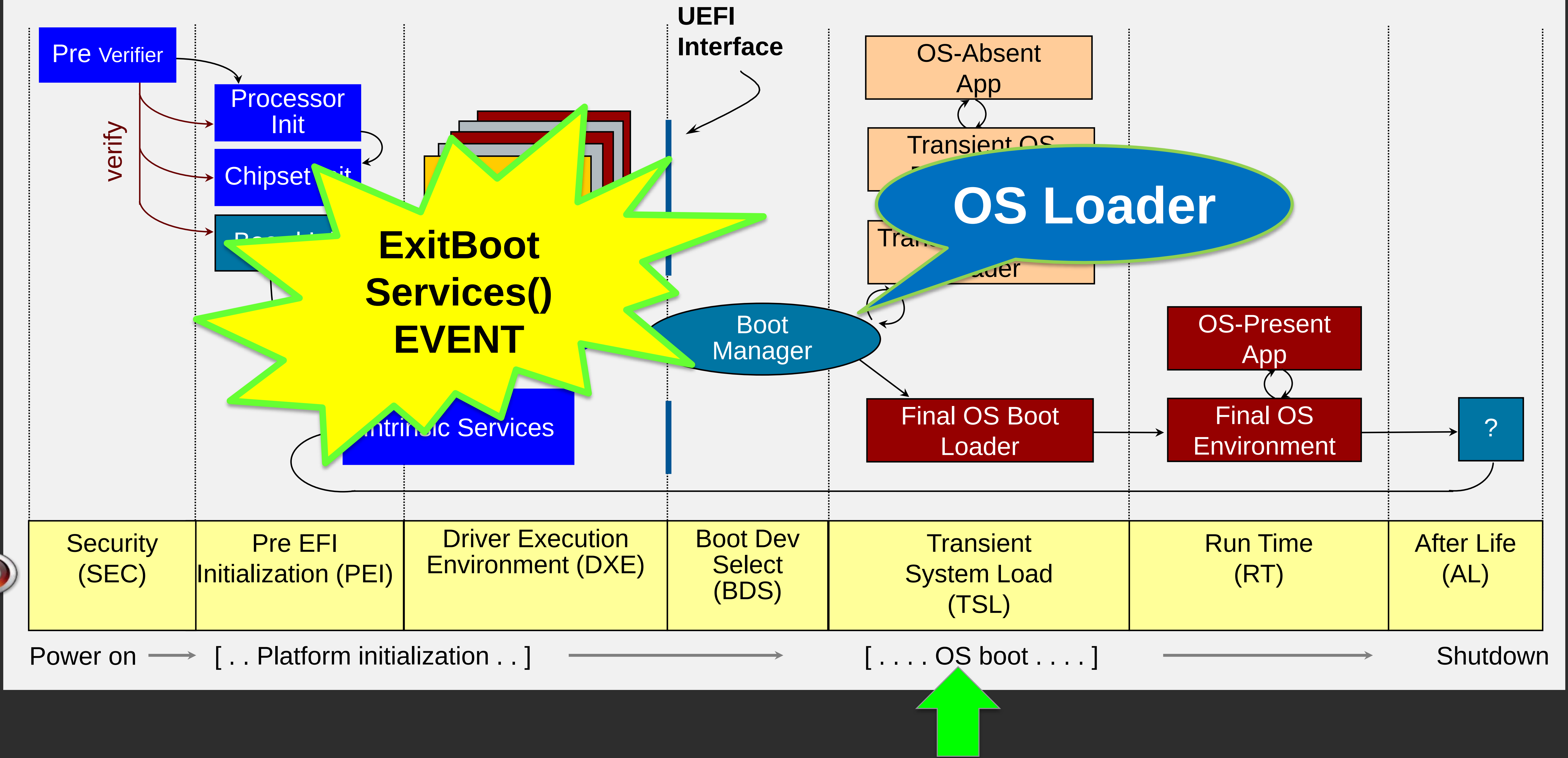
redhat.

