From BIOS to UEFI

Understanding the Modern Firmware Attack Surface

BJÖRN RUYTENBERG
VRIJE UNIVERSITEIT AMSTERDAM

Hardware Security Guest Lecture – Dec 12, 2025

Who Am I

Björn Ruytenberg

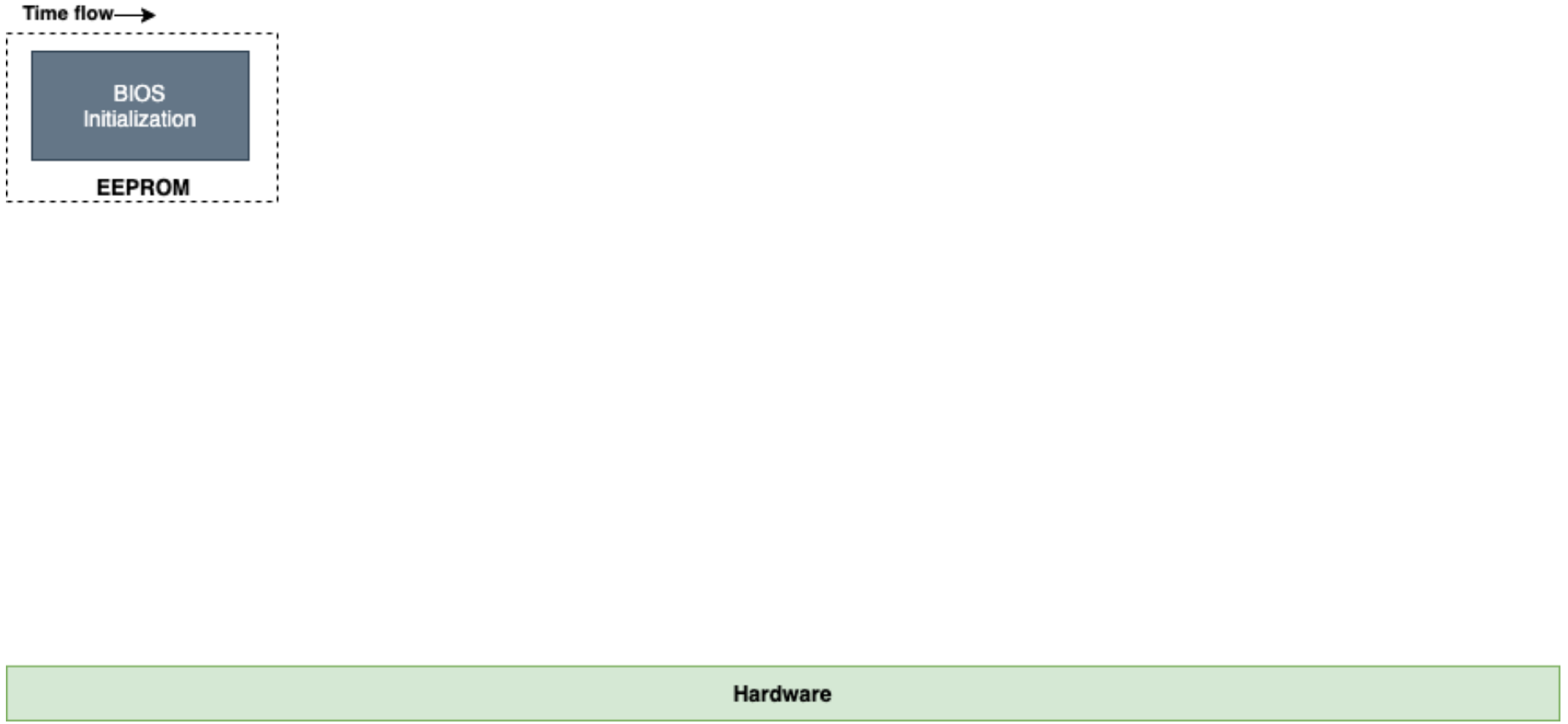
@0Xiphorus

@0Xiphorus@infosec.exchange

- PhD Candidate @ Vrije Universiteit Amsterdam
- Security Researcher, HyperDbg lead maintainer
- UEFI, hypervisor and PCI Express security
- Previous work: Intel Thunderbolt vulnerability research (thunderspy.io), sandbox escapes (Chrome, Firefox, Microsoft Office, Adobe)
- More info: bjornweb.nl

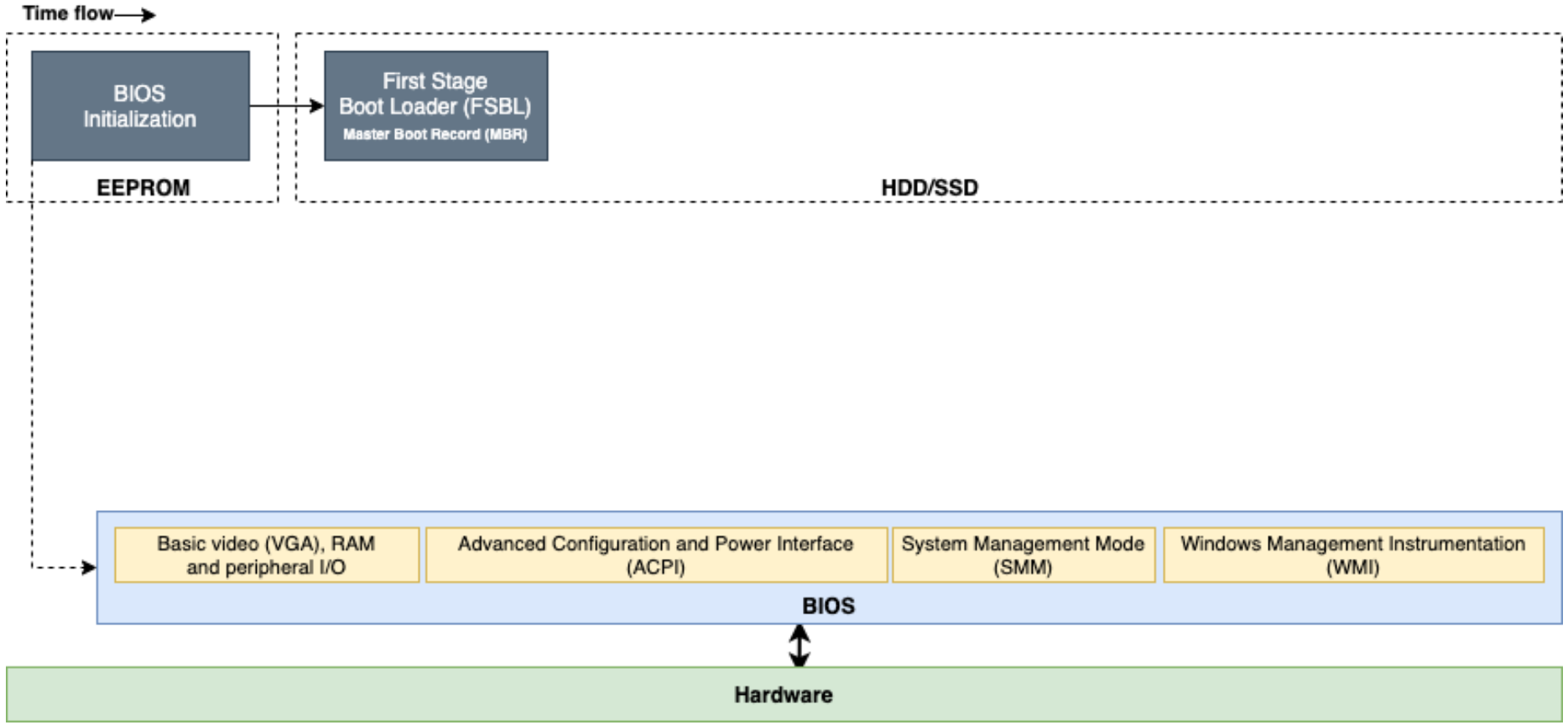
Background – BIOS Booting (MBR)

Legacy Boot



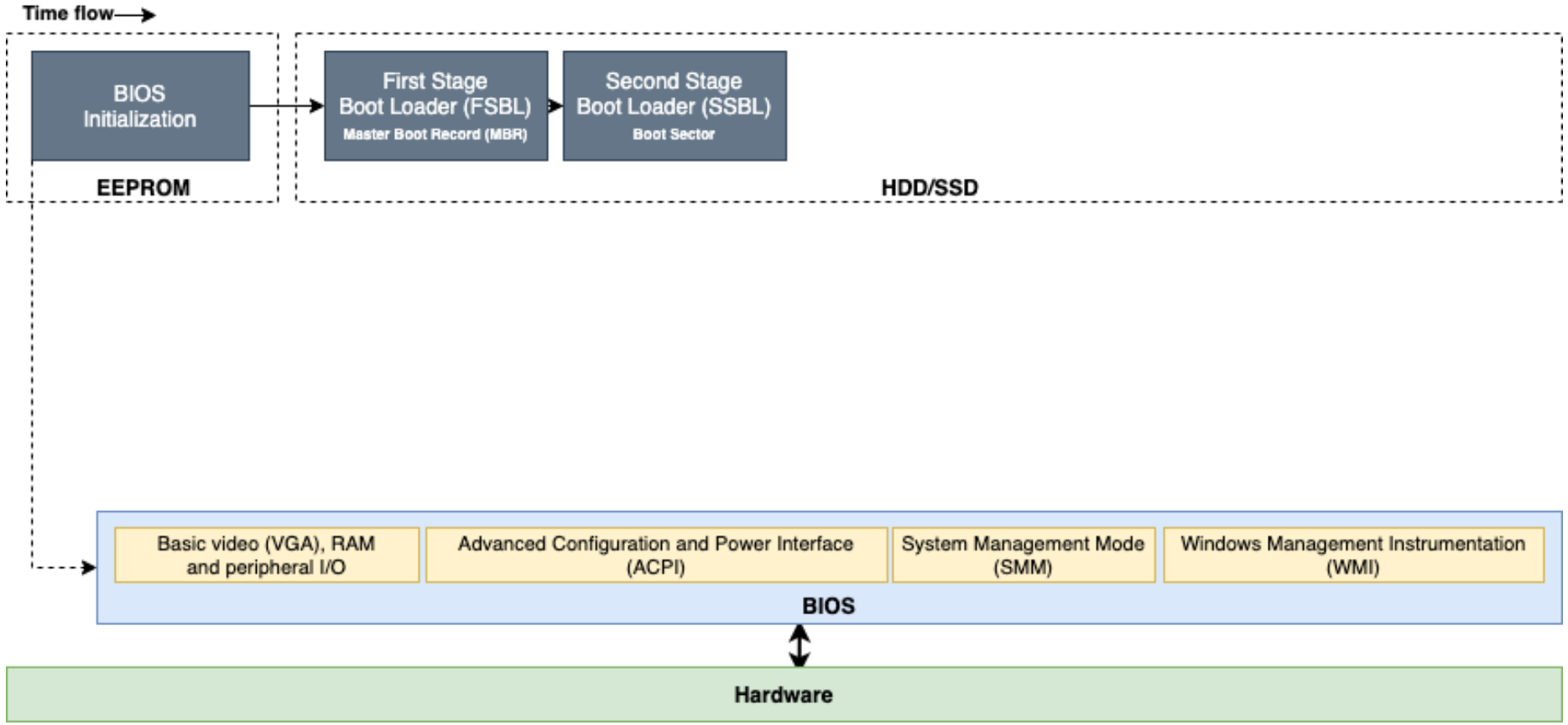
- CPU is powered on, loads Basic Input/Output System (BIOS)

Legacy Boot



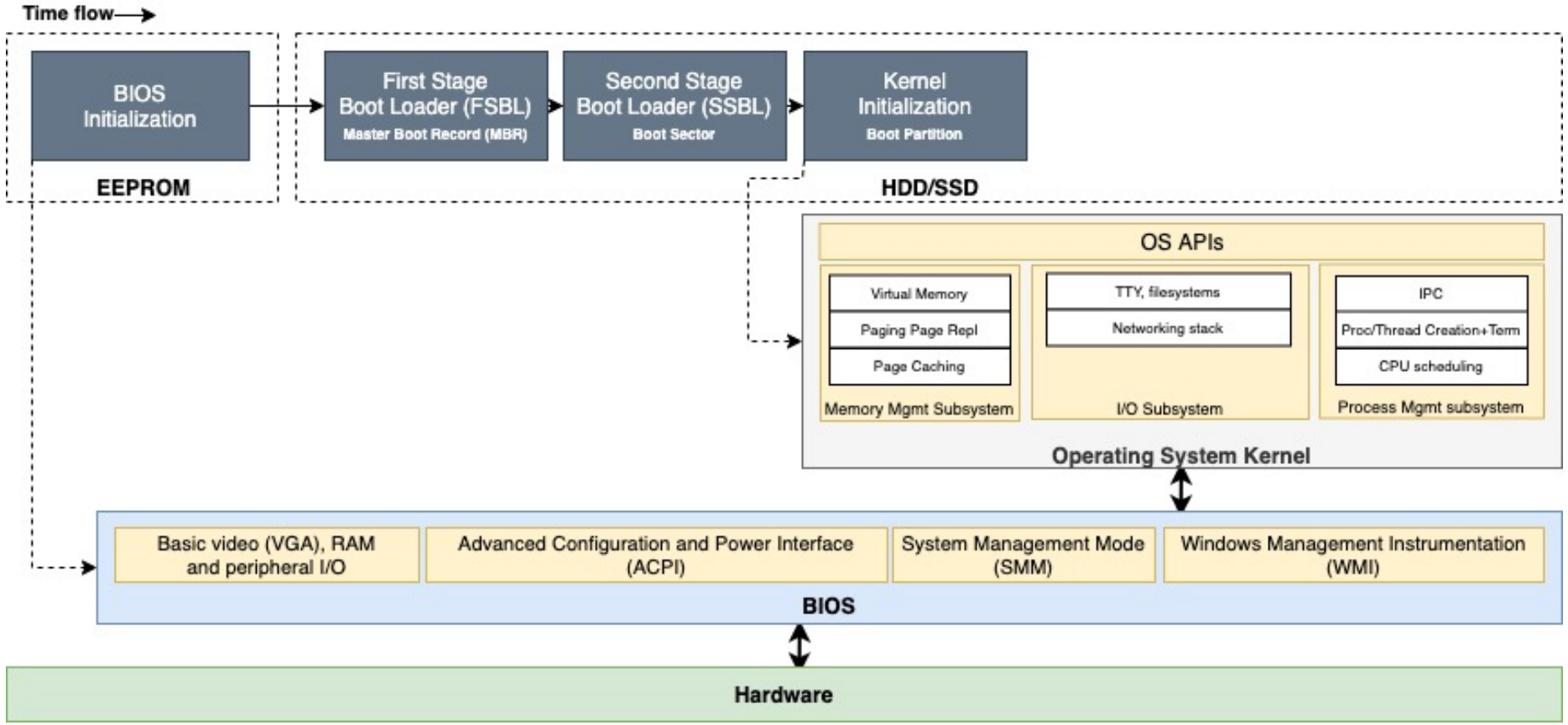
- BIOS
  - Performs basic hardware initialization (e.g. RAM, GPU, storage) and testing (Power On Self Test)
  - Enumerates hardware capabilities and provides basic abstractions to OS, including ACPI, SMM, WMI
  - Loads Master Boot Record (MBR), executes embedded First Stage Boot Loader (e.g. GRUB/NTLDR)

