Storage-Enabled Secure Boot

Kevin Butler, Stephen McLaughlin, Thomas Moyer,
Joshua Schiffman, Patrick McDaniel, and Trent Jaeger

Commodity Secure Boot

In order to ensure data and system integrity, assurances of a system's state are needed. This can be provided through *secure boot*, a mechanism whereby the system is booted in stages and the boot is only allowed to continue if each stage is valid.

However, secure boot systems are not common in everyday systems for the following reasons:

- Many proposals require special hardware (e.g., AEGIS, IBM 4758) that can be expensive
- Encryption-based solutions require substantial key management and those keys are released after the boot

Storage is well-positioned to provide a solution for secure boot because it can ensure that information is not released until the boot is verified.

Goals for a Solution

We consider *autonomously secure disks* that can independently enforce policy. These provide a platform for implementing the desired mechanisms.

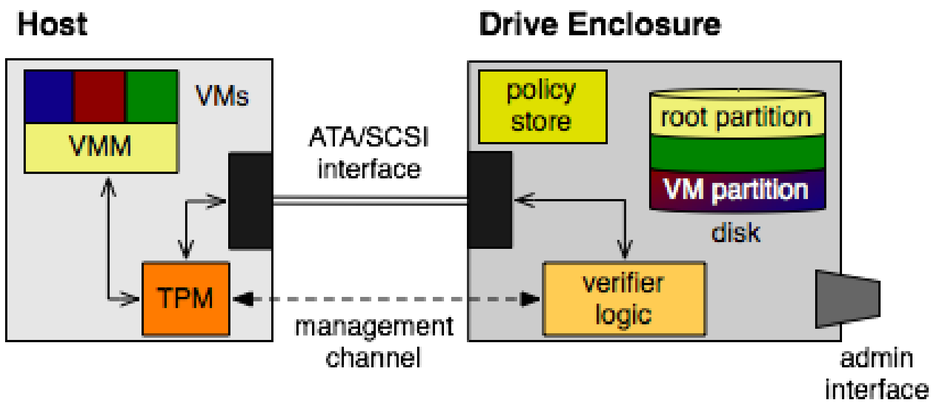
Our goals for the solution are as follow:

- **Secure boot from storage:** a staged boot that assesses the host's system state at set points within the boot process
- **Continuous enforcement of system state:** create a framework for runtime monitoring after the system has booted
- **Usability and ease of management:** provide management functionality through the disk interface or a physical token

Firma

We design *Firma*, a system protecting on-disk storage by *measuring* the host system state from boot time until the OS kernel is loaded.

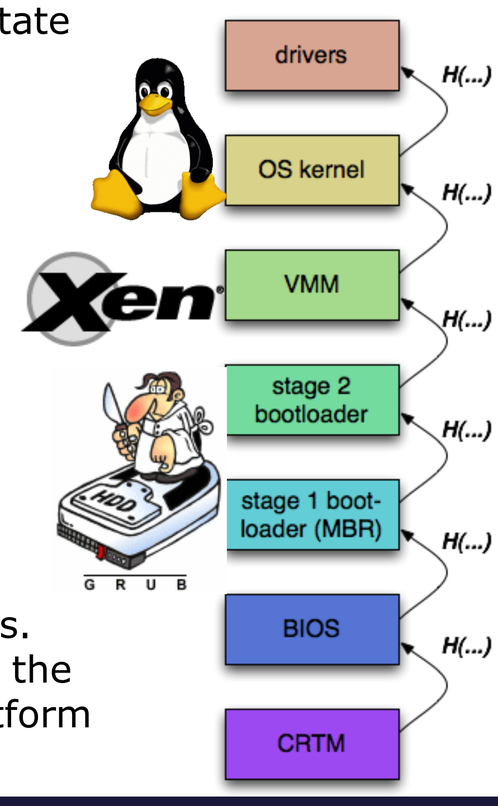
- While a single-user system can be supported by this architecture, we concentrate on a system running virtual machines.



Measuring System Integrity

We use *integrity measurement* to determine the state of the host system. A *measurement* is a hashed fingerprint of the code at a particular stage.

- Hardware support emanates from the *core root of trust for measurement (CRTM)*, which is secured on the host.
- Subsequent steps are measured from the BIOS through the VMM up to the OS and applications.
- Attestation of the code on the disk is performed using a *trusted platform module (TPM)* on the host, which is a tamper-resistant, secure microcontroller found in most modern x86 systems. The *Quote* operation on the TPM is performed and the results compared with values inside the TPM's platform configuration registers (PCRs).



Attestation Protocol

We make use of the TPM's *attestation identity key* (AIK), a unique identifier, and store it on the disk since it is stored in volatile memory on the TPM. A *storage root key* (SRK) is generated by the TPM, with a public portion distributed and a private portion that stays on the TPM.

Pairing

(1) H : generate AIK $= (AIK^+, AIK^-)$
(2) $H \rightarrow D : AIK^+, \{AIK^-\}_{SRK^-}$

Boot

(3) $D \rightarrow H : \{AIK^-\}_{SRK^-}$
(4) $D : n =$ Generate nonce
(5) $D \rightarrow H : Challenge(n)$
(6) $H \rightarrow D : Attestation = Quote + ML$
(7) $D : Validate(Quote, ML)_{AIK^+}$

The disk can receive the measurements in an out-of-band measure or by directly measuring a new system in *measurement mode*, where each stage of the boot is measured and data is recorded to the disk or a token plugged into the host.

Subsequent validation of system state can use approaches such as LKIM, the Linux integrity monitor.

Installation of the system requires a *root of trust installer* (ROTI), while the host system's CRTM provides a self-measurement of the initial state. The disk is authorized by an administrator through a token that the root partition of the disk is writeable, and the VMM and supporting OS are written to disk. The measurement list of installed files is sealed with the host's SRK.

Performance

Operation	Time
Host System	
Measure boot binaries	24.9388 (24.9113, 24.9662)
TPM Extend	39.9934 (39.9887, 39.998)
TPM Quote	880.037 (880.03, 880.045)
Disk Firmware	
Verify quote	70.0msec.

Our prototype shows that less than a second is added to the boot time and in our tests of supporting periodic runtime attestations, and the impact on throughput is negligible.

A full measurement of a virtual machine monitor the size of the VMWare ESX hypervisor takes approximately 9 seconds; we perform this is an error has occurred. These costs can be brought down through *lazy attestation*, hiding the computational costs in seek times for the disk.

Supported by NSF