Novell

fedora

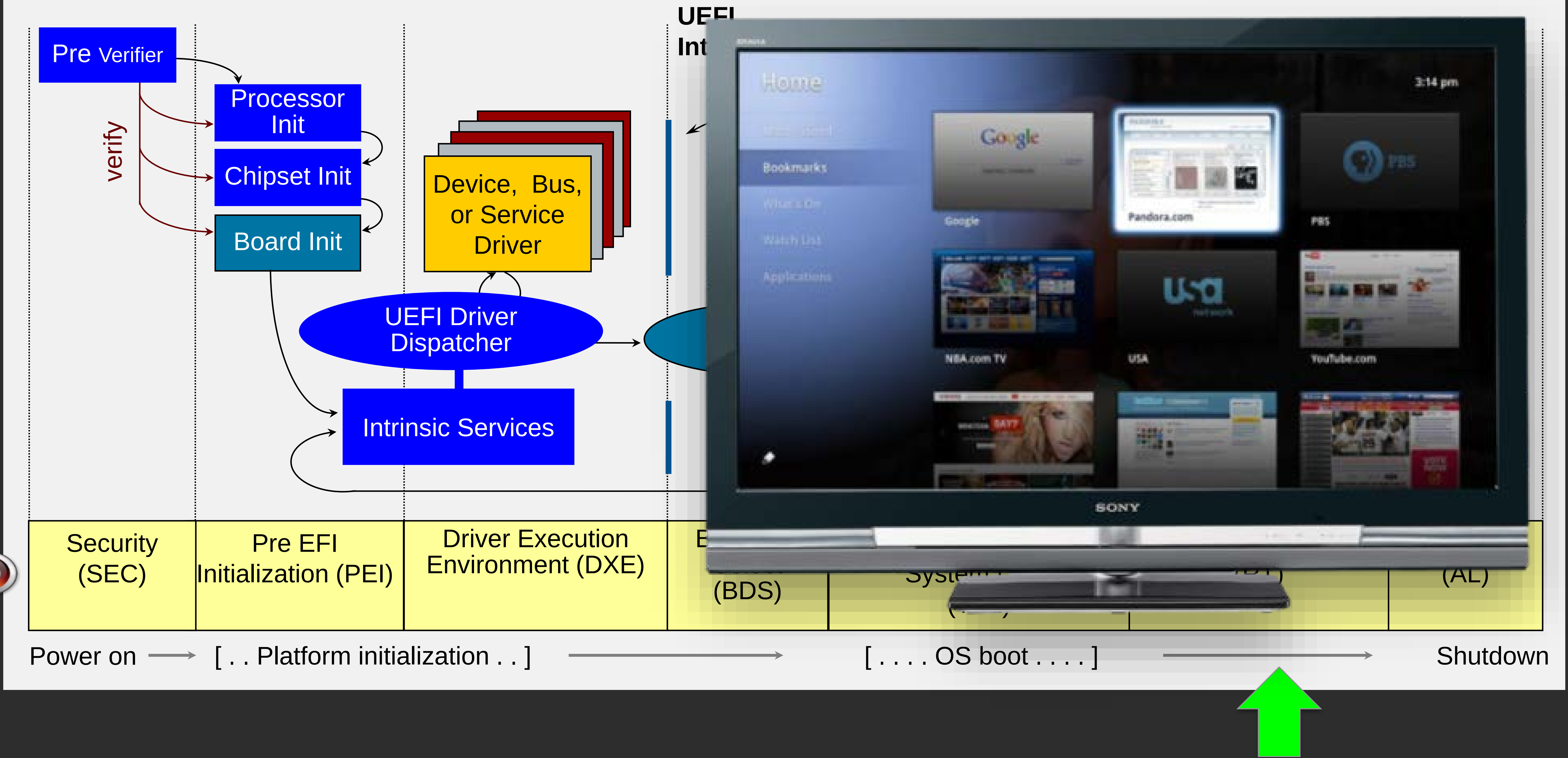
vmware

UEFI - PI & EDK II BOOT FLOW - REVIEW



tianocore

UEFI - PI & EDK II BOOT FLOW - REVIEW



UEFI OS REQUIREMENTS

UEFI Drivers:
Boot devices/console

UEFI OS installer

UEFI OS Loader

Disk Partition/Formats

Firmware Requirements

Set Boot Path to Boot to UEFI OS

UEFI System Classes (based on firmware interfaces)