Legacy Boot



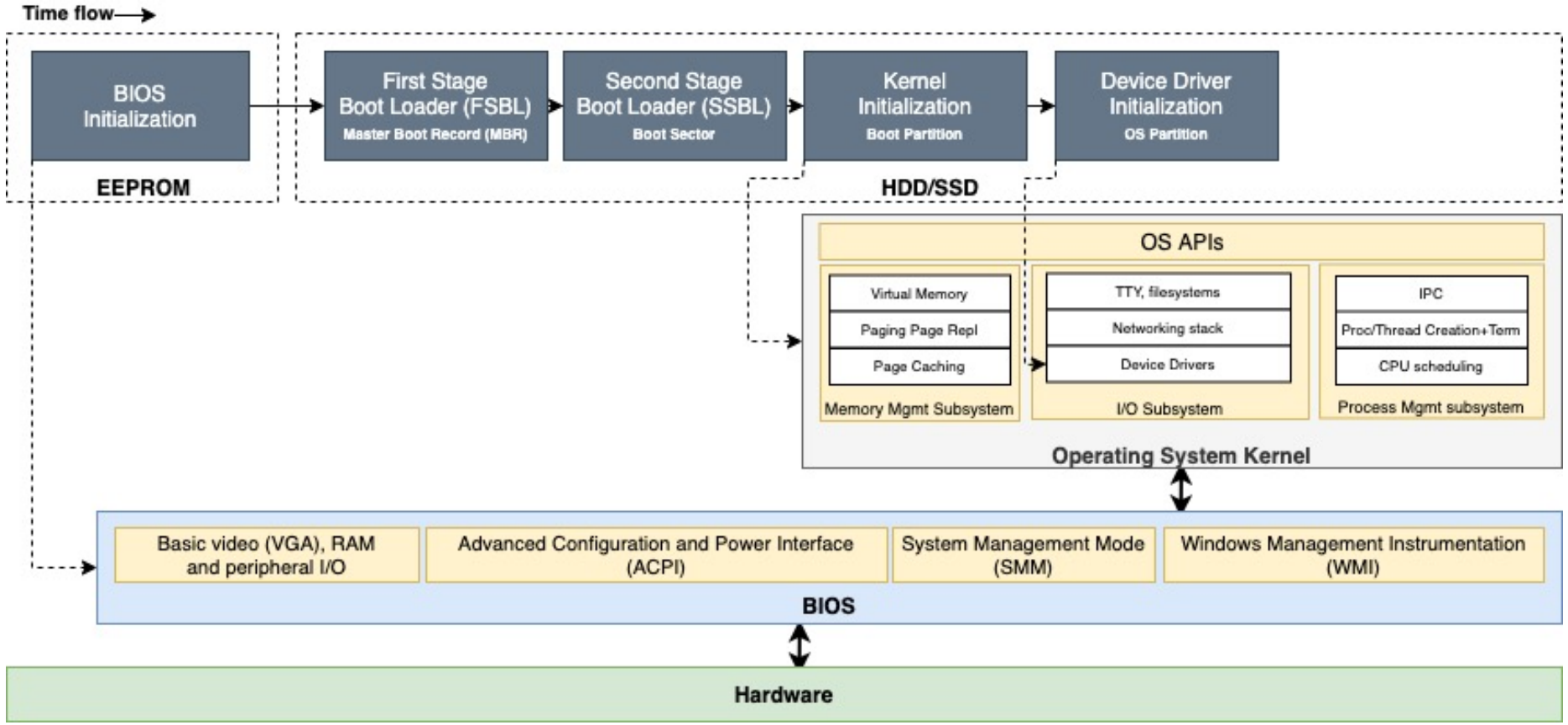
- First Stage Boot Loader
  - Mounts filesystem on first bootable partition
  - Loads SSBL into RAM (e.g. second stage GRUB/NTLDR)

Legacy Boot



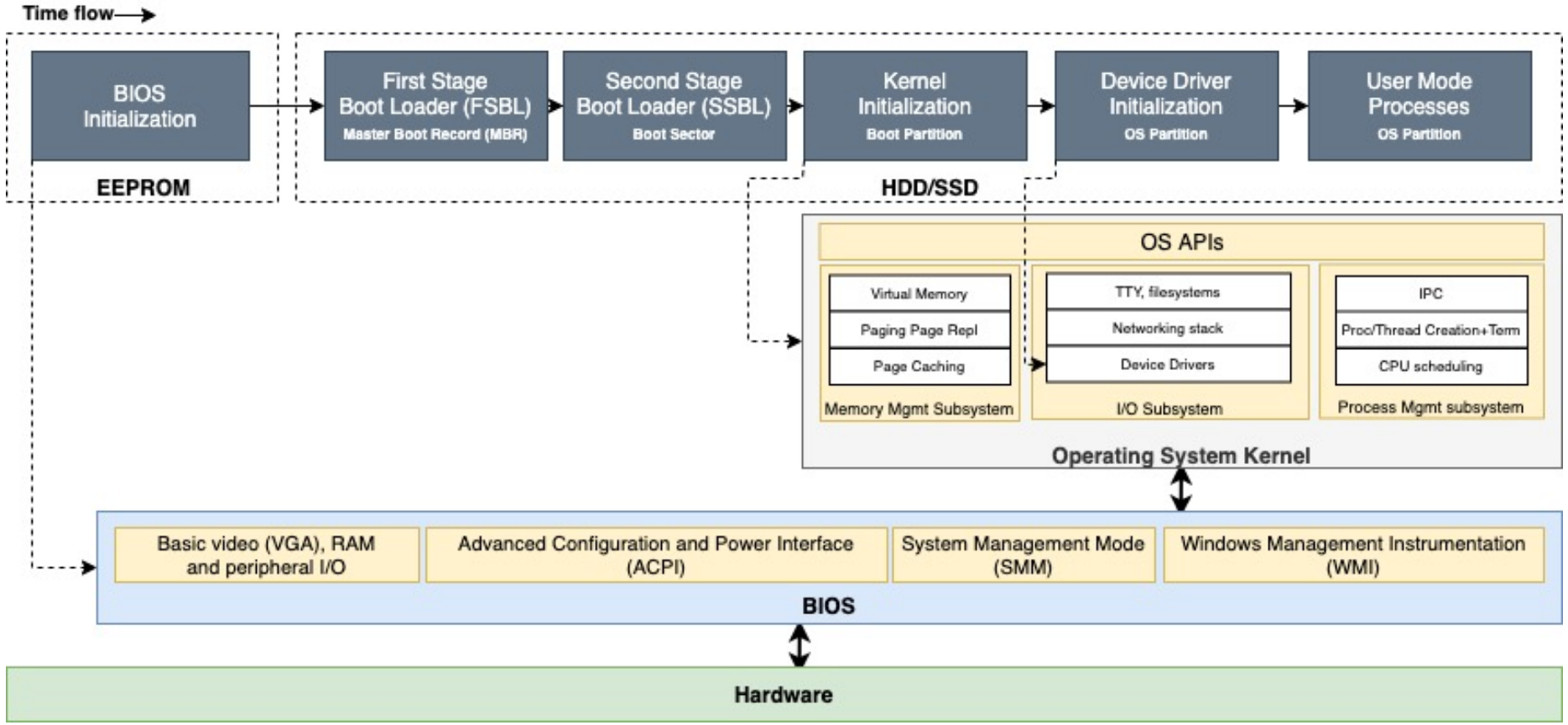
- Second Stage Boot Loader
  - Mounts OS filesystem, loads kernel into RAM, executes kernel

Legacy Boot



- Operating System kernel
  - Initializes memory management, I/O, process management subsystems
  - Loads device drivers, enabling full peripheral functionality

Legacy Boot



- Operating System kernel
  - Launches user mode processes

Legacy Boot – Limitations

- First BIOS implementations date from 1970's
- Legacy technology renders code base difficult to maintain
  - 16-bit "real mode" x86 assembly
  - Typically limited to 1 MB EEPROM
  - MBR imposes inflexible and resource-constrained boot process
    - FSBL limited to 440 bytes, need to resort to SSBL for further system initialization
    - Partition table limited to 4 partitions
    - Uses 32-bits for logical block addressing, limiting storage to 2 TB (16 TB using "Advanced Format")
  - Monolithic architecture complicates adding new hardware support
  - Single-threaded code preventing parallelizing hardware initialization, leading to extended boot times
  - **No security:** boot process can be hijacked by malicious code

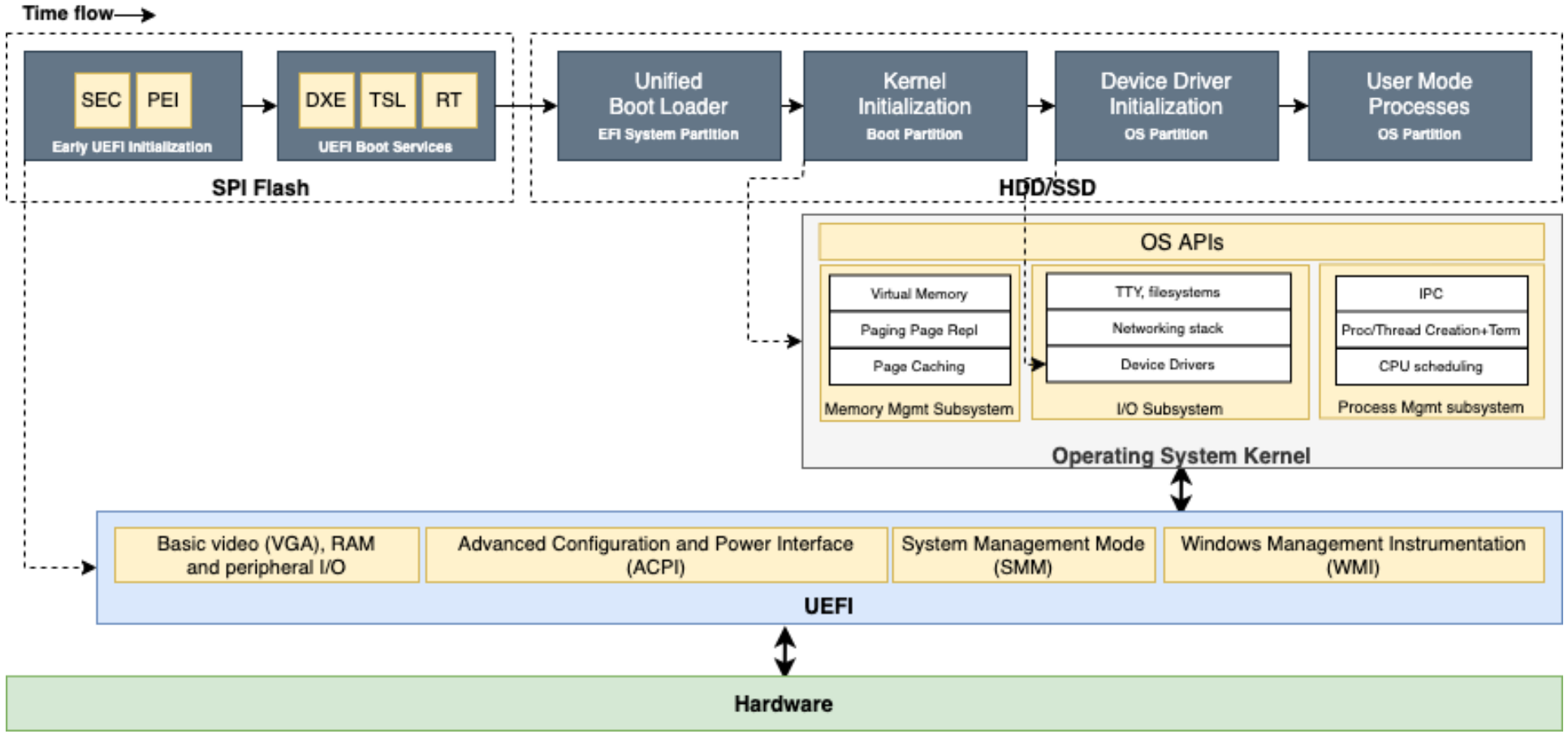
Background – UEFI Booting

UEFI Boot

Unified Extensible Firmware Interface (UEFI)

