UEFI Secure Boot

Where we stand

Parallels™

Profit from the Cloud

James Bottomley

CTO, Server Virtualization; SCSI Subsystem, Parisc Kernel Maintainer

31 January 2013

About Me

- FAQ
- Or more properly, (FGA) Frequently Given Answers
  - I'm kernel maintainer of SCSI and PA-RISC
    - So I'm into crazy and obsolete systems
  - My day job is as CTO of Server Virtualisation for Parallels
  - I only got into Secure Boot because everyone moved faster than I did when the crap was landing
  - I began wearing Bow Ties way before Doctor Who made it cool

Introduction

- UEFI Secure boot is a static way of assigning trust to the boot system
- It is mandated by Microsoft to be enabled in all shipping Windows 8 systems
- The Microsoft Mandate requires all keys to be owned either by the OEM or by Microsoft
- Secure Boot must be capable of being Disabled and the keys replaced
- But no standard mechanism for doing this exists

The Secure Boot Keys

- There are three sets of keys
  - The Platform Key (PK) , designed to be owned by the owner of the hardware
    - Microsoft mandates that this belong to the OEM
  - The Key Exchange Keys (KEK) designed to be owned by trusted entities for boot
    - Microsoft mandates they own at least one of these
  - The Signature Database (db) designed to verify trusted binaries
    - Microsoft mandates they have a key here too.
    - db signatures are required to boot in a trusted environment

How it Works

- PK may only be used to update KEK
  - So the PK owner decides
    - what keys to trust in the key
    - When to be in Setup Mode
- KEK may only be used to update db
  - So all owners of KEKs can update or revoke db keys
- db keys must be used to sign binaries which are trusted by the system.

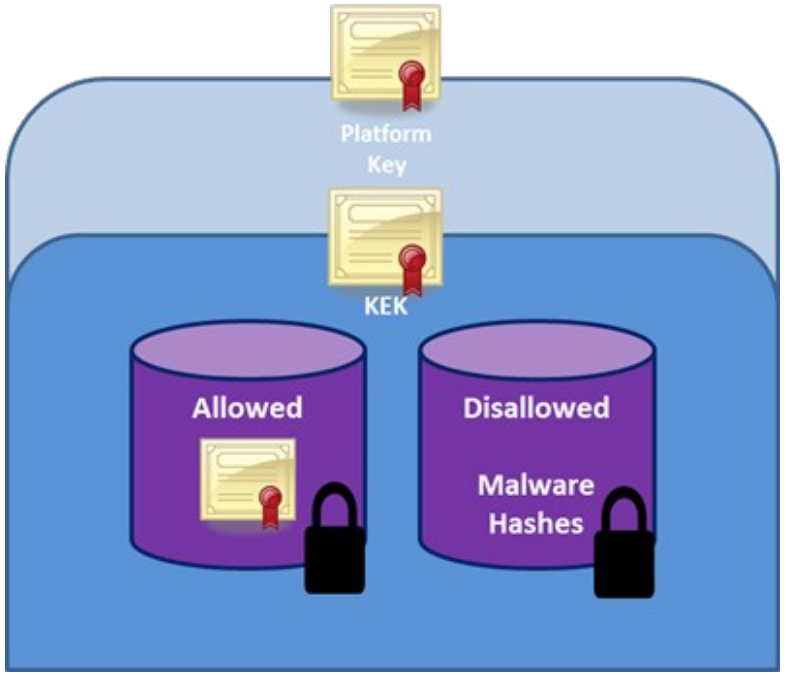

Diagram from Microsoft

How Microsoft Mandates that it Work

- The Windows 8 Logo Requirements are
  - OEM controls Owner Key
  - Microsoft owns keys in KEK and db
    - Several keys, in fact: it looks like Windows boot will be signed by a separate root of trust from the third party signing system
  - On non-ARM systems, secure boot must be disabled via a UEFI menu
    - No mandate for where this is or how easy it is to do.
  - On non-ARM systems, the user must be able to replace all the keys
      - Again, no requirement for key administration
      - OEM can comply by simply having the system remove all the keys

GPLv3 and Secure Boot

- People think GPLv3 requires disclosure of signing keys in a lock down environment
- The Linux Foundation saw this problem in the early drafts of the Microsoft Windows 8 Logo docs and sought to fix it
- However requirement is only that the user be able to boot their own system
- Ejecting the preset keys and installing your own, with which you can then sign your system is sufficient
- Implies reset to setup mode in UEFI interface, as Mandated by Microsoft, satisfies GPLv3 obligation
- FSF Supports this interpretation

The Threat

- Since Microsoft owns all the Signing keys, no Linux boot system will work out of the box without their approval
- Approval requires not booting malware and obeying the Windows 8 logo mandates.
  - Implies simply getting Microsoft to sign a Linux bootloader isn't an option
- Linux won't boot on Windows 8 systems (i.e. all current PC systems) without a Microsoft approved method of booting
  - Trying to explain to users how to disable secure boot isn't an option
  - Because of the non-standard mechanisms for doing so.