UEFI Class 0

- Boots Legacy - int 19 ONLY
- Legacy BIOS Only (16 bit)
- No UEFI or UEFI PI Interfaces

UEFI Class 1

- Boots Legacy - int 19 ONLY
- Uses UEFI / PI Interfaces
- Only legacy BIOS runtime Interfaces

UEFI Class 2

- Boots Legacy - int 19 or UEFI
- Uses UEFI / PI Interfaces
- Legacy BIOS runtime Interfaces w/ **CSM**

Limited Benefits

✓ OEMs / ODMs Internal
✓ Double code development
✓ Compromised security – MBR exposure

UEFI System Classes (based on firmware interfaces)

UEFI Class 0
- Boots Legacy - int 19 ONLY
- Legacy BIOS Only (16 bit)
- No UEFI or UEFI PI Interfaces

UEFI Class 1
- Boots Legacy - int 19 ONLY
- Uses UEFI / PI Interfaces
- Only legacy BIOS runtime Interfaces

UEFI Class 2
- Boots Legacy - int 19 or UEFI
- Uses UEFI / PI Interfaces
- Legacy BIOS runtime Interfaces w/ **CSM**

UEFI Class 3
- Boots **ONLY** UEFI
- Uses UEFI / PI Interfaces
- Runtime exposes only UEFI interfaces

UEFI System Classes (based on firmware interfaces)

Full Benefits

- ✓ UEFI Innovation
- ✓ Smaller code size/ Validation
- ✓ Extensibility

Only Class after 2020

Enabling *Secure Boot* creates another Class

UEFI Class 3 +

- Boots **ONLY** UEFI
- Uses UEFI / PI Interfaces
- Runtime exposes only UEFI interfaces

UEFI Secure Boot “ON”

Required UEFI Drivers: OS Install & Boot

- Boot Device
- Console Output
- Console Input
- NVRAM Driver

UEFI OS LOADER

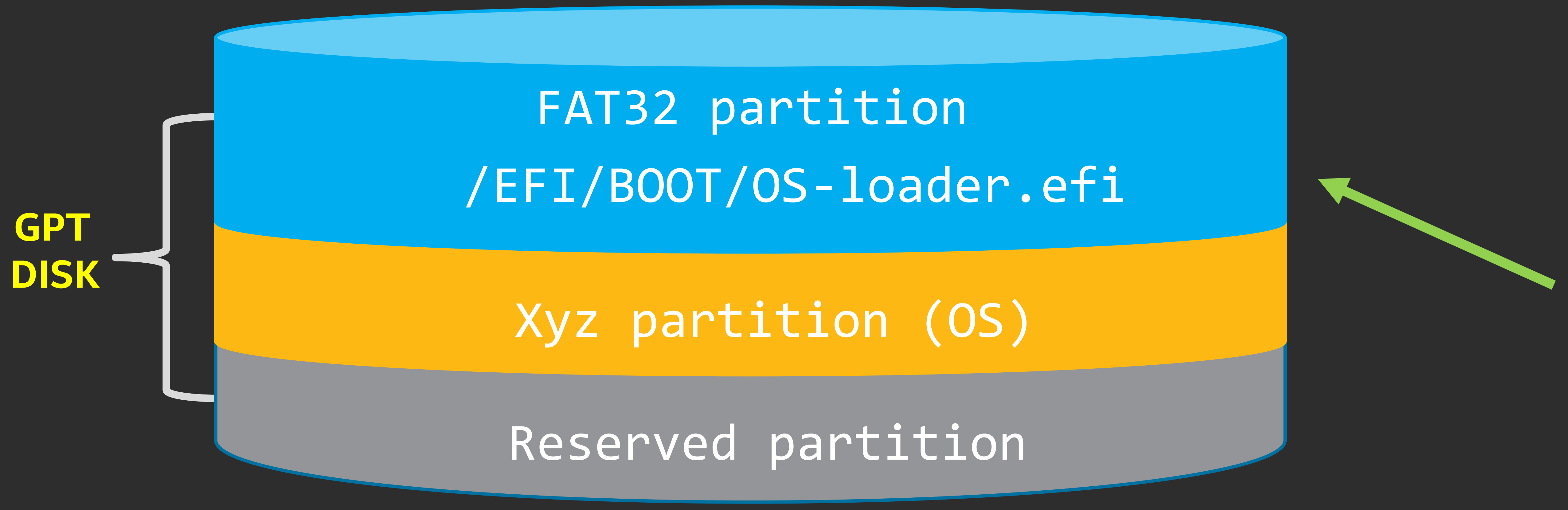
- OS install process includes UEFI loader