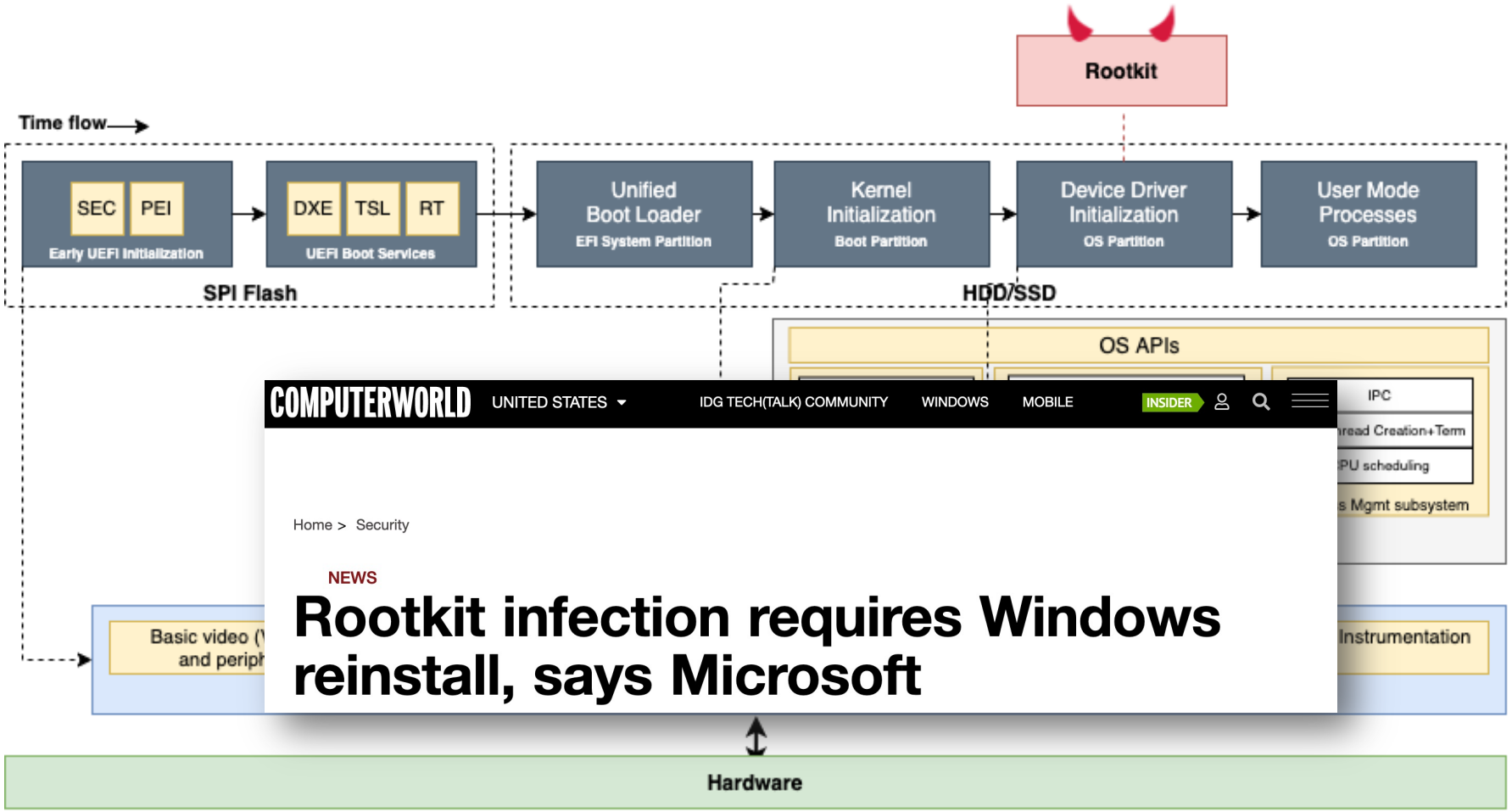
- Early versions developed exclusively by Intel (2000). Contributed to UEFI Forum and subsequently adopted by
  - Apple, for Intel Macs (2006 – today)
  - x86 OEM/ODMs (2011 – today)
  - Recent ARM and PowerPC implementations available, but not commonly used
- Stored on SPI flash, typically up to 64 MB
- GUID Partition Table (GPT) boot alleviates most MBR limitations
  - Unified boot loader binary stored on FAT32-formatted EFI System Partition (ESP)
  - Partition table enables up to 128 partitions
  - Uses 64 bits for logic block addressing – storage addressable up to 9.4 ZB
- Boot stages
  - Security Phase (SEC): Minimal assembly to initialize microcode and co-processors (Intel ME, AMD PSP, TPM)
  - Pre-EFI Initialization (PEI): Basic hardware initialization + POST, ACPI sleep/resume handling
  - Driver Execution Environment (DXE): initialize boot time device drivers
  - Transient System Load (TSL): Load user-selected boot loader from ESP
  - Runtime (RT): UEFI hands over control to OS boot loader
- Note: often still referred to as “BIOS” in vendor documentation, literature

UEFI Boot

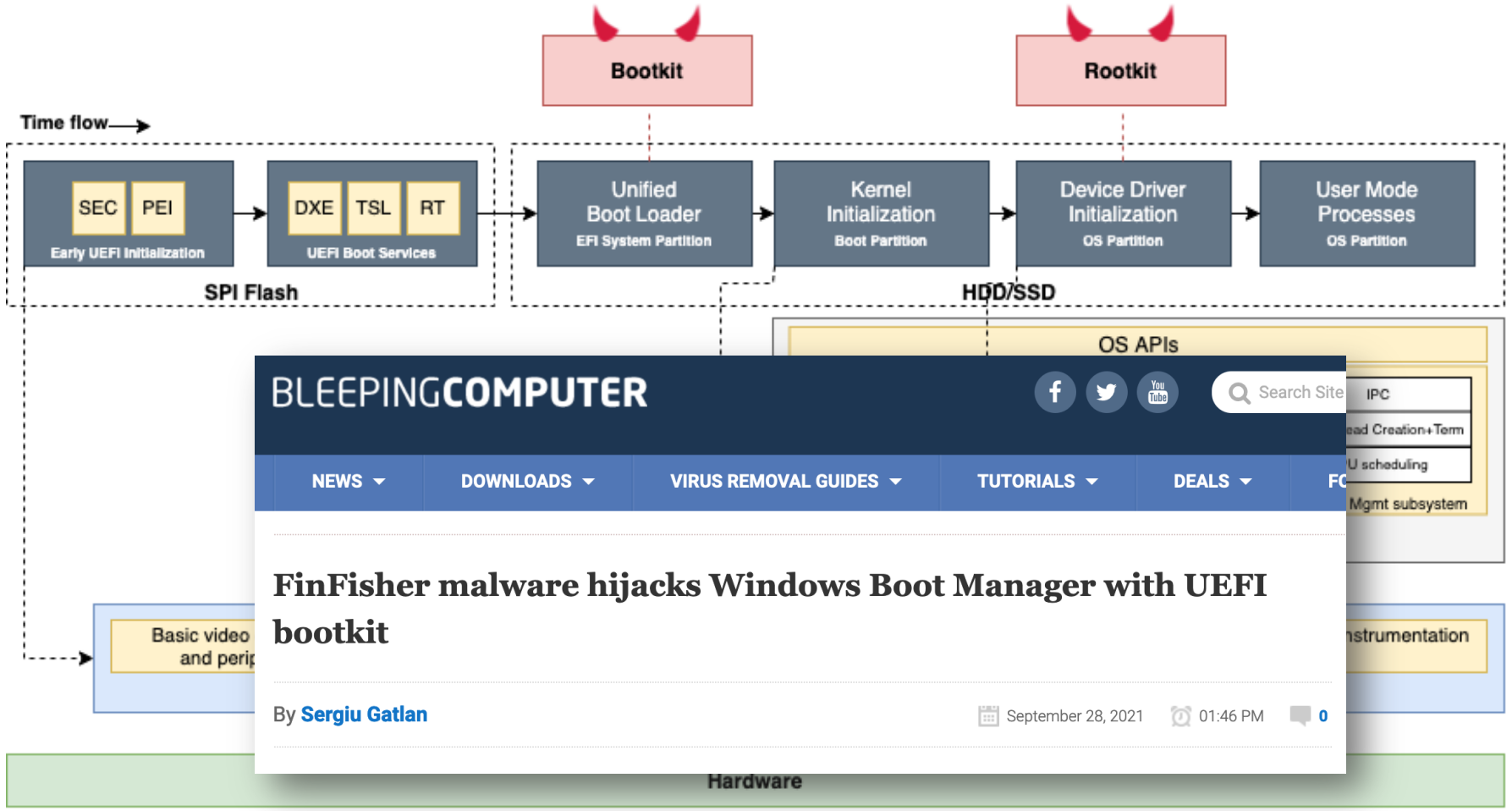


What about security?

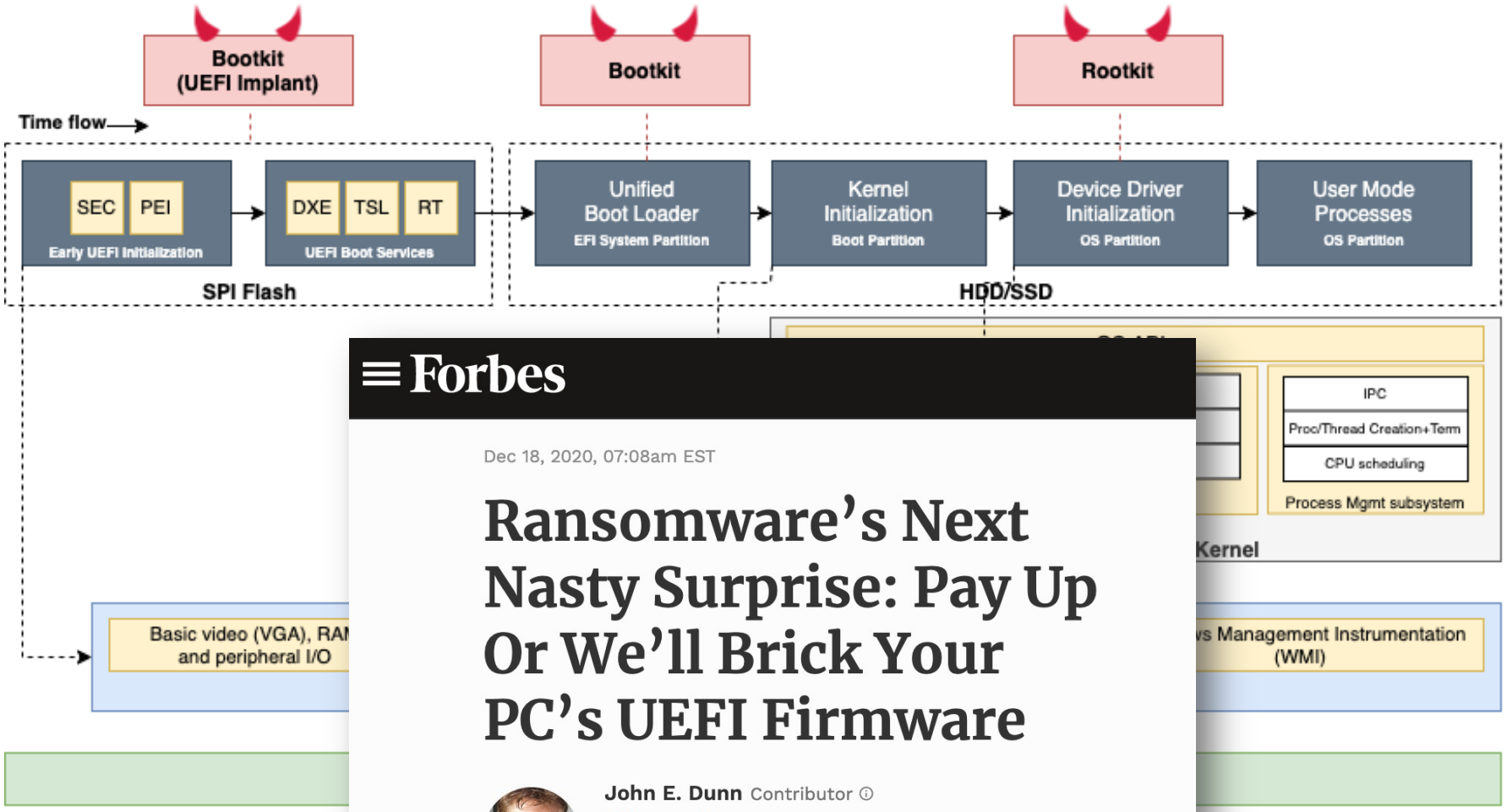
UEFI Boot – Attack Vectors



UEFI Boot – Attack Vectors



UEFI Boot – Attack Vectors



Forbes

Dec 18, 2020, 07:08am EST

Ransomware's Next Nasty Surprise: Pay Up Or We'll Brick Your PC's UEFI Firmware

John E. Dunn Contributor ⓘ

Cybersecurity

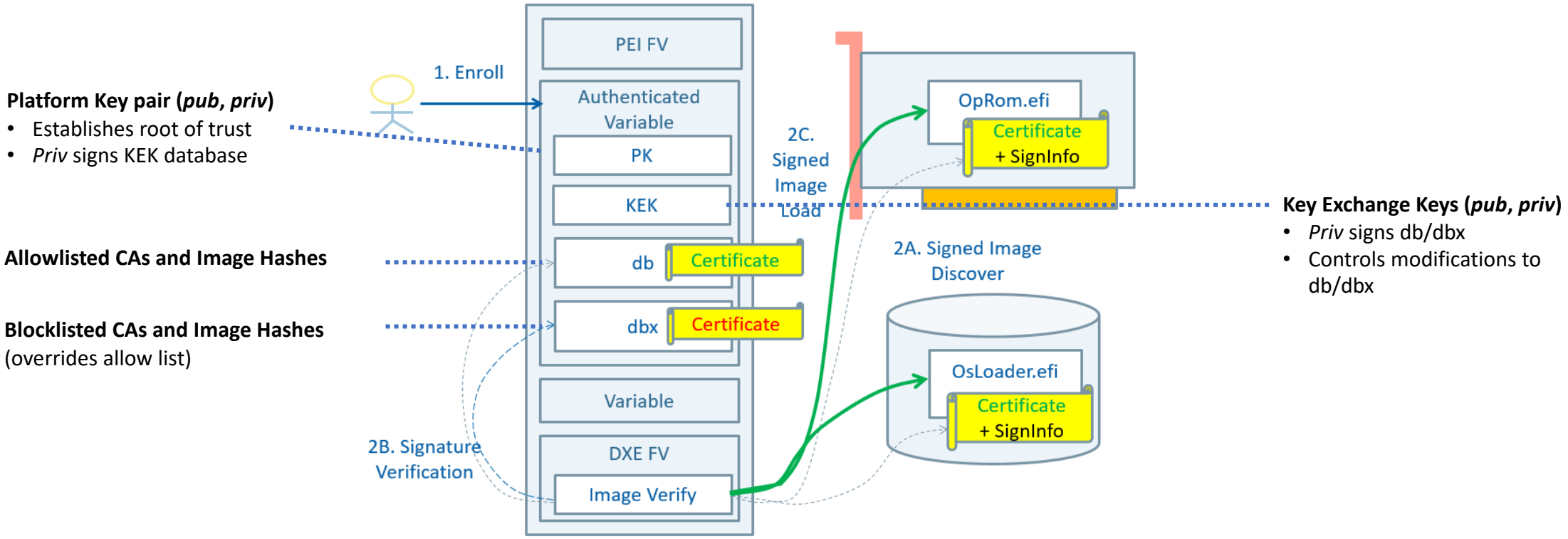
Trying to explain cybersecurity woes and why they matter

Addressing UEFI Boot Attack Vectors

- **Verified Boot / Boot Guard**
  - Protects against malicious firmware implants
  - Cryptographically verifies UEFI integrity
- **Driver signing**
  - Ensures driver authenticity (originates from device vendor) and integrity (tamper-resistance)
  - OS vendors require device vendors to submit drivers for evaluation
    - Linux (kernel.org): submit code for functional + security review
    - Windows: semi-blackbox testing through WHQL program
  - Upon passing verification
    - Linux: distribution vendors sign pre-compiled driver binaries
    - Windows: Microsoft signs driver's commercial release certificate; device vendor signs driver using private key to latter certificate
- **Secure Boot**
  - Protects against malicious, unsigned code early in boot process
  - Cryptographically verifies boot chain: OS bootloader, kernel, drivers