The Opportunity

- Secure boot gives users a way of protecting their systems from external intrusion
- Supporting it end to end would facilitate Linux playing in secure environments
- To be effective, must carry the root of trust through the secure boot to the Operating System environment
  - May require other trust implementations like signed modules
  - Or disallowing root access to PCI configuration space

The Linux Response

- Two Challenges
  1. Keep the Ecosystem booting easily in the face of secure boot
  2. Enhance Security policy for distributions by taking advantage of secure boot.
- The Linux Foundation response has concentrated exclusively on 1.
- The Linux Distributions are Investigating and preparing for 2.
- Red Hat and Canonical already shipping Secure Boot in some form

Original LF Plans

- Develop a set of tools to enable owner to easily take control of the system and manage the keys
  - Allows ejecting of OEM and Microsoft keys and installing your own
- Tools also permit the creation of signed binaries to reset the platform to setup mode
  - Just in case something goes wrong with UEFI interface
- A signed pre-bootloader that will boot any unsigned bootloader with a present user test
  - And will install bootloader signature in setup mode to avoid the present user test

What the Distributions are Doing

- Red Hat (Matthew Garrett) interacted with UEFI forum and OEMs to create the Shim bootloader
  - Boots a signed second stage loader, which boots a signed kernel
  - Kernel is locked down by module signing and other measures
- SUSE has Machine Owner Key (MOK) approach
  - Shim modified to accept key updates from present user
  - Means user can resign the boot loader and install their own key
- Both approaches require signing shim with the microsoft key

SHIM + MOK Solutions

- MOK means Machine Owner Key
  - Stored in a new MokList variable which is NV+BS
- Matthew Garrett adding ability to store hashes in MOK database
  - Means shim + MOK can now chain unsigned bootloaders
  - Unsigned bootloaders can be authorised on the fly using the MOK solution
- Shim runs an EFI binary if the key or hash is
  - in db (allowed signatures) or MokList
  - And not in dbx (forbidden signatures)
- SHIM solutions only work with legacy link loaders, like Grub and efiboot

Architectural Problems

- All current secure boot workarounds overcome the UEFI signature check by linking and running UEFI binaries themselves
  - Means they are essentially link loaders themselves
- Unfortunately, this means they **won't** work with the new generation of Bootloaders like gummiboot.
  - Rely on kernel EFI stubs and use BS->LoadImage()
- Discovery of this problem lead to a complete re-architecture of the LF secure boot solution in November/December
- Plus UEFI redefined authorisation returns.
  - EFI_ACCESS_DENIED and EFI_SECURITY_VIOLATION

A New Architecture for Secure Boot

- Instead of building a Link Loader, Build a plug in for the UEFI Security Architecture
  - Documented in the Platform Initialization Specification
    - DXE Security Architectural Protocols
    - EFI_SECURITY_ARCH_PROTOCOL (PI1.1)
    - EFI_SECURITY2_ARCH_PROTOCOL (PI 1.2)
- A resident program can replace these protocols and do its own authorisation of binaries for the platform
- Redesign Pre-Loader to do this
  - However, cannot do present user tests from this hook, must have MOK like system instead

A look at the New Architecture

- Pre-Loader now installs a simple hook which
  - Intercepts both security arch protocols (if they exist)
  - Chains the call to the previous protocol
  - If that succeeds, return EFI_SUCCESS
  - If that fails, look up the binary hash in the MokList variable
  - Authorise only if hash is present in MokList and not dbx
- Because Pre-Loader is resident, the intercepted security architecture remains in-place for all future users of BS->LoadImage
  - i.e. GummiBoot now works (with a couple of patches)

Ancillary Programs

- Pre-Loader is no longer interactive
  - At least not when it is the resident authentication system
- Need a HashTool to enrol the hash of binaries to be booted
- Also need a KeyTool to help display, save and manipulate the contents of the Authenticated variables
- Pre-Loader now starts HashTool to enrol the hash of loader.efi into the MOK database (MokList)

Security Auditing the LF system

- Because it relies on Hashes not X509 keys, there are no hidden secrets
  - Means all elements of the system can be externally verified
- Signed system is built using the openSUSE build service
  - This makes all elements of the build completely verifiable
  - Even after Microsoft has signed them.
- Only pre-authorised (by hash) binaries are HashTool and KeyTool from the same build.

Interactions with Microsoft

- KeyTool discovered flaws in shipping UEFI implementations
  - Could be used to delete the Platform Key on several systems without knowing the currently installed key
  - MS insisted on rewrites to make this impossible
- Other flaws were discovered (traversing multiple keys with the same signature header was wrong)
- Authorisation takes about a week or two.
- Finally fixed bootloader was submitted on 21 Jan
  - So hopefully a signed one should be with us any day now

Demo

- Resources:
  - https://build.opensuse.org/project/show?project=home%3Ajejb1%3AUEFI
  - http://git.kernel.org/?p=linux/kernel/git/jejb/efitools.git;a=summary
- Includes tianocore qemu image for UEFI plus tools for taking control of system and building keys and signature lists.

Questions?

jbottomley@parallels.com
James.Bottomley@HansenPartnership.com