  `/efi/boot/bootx64.efi /efi/redhat/grub.efi`
- Call UEFI boot & runtime services to start OS
- Exit UEFI Boot Services
- Transfer control to native OS

UEFI OS INSTALLER

- Discover UEFI storage devices
- Setup storage device: GPT w/ FAT32 boot partition
- Create boot variables `BootXXXX` and set the `BootNext`

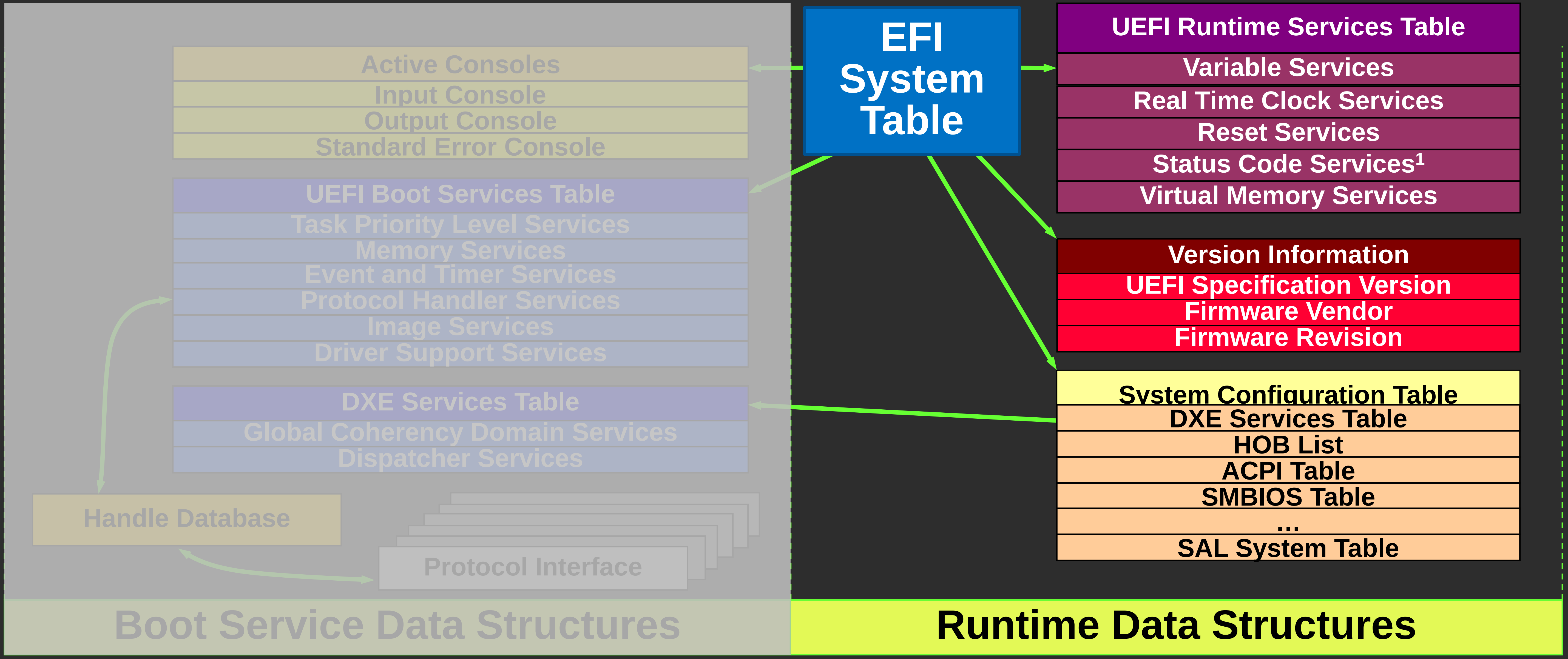
Disk Partition and Format



INTERFACE INSIDE OS RUNTIME

UEFI Runtime Services

Runtime Services Available to the UEFI Aware OS



Accessing RT services from Windows API

- GetFirmwareEnvironmentVariable: [MSDN Link]
- SetFirmwareEnvironmentVariable: [MSDN Link]
- Example: (determine if UEFI or Legacy BIOS)

```
 int main(int argc, char*argv[])
{
        GetFirmwareEnvironmentVariableA("",
           "{00000000-0000-0000-0000-000000000000}",NULL,0);
        if (GetLastError() ==ERROR_INVALID_FUNCTION) {
              printf("Legacy"); // This.. is.. LEGACY BIOS....
              return 1;
        } else{
              printf("UEFI"); // This.. is.. UEFI
              return 0;
        }
        return 0;
}
```

tianocore

Accessing RT services from Linux OS