Secure Boot: A Closer Look

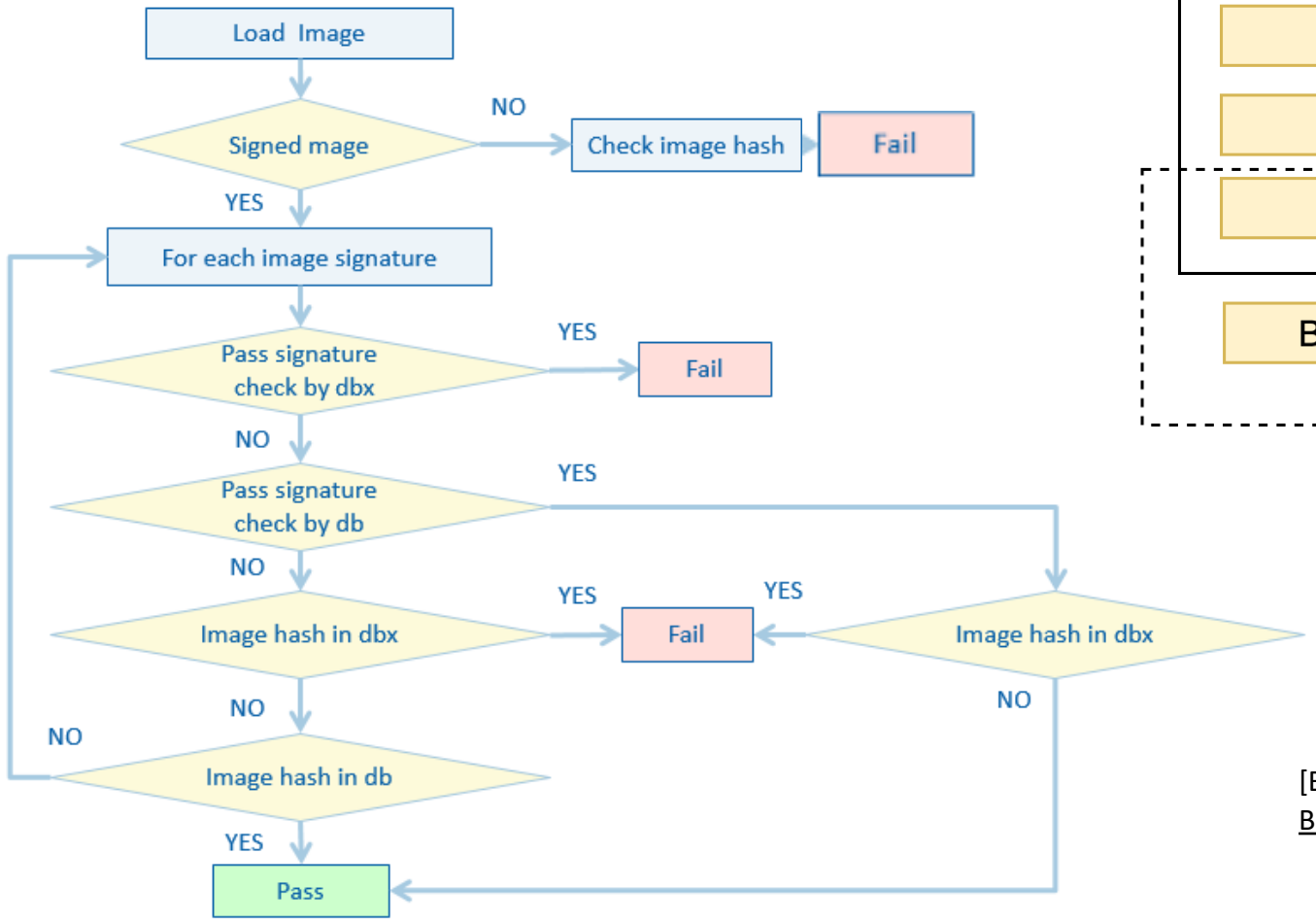
- Protects against malicious, unsigned code early in boot process
- Cryptographically verifies boot chain: OS bootloader, kernel, drivers

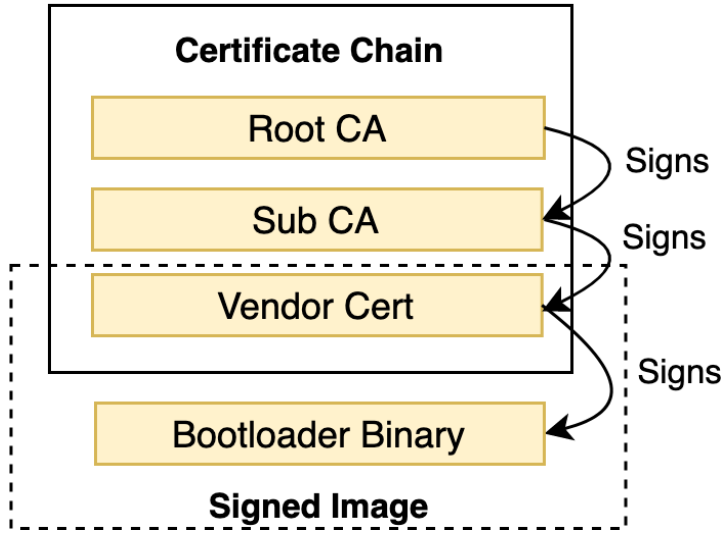


[Based on "Understanding the UEFI Secure Boot Chain" – TianoCore/EDK2]

Secure Boot: A Closer Look

- Protects against malicious, unsigned code early in boot process
- Cryptographically verifies boot chain: **OS bootloader**, kernel, drivers

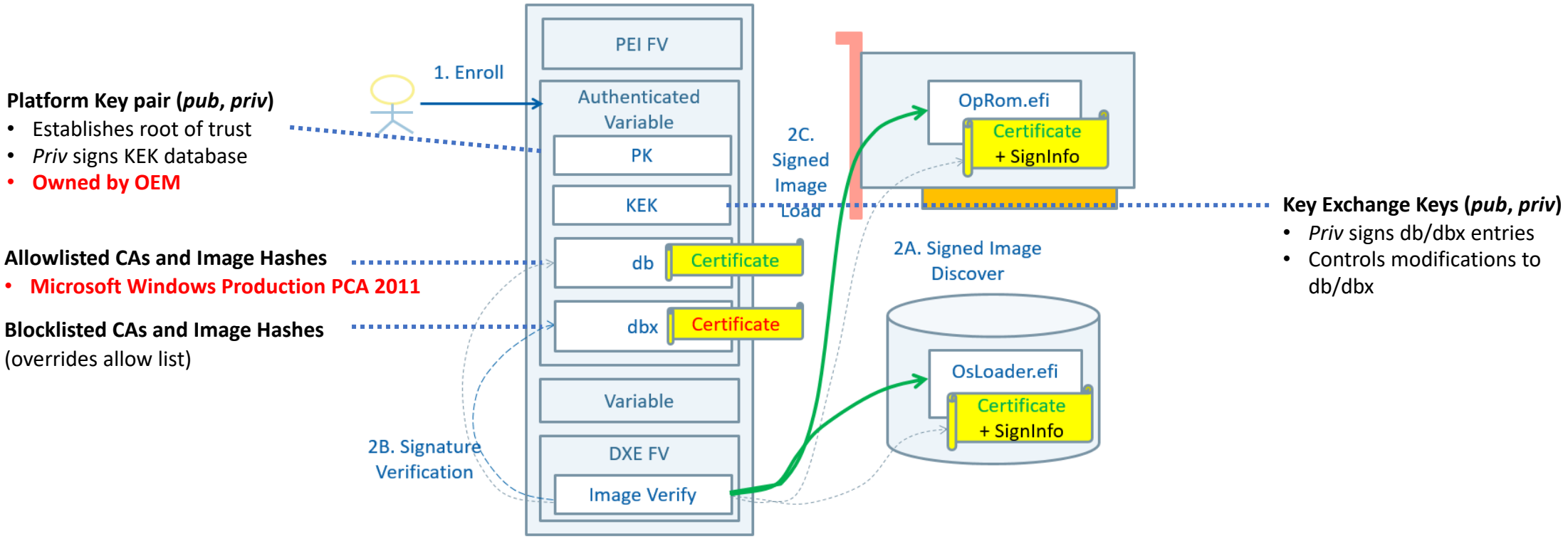




[Based on "Understanding the UEFI Secure Boot Chain" – TianoCore/EDK2]

Secure Boot: A Closer Look

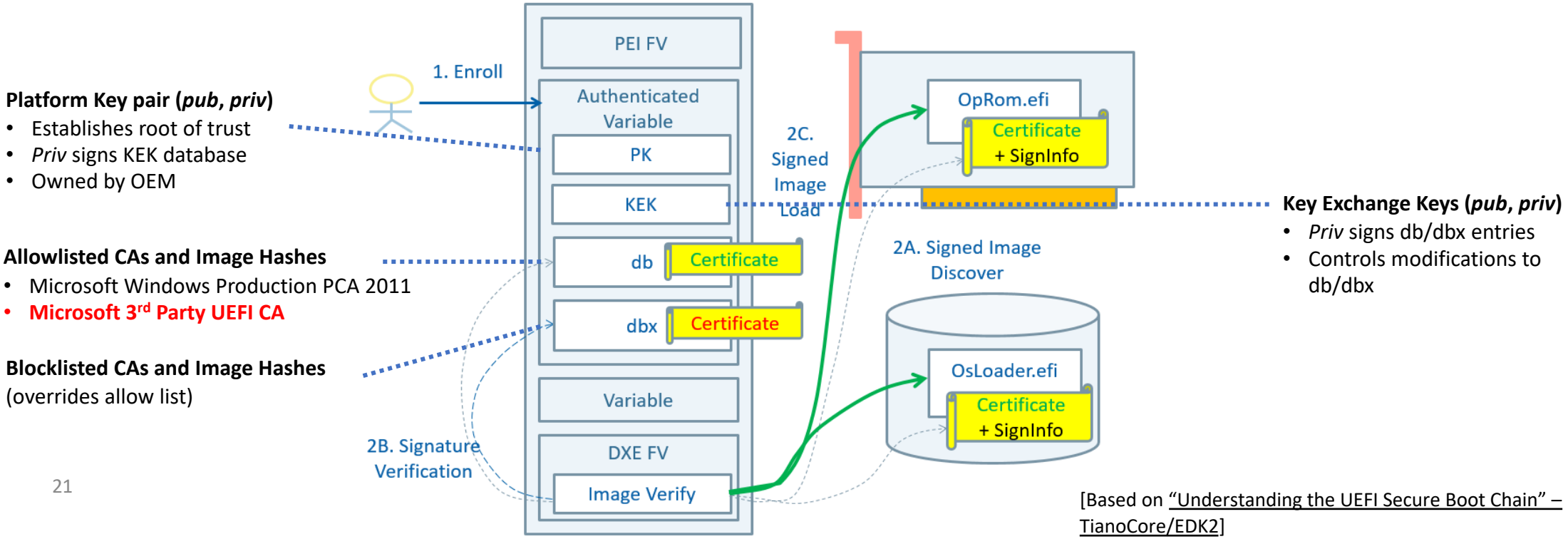
- Secure Boot trust model similar to WebPKI, but with significant exceptions
  - OEM-controlled PK; Root CAs exclusively controlled by Microsoft
- Raises concerns on Secure Boot chain ownership



[Based on "Understanding the UEFI Secure Boot Chain" – TianoCore/EDK2]

Secure Boot: A Closer Look

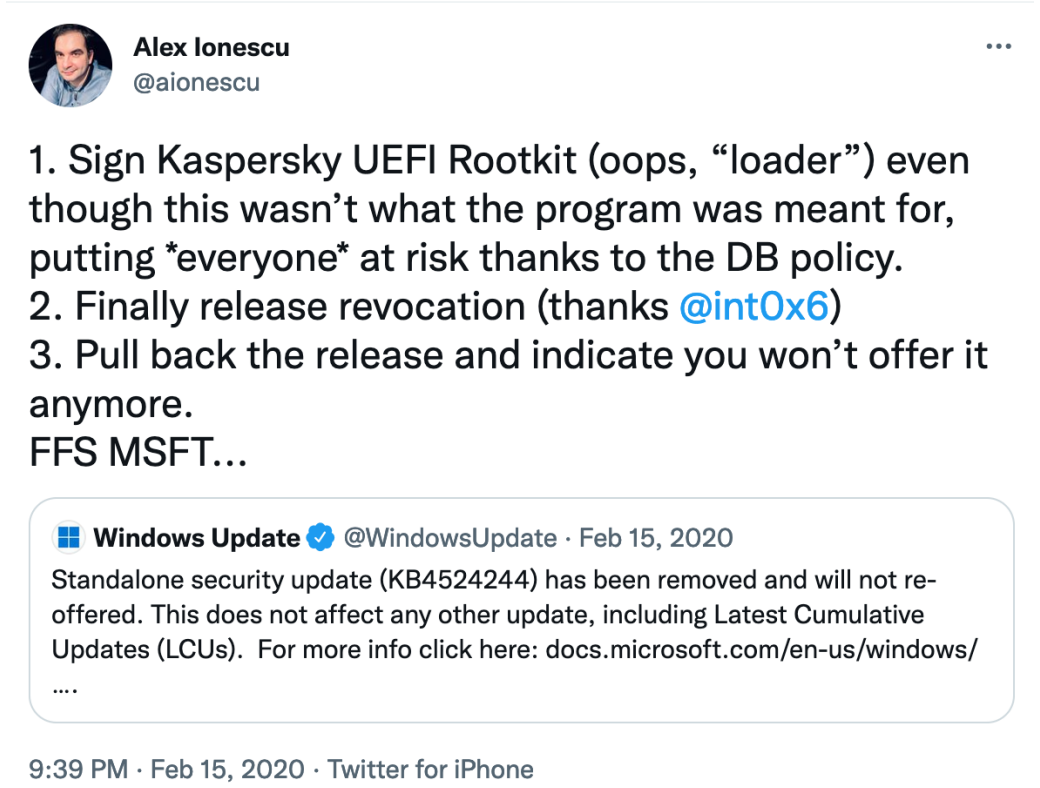
- Recent UEFI implementations enable user-replaceable PK
- Secondary Microsoft Root CA signs third party boot loaders
  - Red Hat: SHIM first stage boot loader
    - Used to chainload e.g. GRUB2, systemd-boot, EFI Stub
    - Enables users to self-sign boot loaders + drivers using “Machine Owner Key” (MOK)
  - Anti-malware and imaging solutions, e.g. live USB flash drive environment



[Based on “Understanding the UEFI Secure Boot Chain” – TianoCore/EDK2]

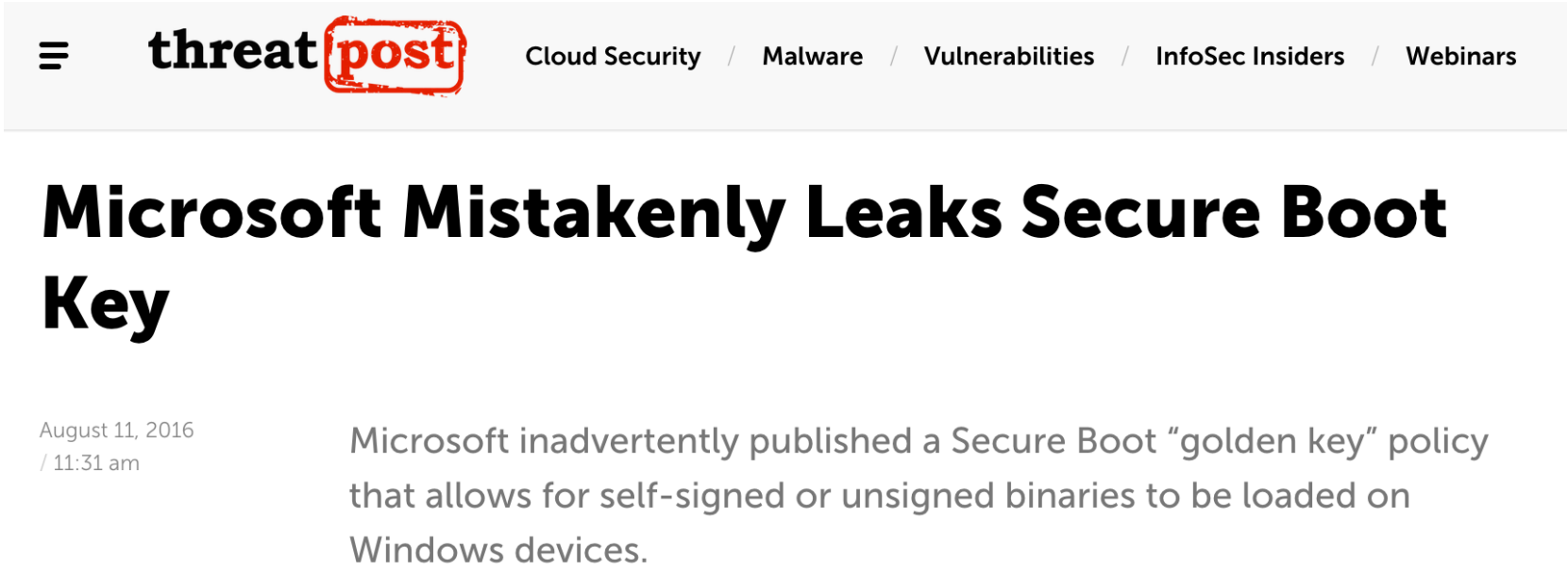
Secure Boot: How Secure is It?

- Boot chain security relies on
  - Root and subordinate CAs ensuring appropriate key management practices
  - Root and subordinate CAs signing only trustworthy binaries



Secure Boot: How Secure is It?

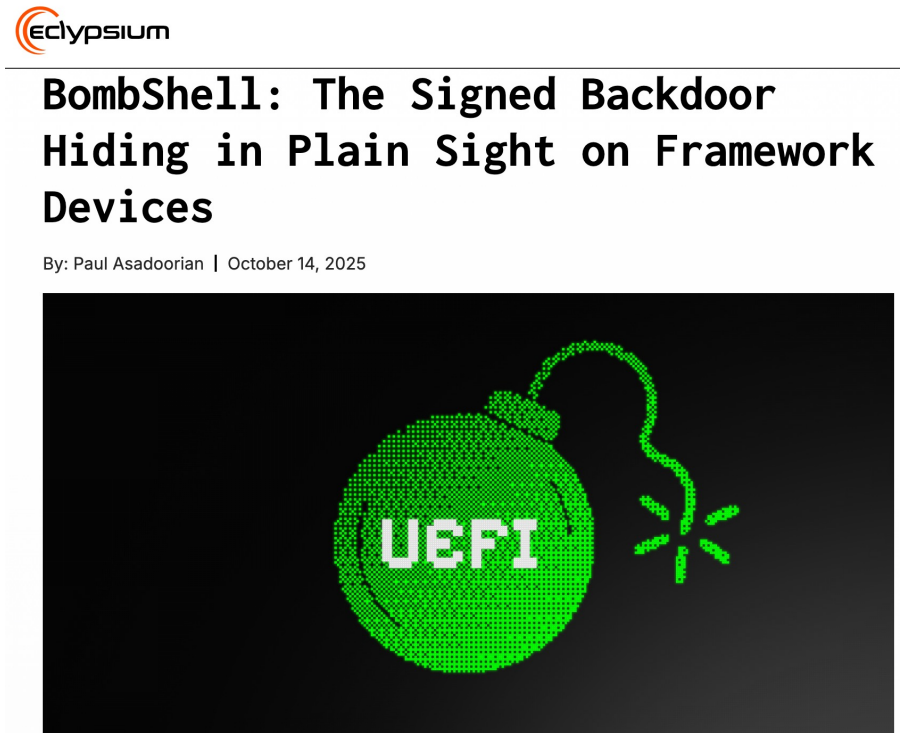
- Boot chain security relies on
  - Root and subordinate CAs ensuring appropriate key management practices
  - Root and subordinate CAs signing only trustworthy binaries
  - Signed binaries verifying any subsequently chainloaded binaries



Secure Boot: How Secure is It?

- Boot chain security relies on
  - Root and subordinate CAs ensuring appropriate key management practices
  - Root and subordinate CAs signing only trustworthy binaries
  - Signed binaries verifying any subsequently chainloaded binaries
  - Signed binaries not introducing vulnerabilities of their own

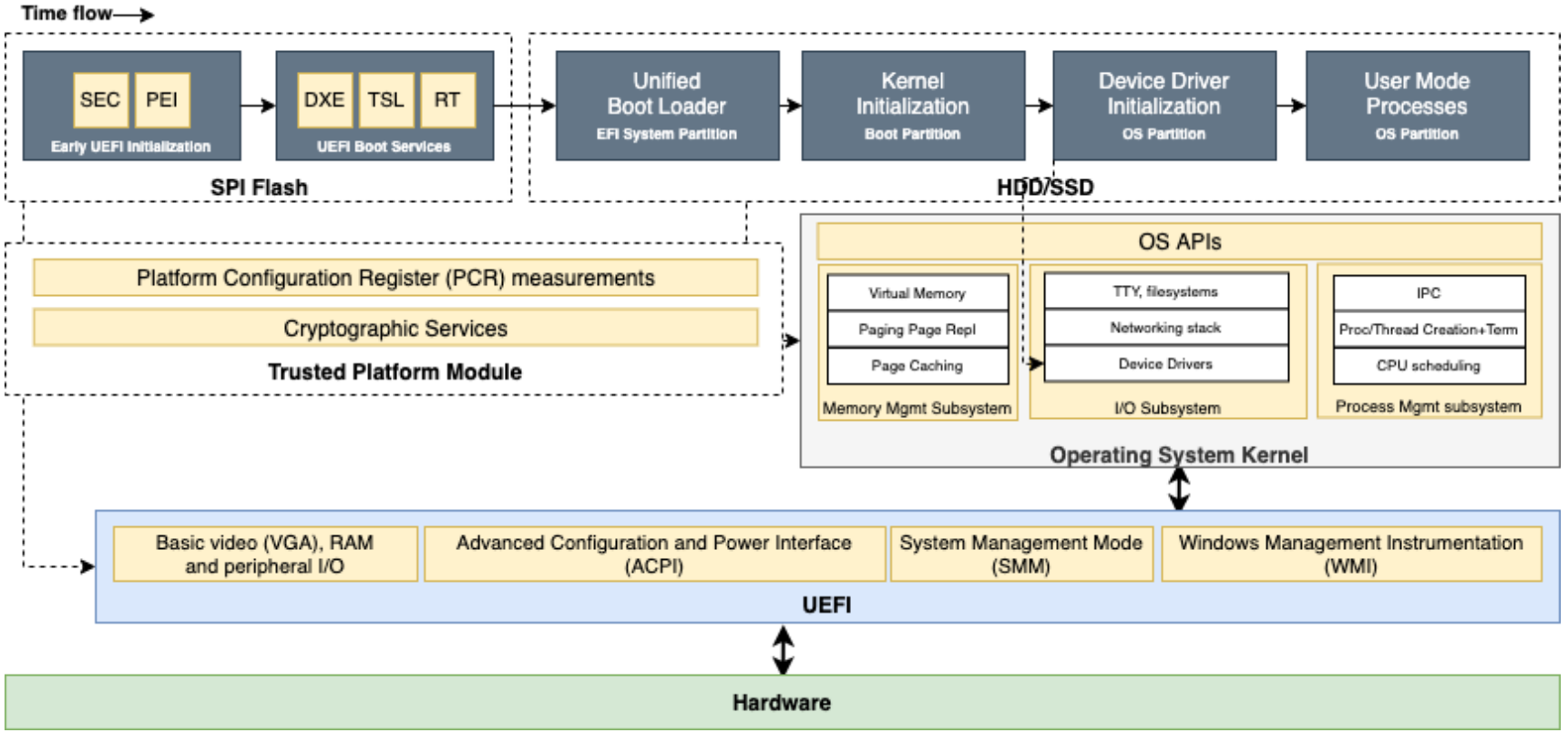
"BootHole" vulnerability in the GRUB2 bootloader opens up Windows and Linux devices using Secure Boot to attack. All operating systems using GRUB2 with Secure Boot must release new installers and bootloaders.



Secure Boot: How Secure is It?

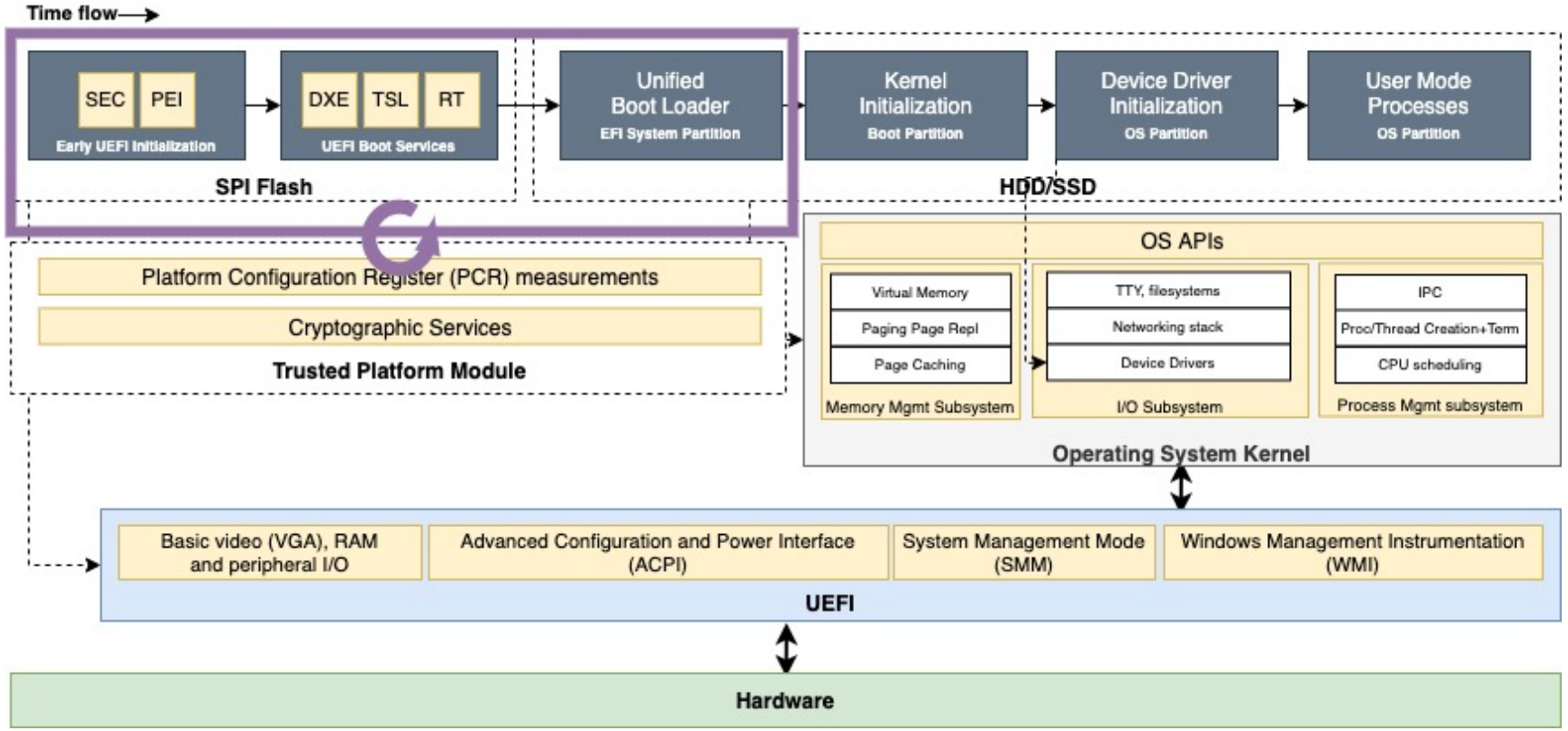
- Boot chain security relies on
  - Root and subordinate CAs ensuring appropriate key management practices
  - Root and subordinate CAs signing only trustworthy binaries
  - Signed binaries verifying any subsequently chainloaded binaries
  - Signed binaries not introducing vulnerabilities of their own
  - If all fails, CAs timely identifying affected public keys/binaries; OEMs distributing dbx updates

Measured Boot: Essentials (1)



- Requires autonomous, distinct agent that continuously monitors boot process

Measured Boot: Essentials (2)



- Typical x86 approach employs TPM
  - Learns about known good state, i.e. “one-time” setup that computes hash over UEFI image, UEFI config, bootloader and other boot-related parameters.
  - Stores state as PCR tuple. Continuously computes PCR on each boot cycle – e.g.