Firmware Test Suite, it includes a Linux kernel driver to help with it's interactions with UEFI. Note that this is a Linux-centric test suite, solution won't work for other OSes.

- http://kernel.ubuntu.com/git/hwe/fwts.git
- https://bugs.launchpad.net/ubuntu/+source/linux/+bug/1633506
- https://patchwork.kernel.org/patch/9323781/
- http://www.basicinputoutput.com/2016/03/introduction-to-firmware-test-suite-fwts.html

SECURITY WITH UEFI

How does UEFI ensure the Operating System is trusted?

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

BOOT SECURITY TECHNOLOGIES

Hardware Root of Trust	Boot Guard, Intel® TXT
Measured Boot	Using TPM[1] to store hash values
Verified Boot	→ Boot Guard + UEFI Secure Boot

[1]TPM – Trusted Platform Module

Resources: https://firmwaresecurity.com/2015/07/29/survey-of-boot-security-technologies/

tianocore

HARDWARE ROOT OF TRUST

Boot Guard

CPU verifies signature

Verification occurs before system FW starts

Hash of public key is fused in CPU

Verification

Intel® TXT

Uses a Trusted Platform Module (TPM) & cryptographic

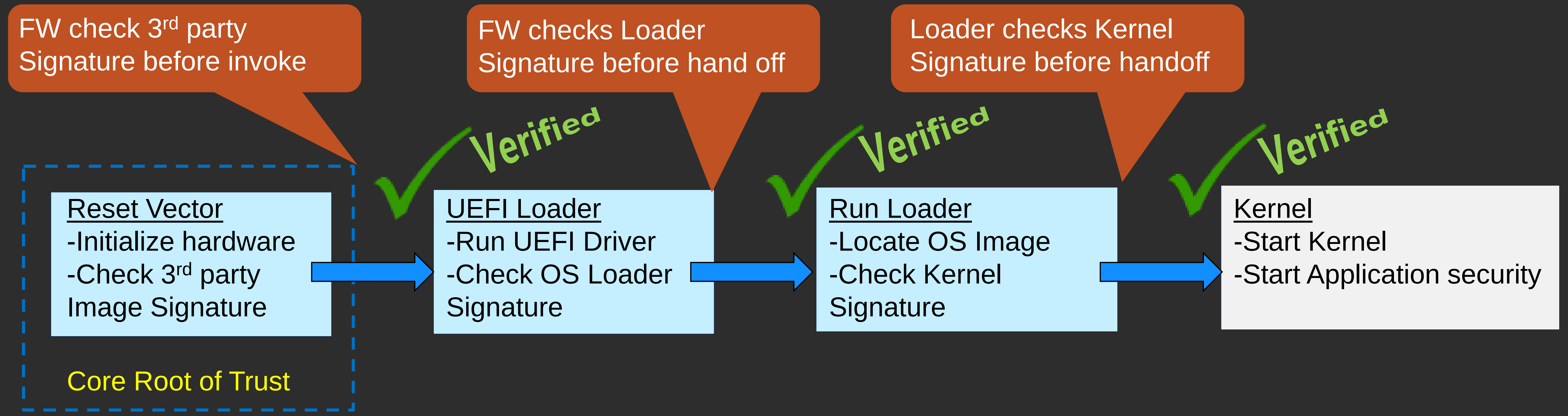
Provides Measurements

Measurements

UEFI SECURE BOOT

Software ID checking during every step of the boot flow:

1. UEFI System FW (updated via secure process)
2. Add-In Cards (signed UEFI Option ROMs)
3. OS Boot Loader (checks for “secure mode” at boot)



AUTHENTICATED VARIABLES

PK

KEK

DB

DBX

SetupMode

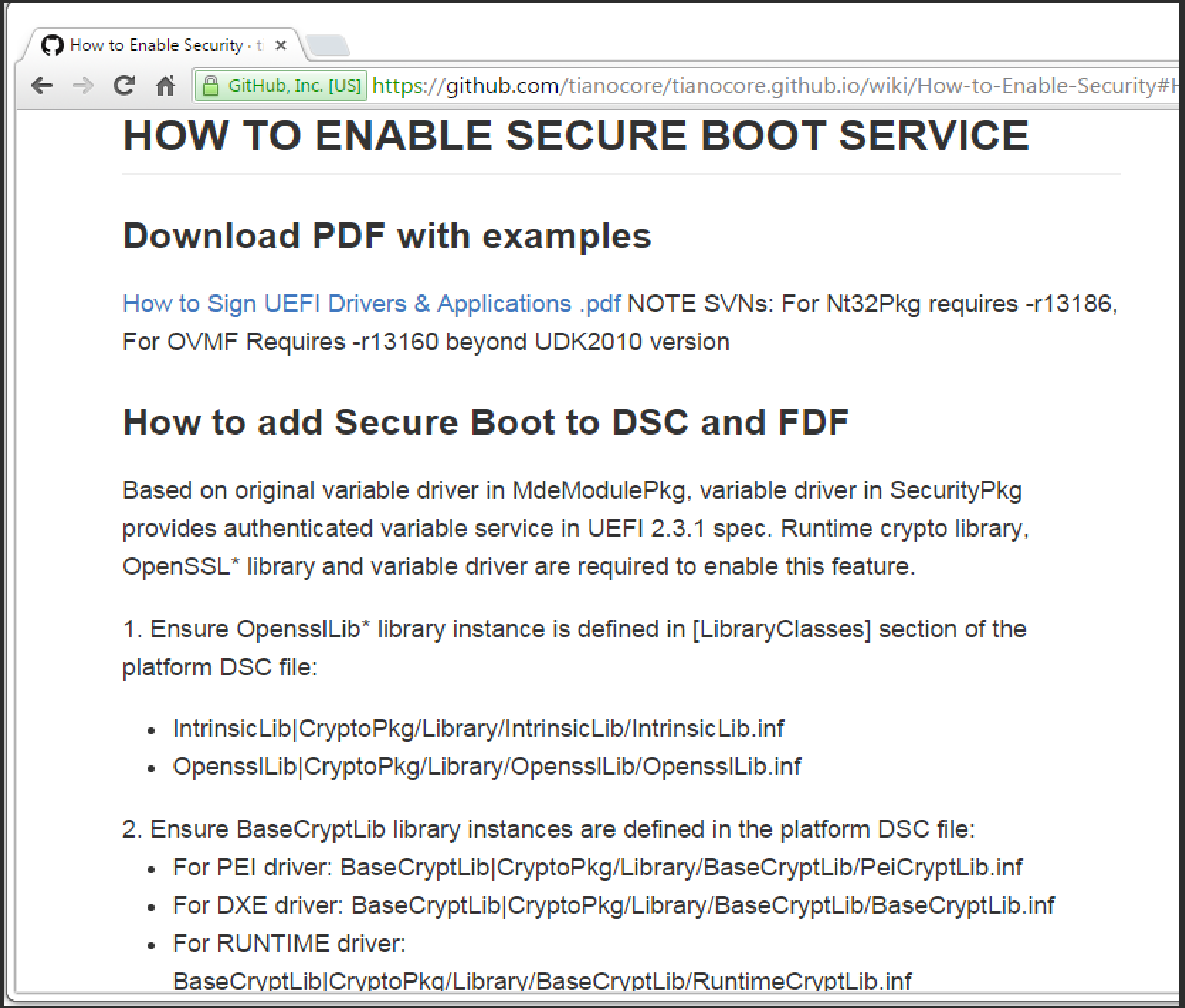
SecureBoot

```
2.0 Shell> dmpstore SecureBoot
Variable - RS+BS - '8BE4DF61-93CA-11D2-AA0D-00E098032B8C:SecureBoot' - DataSize
= 0x01
00:                                                00   *.*
```