    $PCR[11]_n = SHA256(PCR[11]_{n-1} \,\#\, ValueToExtendWith)$
  - Common use case: PCR changes ⇔ boot chain modified

Measured Boot: Platform Configuration Registers (1)

0	Firmware	UEFI Boot Component	Core system firmware executable code	UEFI TPM event log
1	Firmware	UEFI Boot Component	Core system firmware data/host platform configuration; typically contains serial and model numbers	UEFI TPM event log
2	Firmware	UEFI Boot Component	Extended or pluggable executable code; includes option ROMs on pluggable hardware	UEFI TPM event log
3	Firmware	UEFI Boot Component	Extended or pluggable firmware data; includes information about pluggable hardware	UEFI TPM event log
4	Firmware	UEFI Boot Component	Boot loader and additional drivers; binaries and extensions loaded by the boot loader	UEFI TPM event log
5	Firmware	UEFI Boot Component	GPT/Partition table	UEFI TPM event log
7	Firmware	UEFI Boot Component	SecureBoot state	UEFI TPM event log

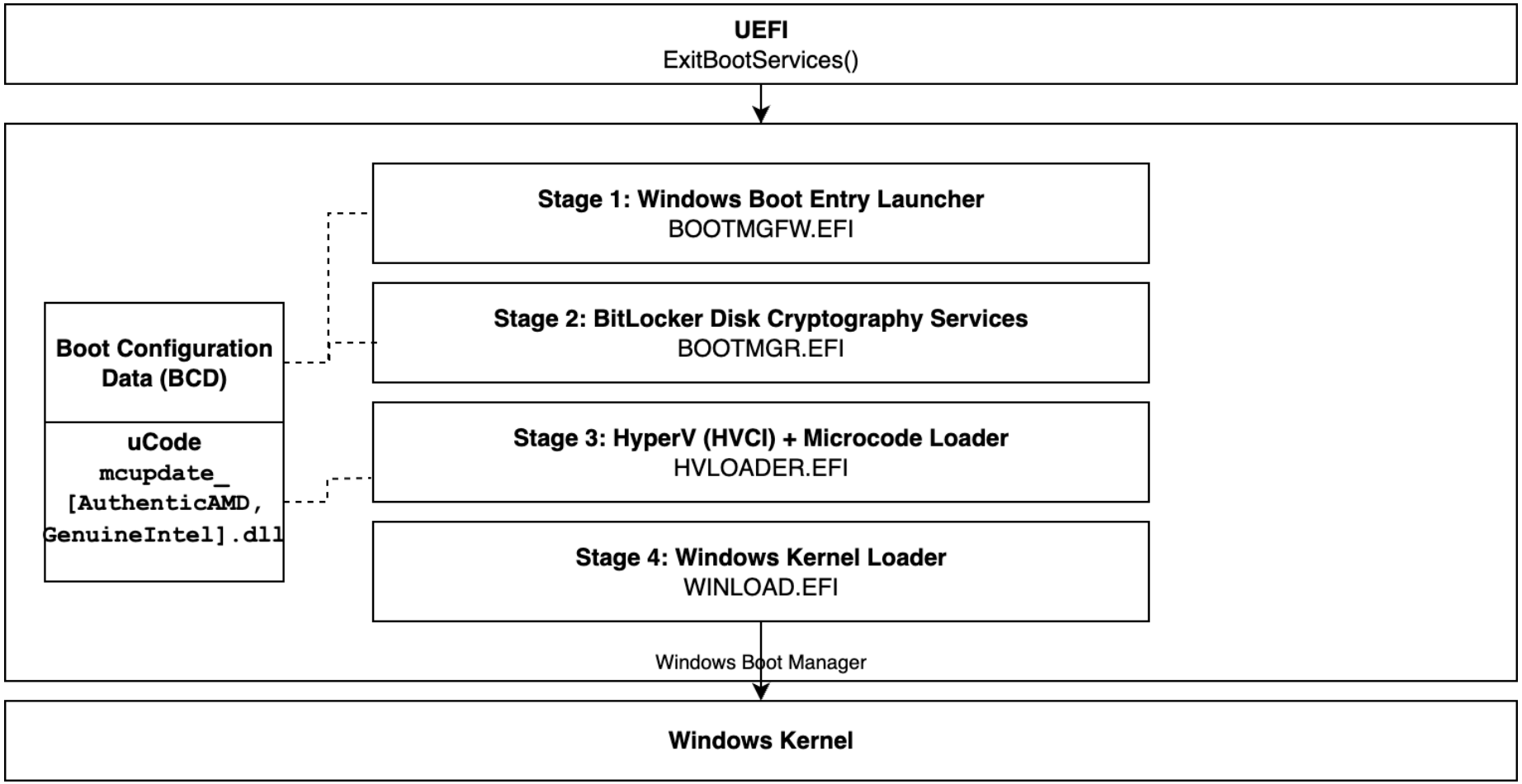
["PC Client Specific Platform Firmware Profile Specification 2.0", Trusted Computing Group]

Measured Boot: Platform Configuration Registers (2)

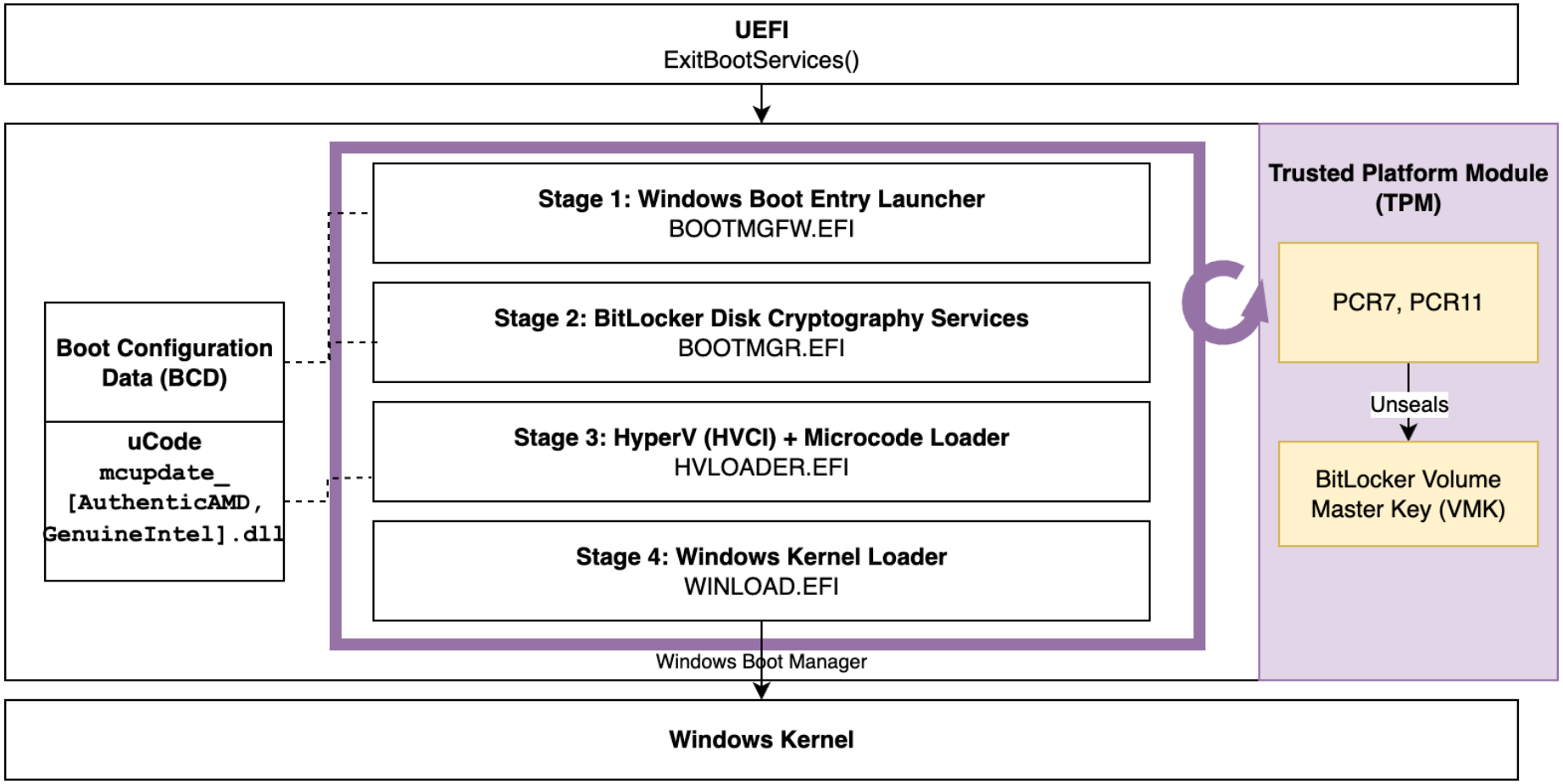
8	`grub` 🍲	UEFI Boot Component	Commands and kernel command line	UEFI TPM event log
9	`grub` 🍲	UEFI Boot Component	All files read (including kernel image)	UEFI TPM event log
	Linux kernel 🫘	Kernel	All passed initrds (when the new `LOAD_FILE2` initrd protocol is used)	UEFI TPM event log
10	IMA 📐	Kernel	Protection of the IMA measurement log	IMA event log
11	`systemd-stub` 🚀	UEFI Stub	All components of unified kernel images (UKIs)	UEFI TPM event log
	`systemd-pcrphase` 🚀	Userspace	Boot phase strings, indicating various milestones of the boot process	Journal (for now)
12	`systemd-stub` 🚀	UEFI Stub	Kernel command line, system credentials and system configuration images	UEFI TPM event log
13	`systemd-stub` 🚀	UEFI Stub	All system extension images for the initrd	UEFI TPM event log
14	`shim` 🔑	UEFI Boot Component	“MOK” certificates and hashes	UEFI TPM event log

[“PC Client Specific Platform Firmware Profile Specification 2.0”, Trusted Computing Group]

Measured Boot: Windows Boot Chain (3)

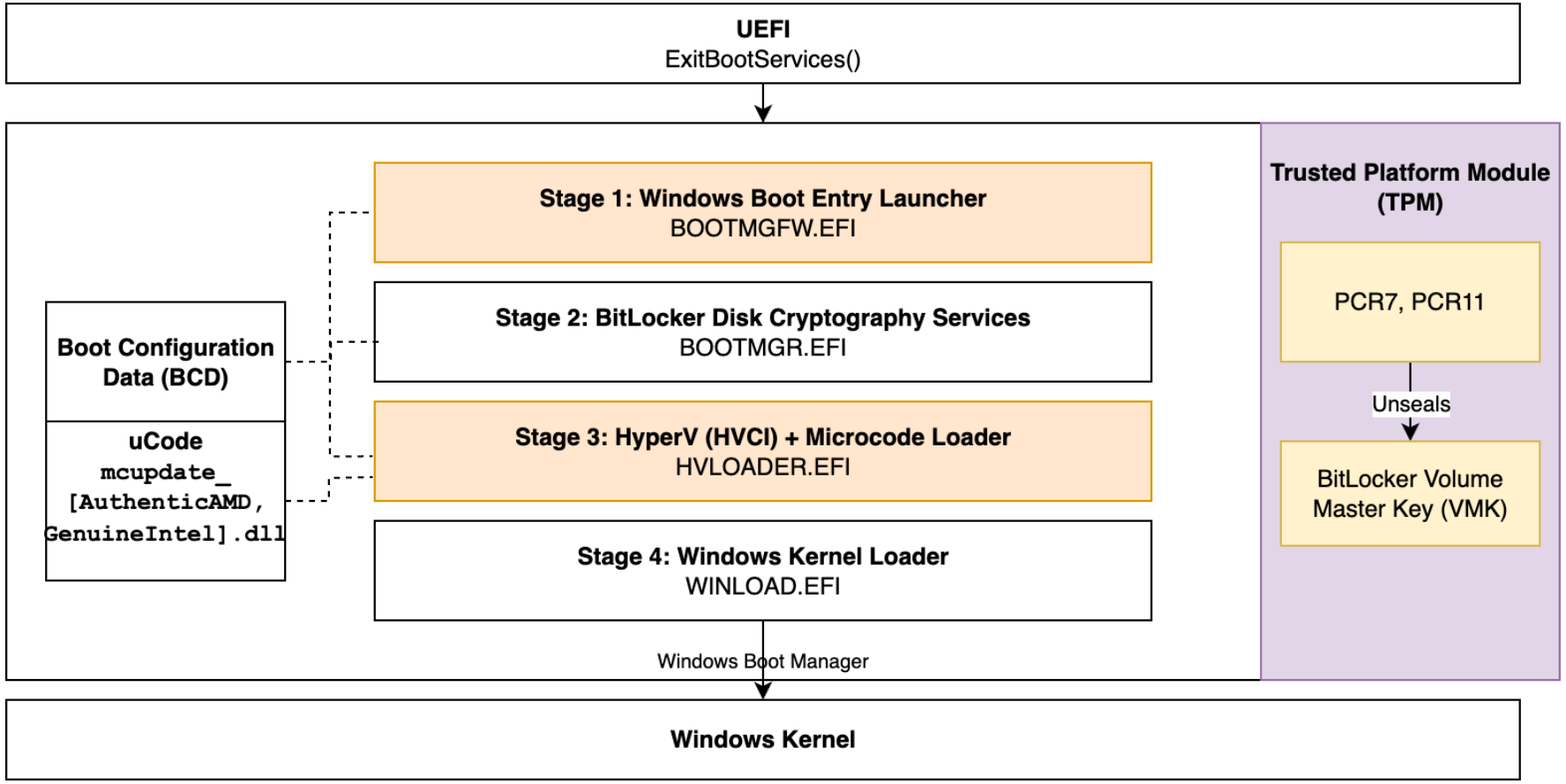


Measured Boot: Windows Boot Chain (4)



Measured Boot: How Secure Is It?

Measured Boot: How Secure Is It?