Security Package Project Page Wiki Link

- Wiki Link: How-to-Enable-Security
- PDF: How to Sign UEFI Images V1.31
- Build command line switch - SECURE_BOOT_ENABLE = TRUE
- Install the OpensslLib CryptoPkg :

From edk2: “`git submodule update --init`”



How to Enable Security · t ×

GitHub, Inc. [US] https://github.com/tianocore/tianocore.github.io/wiki/How-to-Enable-Security#H

HOW TO ENABLE SECURE BOOT SERVICE

Download PDF with examples

How to Sign UEFI Drivers & Applications .pdf NOTE SVNs: For Nt32Pkg requires -r13186, For OVMF Requires -r13160 beyond UDK2010 version

How to add Secure Boot to DSC and FDF

Based on original variable driver in MdeModulePkg, variable driver in SecurityPkg provides authenticated variable service in UEFI 2.3.1 spec. Runtime crypto library, OpenSSL* library and variable driver are required to enable this feature.

1. Ensure OpensslLib* library instance is defined in [LibraryClasses] section of the platform DSC file:

- IntrinsicLib|CryptoPkg/Library/IntrinsicLib/IntrinsicLib.inf
- OpensslLib|CryptoPkg/Library/OpensslLib/OpensslLib.inf

2. Ensure BaseCryptLib library instances are defined in the platform DSC file:

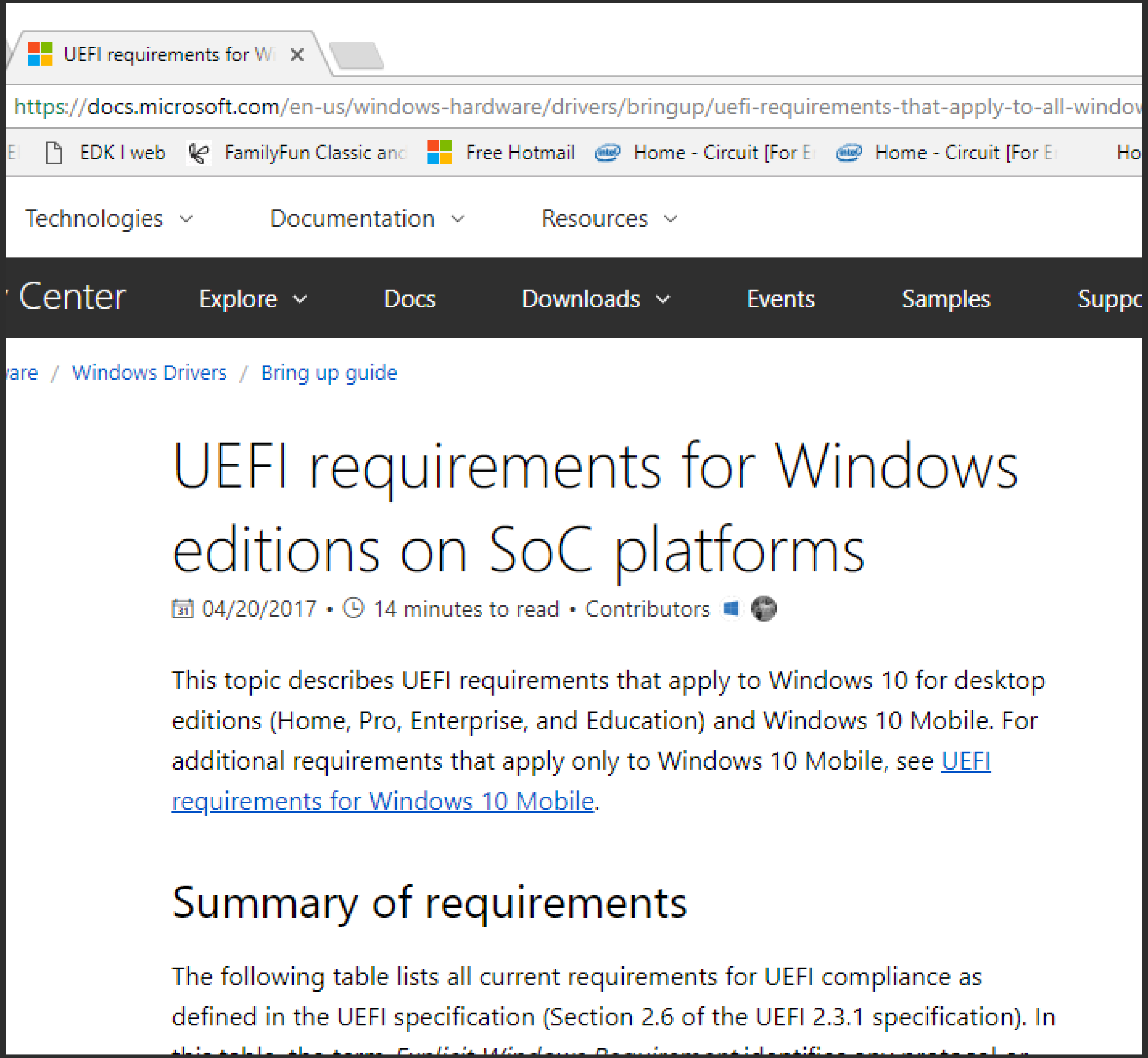
- For PEI driver: BaseCryptLib|CryptoPkg/Library/BaseCryptLib/PeiCryptLib.inf
- For DXE driver: BaseCryptLib|CryptoPkg/Library/BaseCryptLib/BaseCryptLib.inf
- For RUNTIME driver: BaseCryptLib|CryptoPkg/Library/BaseCryptLib/RuntimeCryptLib.inf

Windows Secure Boot Key Creation and Management Guidance

- Windows - Secure Boot Key Creation & Management Guide
- Creation and management of the Secure Boot keys and certificates in a manufacturing environment.
- Addresses questions related to creation, storage and retrieval of Platform Keys (PKs), secure firmware update keys, and third-party Key Exchange Keys (KEKs).

Many Platforms are Requiring UEFI Secure Boot Enabled

- Secure Boot now mandated for specific platforms
- See “Security requirements” on UEFI requirements for Windows editions on SoC Platforms



SUMMARY

- Explain How the OS and UEFI Work together
- Explain the UEFI Requirements for UEFI aware OS
- Explain How Secure Boot Fits with UEFI

Questions?

RETURN TO MAIN TRAINING PAGE

Return to Training Table of contents for next presentation link

tianocore

tianocore

ACKNOWLEDGEMENTS

BACKUP

UEFI SECURE BOOT

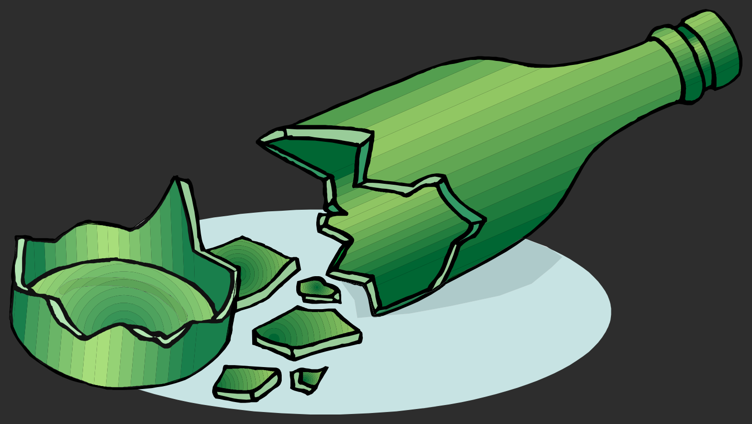
- Deficiency: Boot path malware targets
- UEFI and Secure Boot harden the boot process
- Firmware/software in the boot process must be signed by a trusted Certificate Authority (CA)
- Firmware image is hardware-protected
- 3rd party drivers signed using CA-holding trusted keys
- Trusted signing key's database factory-initialized and OS-updated

WHY??? SECURE BOOT WITH UEFI

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**

UEFI SECURE BOOT FLOW