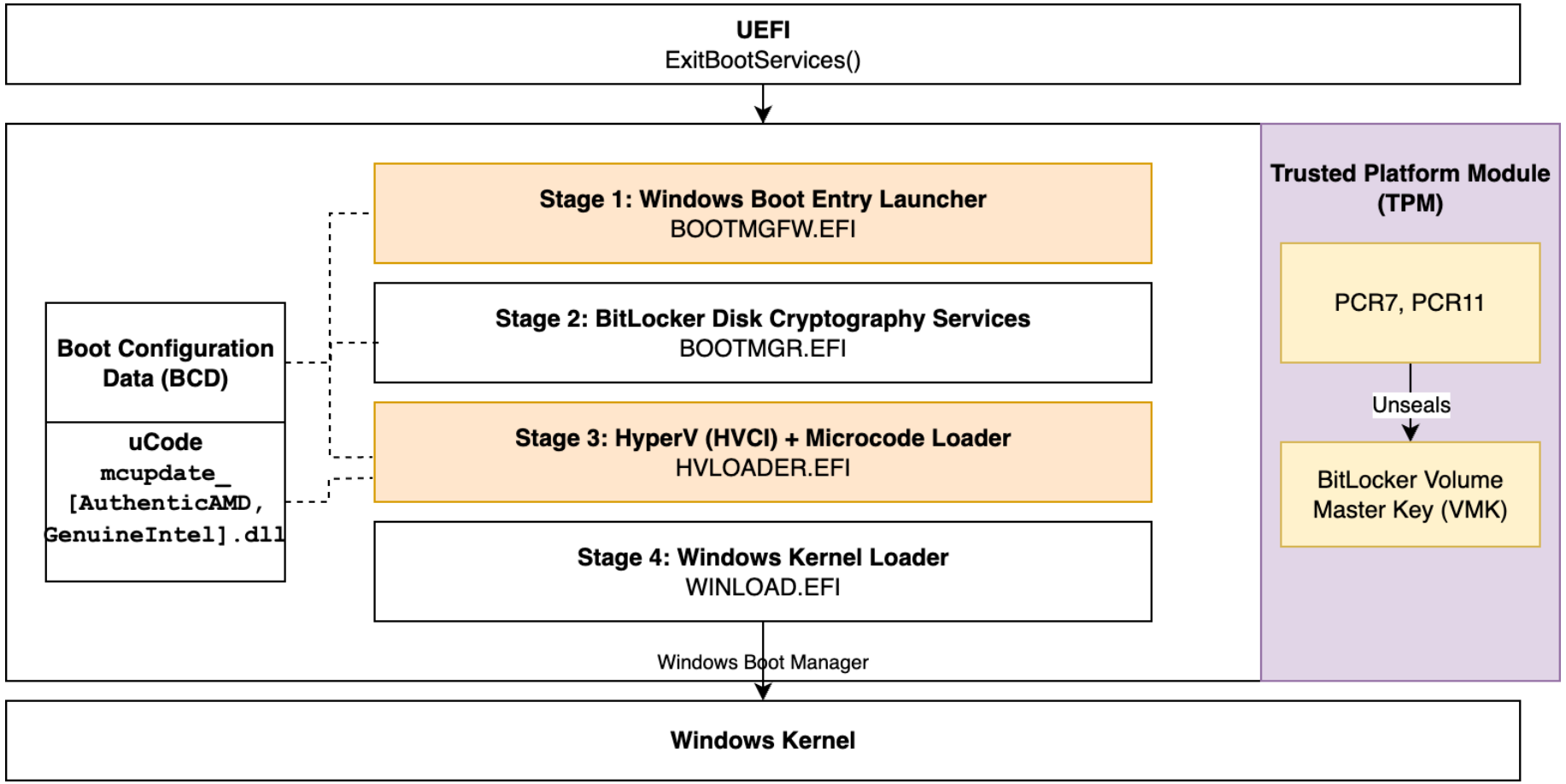
- Black Lotus (2023)
  - APT-sponsored bootkit campaign; exploits Windows bootloader 0-day (“Baton Drop”)
  - Disables BitLocker measured boot through OS runtime privilege escalation
  - More details in bonus slides

Conclusion

- From BIOS to UEFI: transition from legacy, inflexible, insecure BIOS/MBR booting to modern UEFI with richer functionality and scalability
- Modular architecture, larger firmware space, faster parallelized initialization, GPT support, and robust extensibility
- Strong, but not flawless boot-time security mitigations
  - Secure Boot as strong as weakest link in 1) PKI-based trust hierarchy, and 2) chainloaded binaries
  - Measured Boot conceptually strong, implementation-wise a work in progress
  - Verified Boot remains unchallenged… or does it?

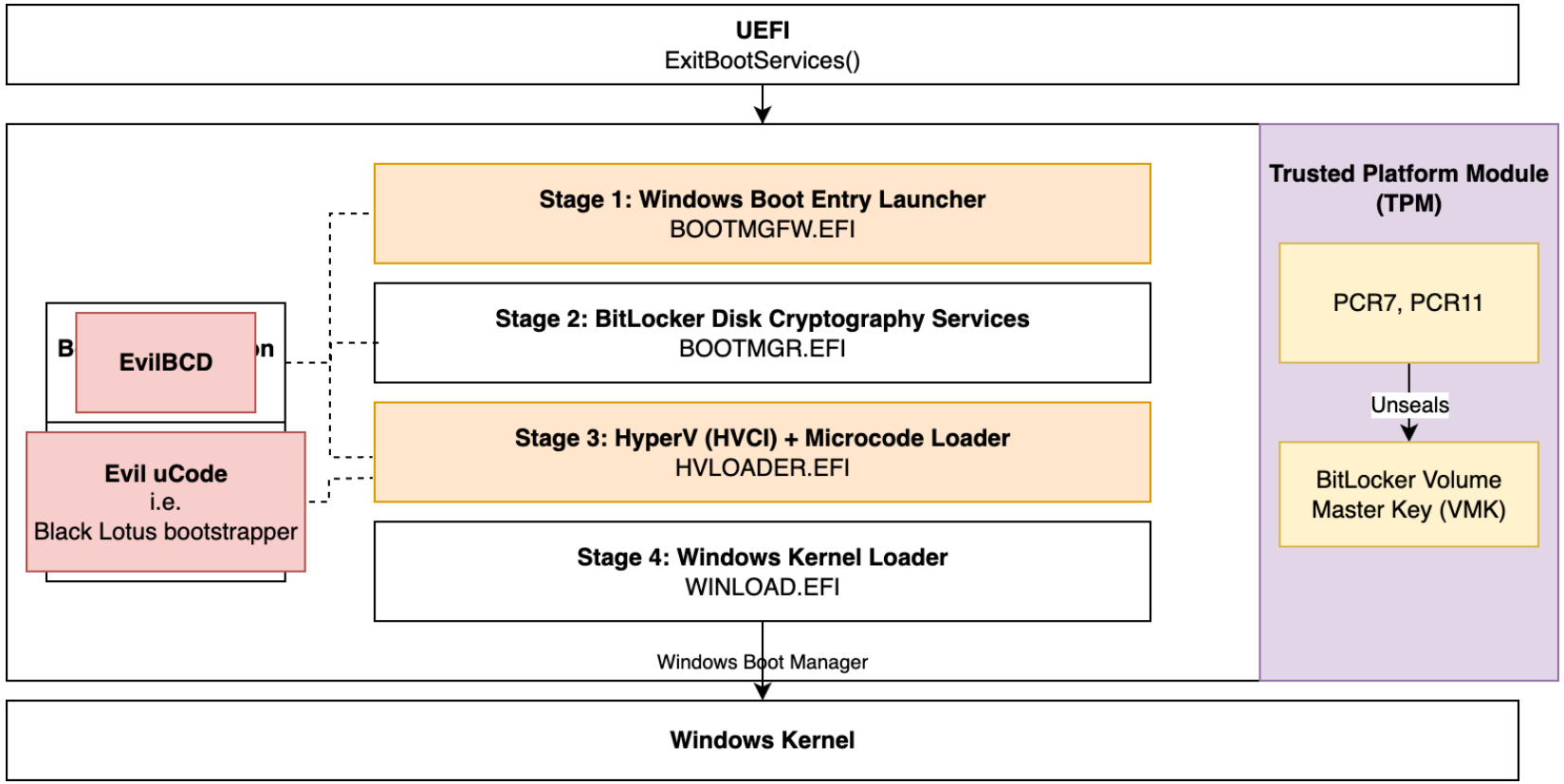
Bonus slides – Black Lotus (2023)

Black Lotus (1): Entry Point



- Baton Drop (CVE-2022-21894)
  - Bootmgfw BCD parser vulnerability: `truncatememory` allows limiting accessible PA range
  - Microsoft issues WBM update, but does not revoke trust through DBX

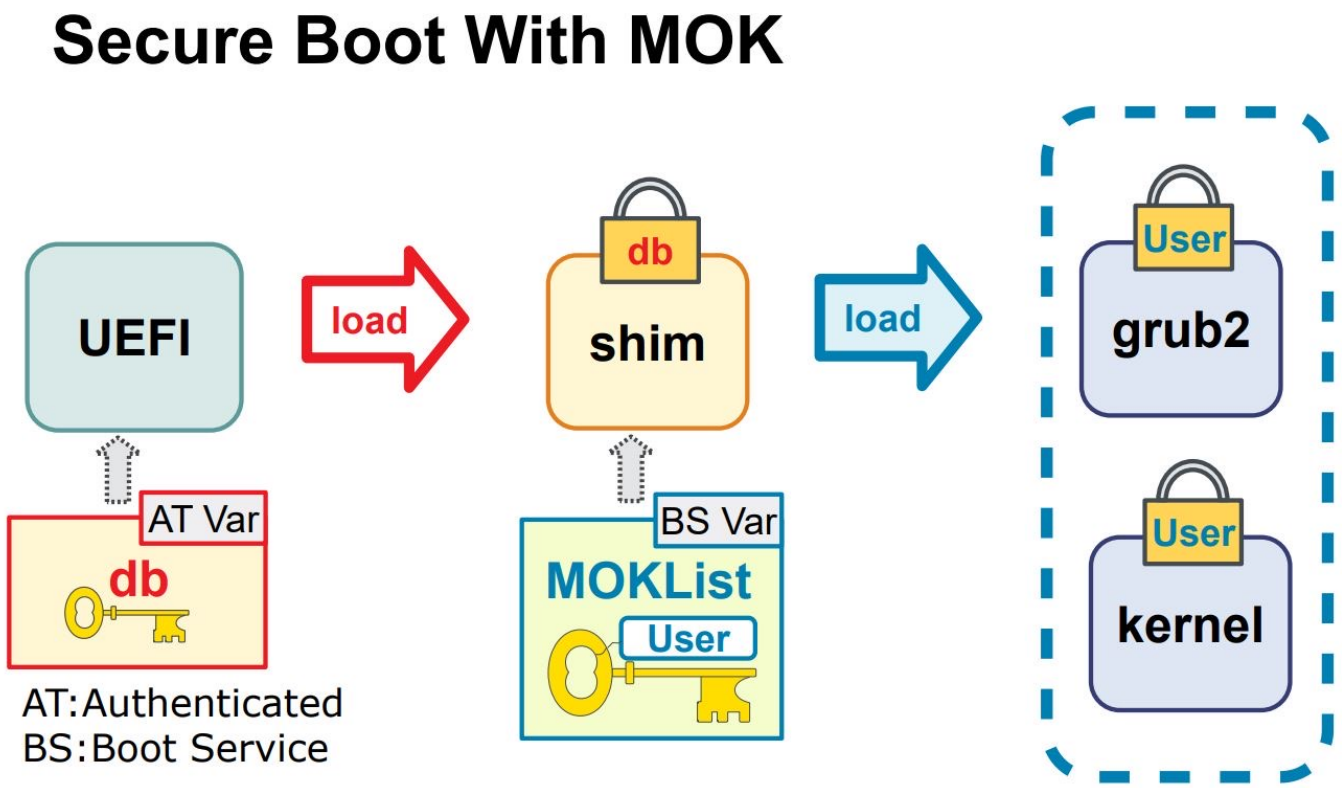
Black Lotus (2): Attack Flow



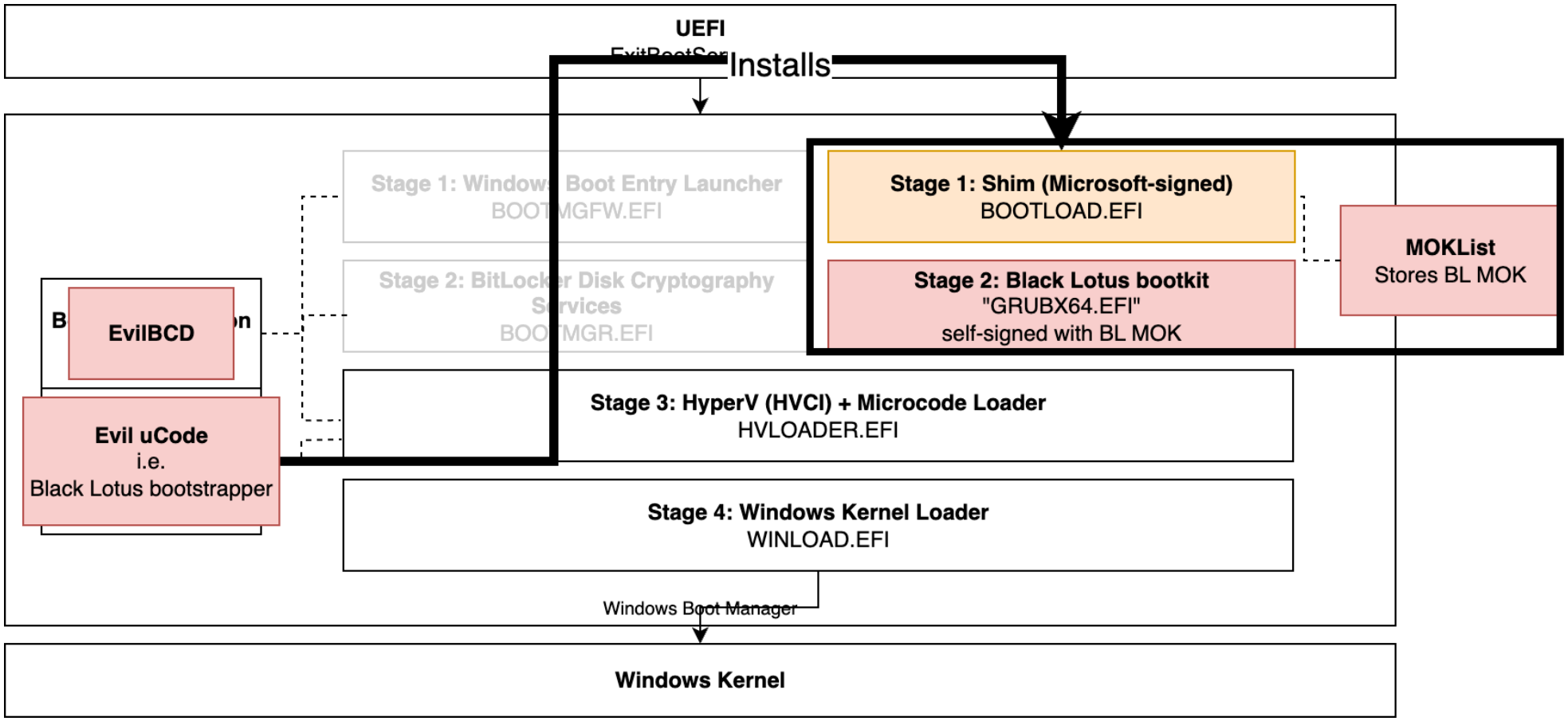
- **Black Lotus Dropper**: at Windows runtime (e.g. browser drive-by-download w/privesc, supply chain attack, social eng), installs **EvilBCD** and **Evil uCode**
- WBM Secure Boot policy stored at 0x10000000
- **EvilBCD** sets `truncatememory` = 0x10000000, effectively disabling Secure Boot, and thus enabling `testsigning`, `nointegritychecks`

Pit stop: Shim

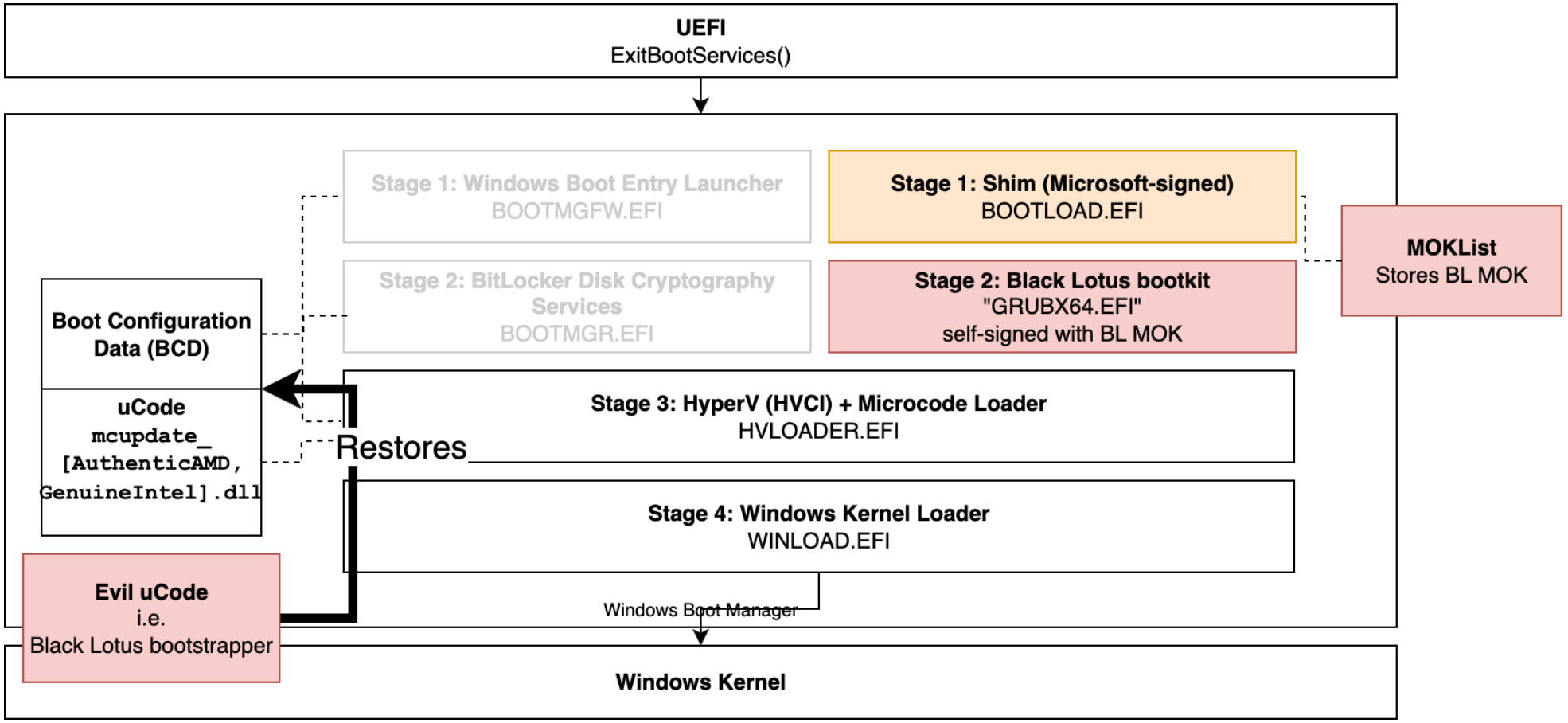
- Red Hat: SHIM first stage boot loader
  - Used to chainload e.g. GRUB2, systemd-boot, EFI Stub
  - Enables users to self-sign boot loaders + drivers using “Machine Owner Key” (MOK)



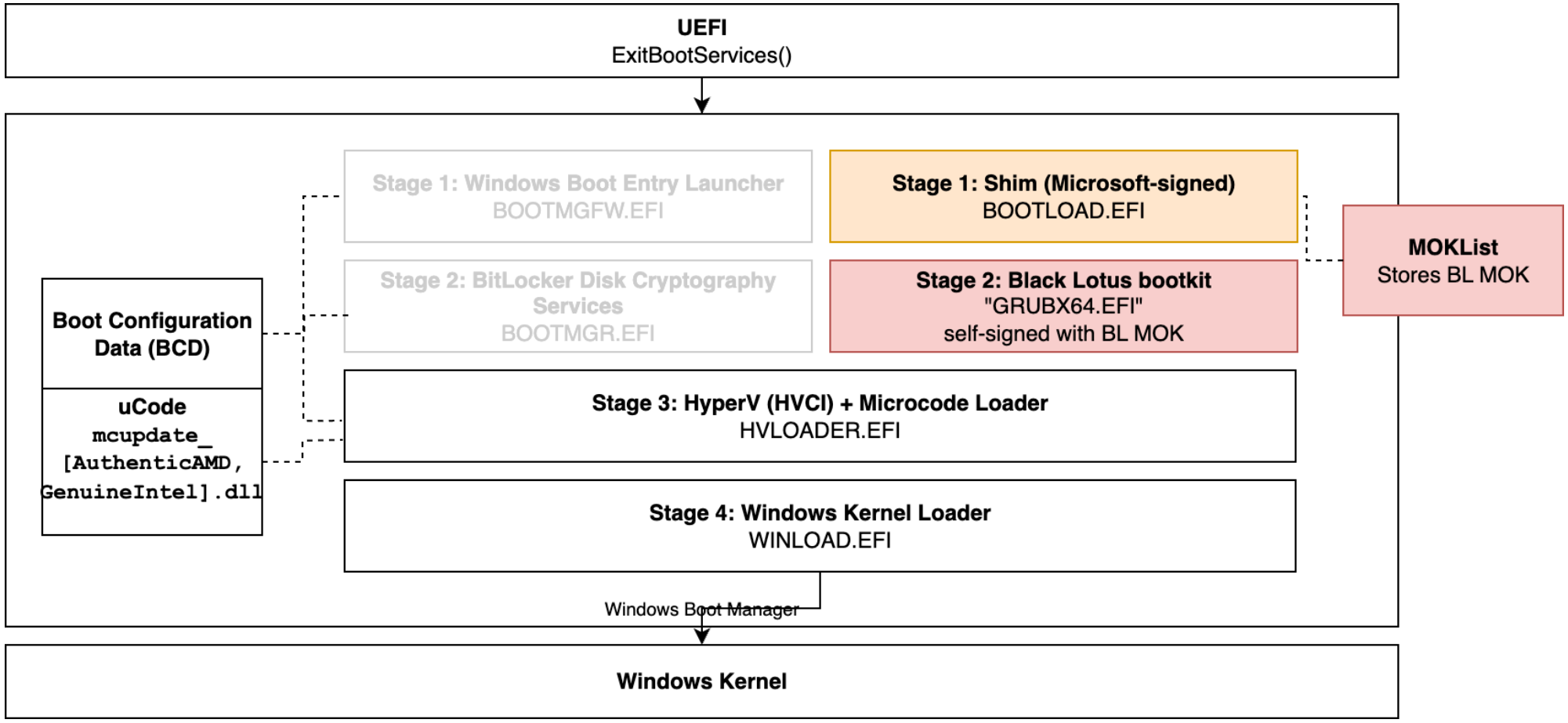
Black Lotus (3): Persistence



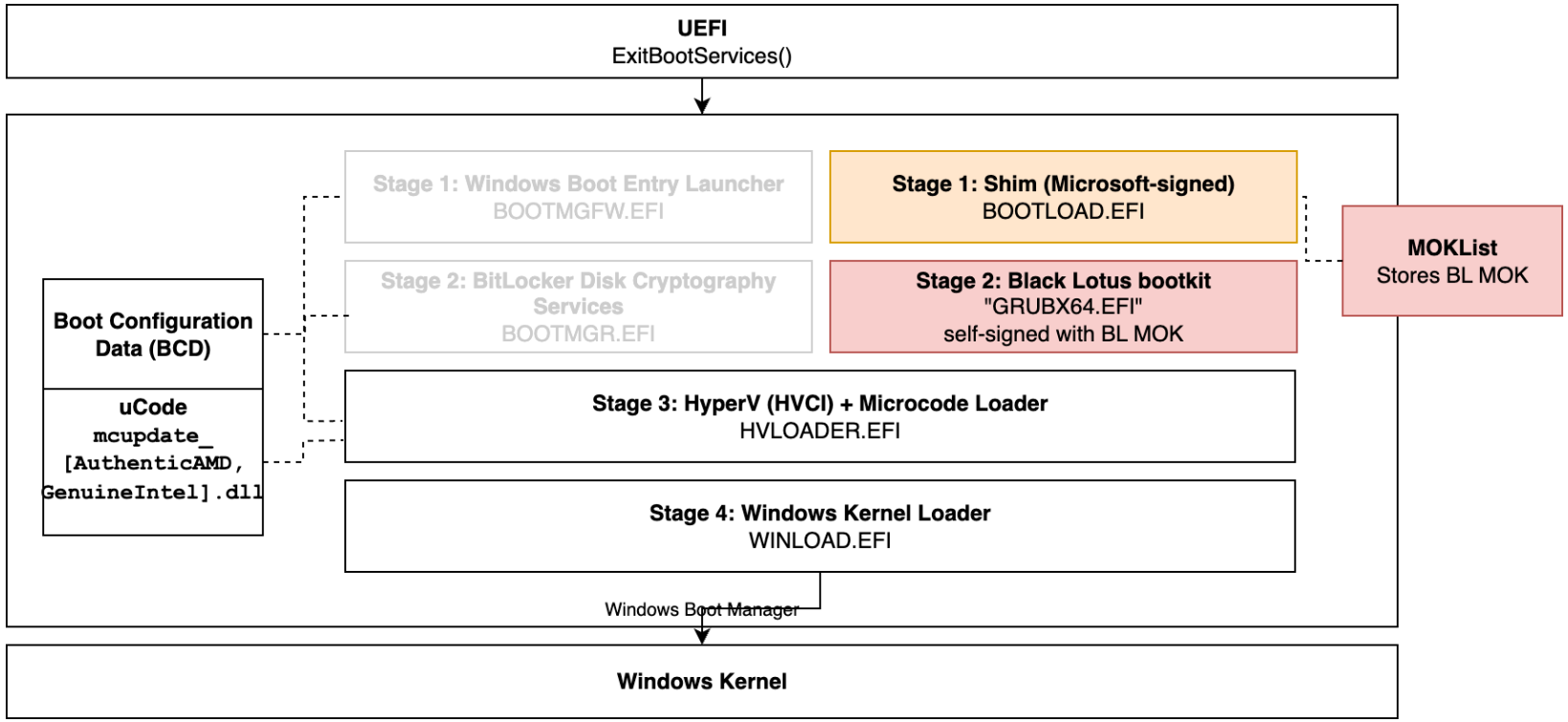
Black Lotus (4): Clean Up



Black Lotus (5): Compromised Boot Chain

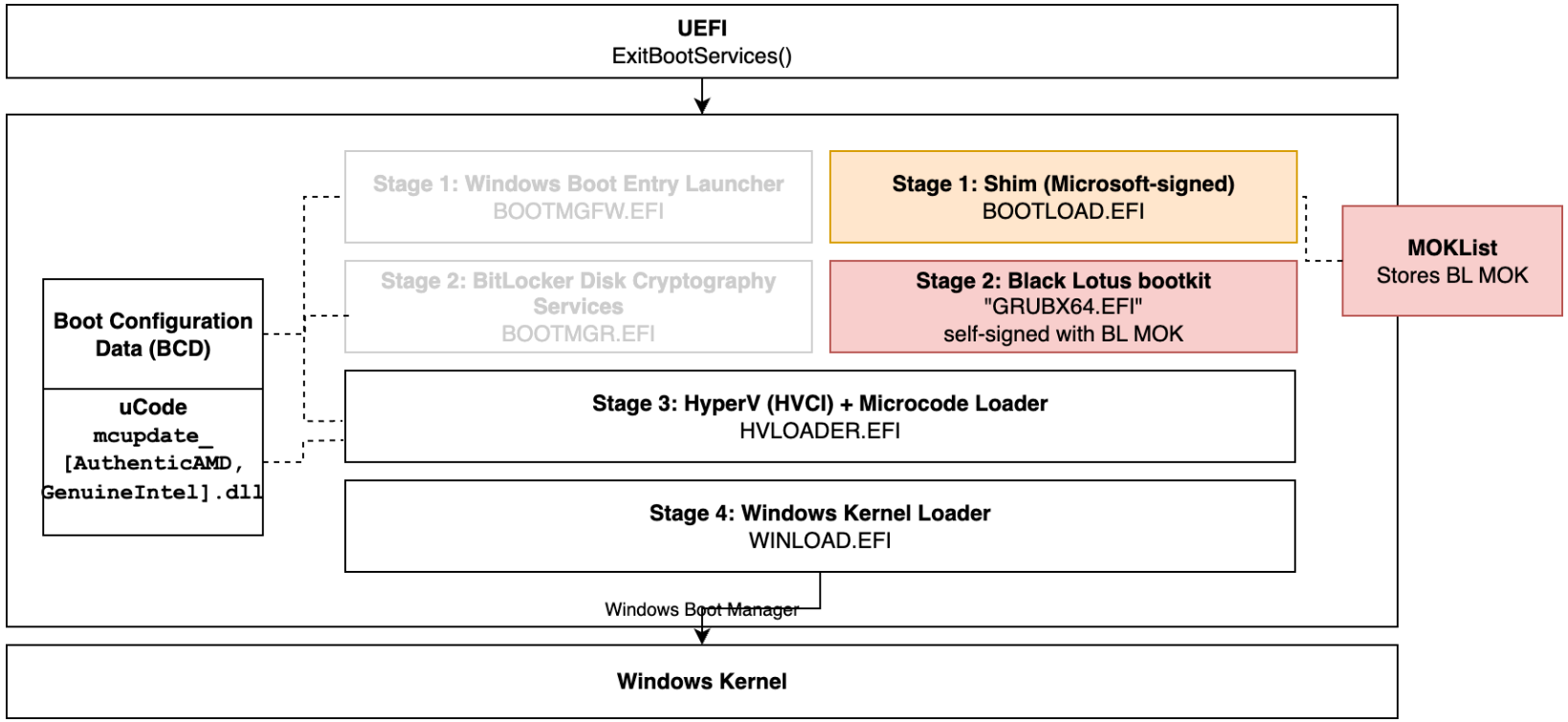


Black Lotus (6): Compromised Boot Chain



- But… what about TPM's PCR measurements?
  - As part of first stage, Black Lotus dropper simply disables BitLocker measured boot through `DisableKeyProtectors`

Black Lotus (7): Compromised Boot Chain



- But… what about TPM's PCR measurements?
  - As part of first stage, Black Lotus dropper simply disables BitLocker measured boot through `DisableKeyProtectors`
- Windows kernel-based C&C agent allows further system compromise