

Defeating Invisible Enemies: Firmware Based Security in OpenPOWER Systems

Linux Security Summit 2017

George Wilson
IBM Linux Technology Center

Agenda

Introduction

The Case for Firmware Security

What OpenPOWER Is

Trusted Computing in OpenPOWER

Secure Boot in OpenPOWER

Current Status of Work

Benefits of Open Source Software

Conclusion

Introduction

Disclaimer

These slides represent my views, not necessarily IBM's

All design points disclosed herein are subject to finalization and upstream acceptance

The features described may not ultimately exist or take the described form in a product

Background

The PowerPC CPU has been around since 1990

Introduced in the RS/6000 line

Usage presently spans embedded to server

IBM PowerPC servers traditionally shipped with the PowerVM hypervisor and ran AIX and, later, Linux in LPARs

In 2013, IBM decided to open up the server architecture: OpenPOWER

OpenPOWER runs open source firmware and the KVM hypervisor with Linux guests

Firmware and software designed and developed by the IBM Linux Technology Center

“OpenPOWER needs secure and trusted boot!”

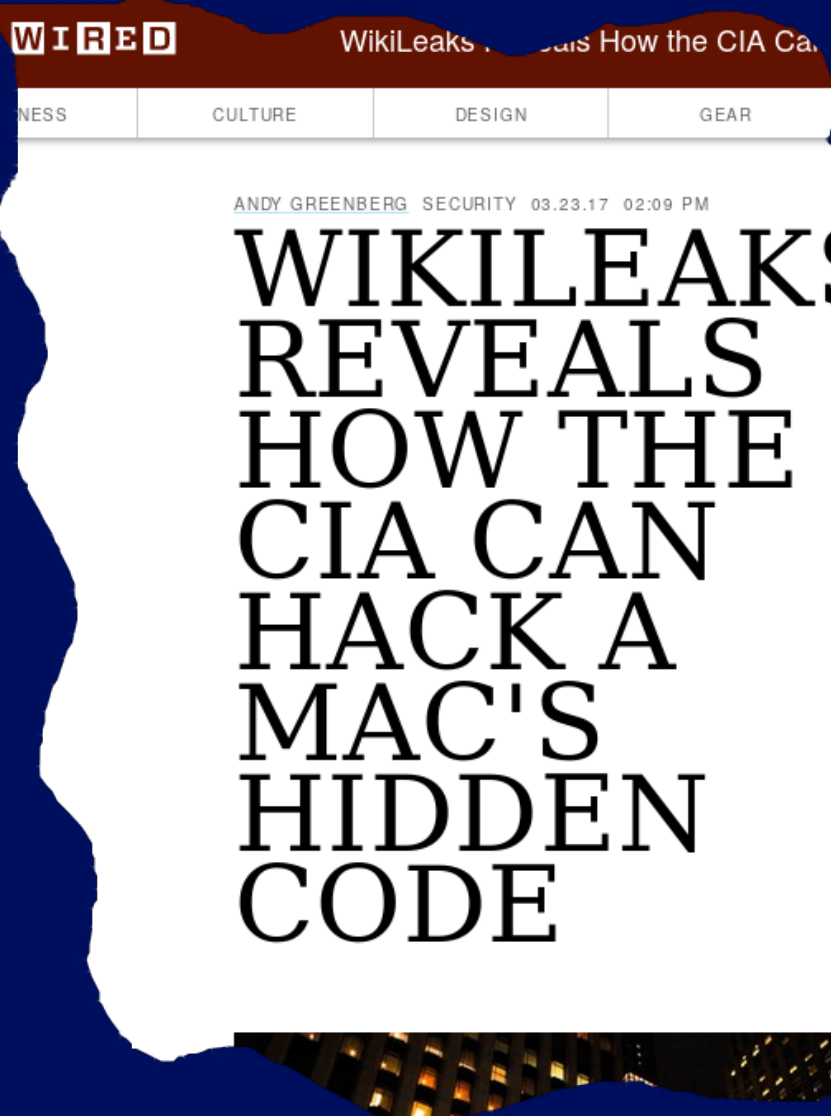


The Case for Firmware Security

Leaks

Wikileaks Vault 7 Year 0 Dump

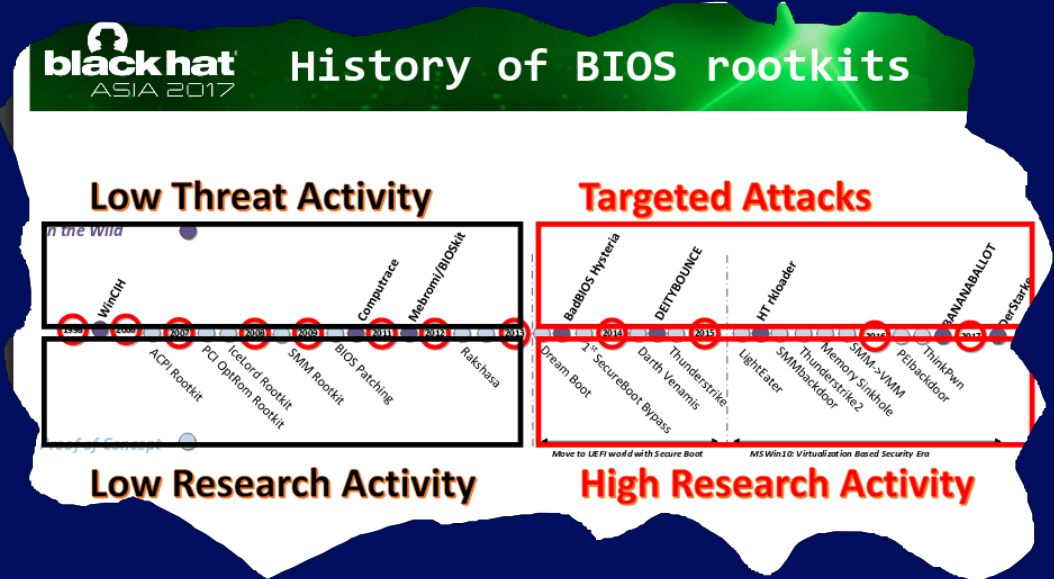
NSA ANT Catalog



Industry Surveys

UEFI Firmware Rootkits: Myths and Reality – Matrosov

Firmware Is the New Black – Analyzing Past Three Years of BIOS/UEFI Security Vulnerabilities – Branco et al.



Dataset & Methodology (1/2)

- It is quite hard to provide really good analysis of datasets related to security issues:
 - A must see if somehow you wonder why: <https://media.blackhat.com/us-13/US-13-Martin-Buying-Into-The-Bias-Why-Vulnerability-Statistics-Suck-Slides.pdf>
- We used Intel's PSIRT Data in different ways:
 - For the past 3 years (**124** issues related to BIOS/Firmware), to give an 'idea' of the amount of issues (which adds attrition bias as well)
 - For as long as we could find (Circa 2007), to generate taxonomy

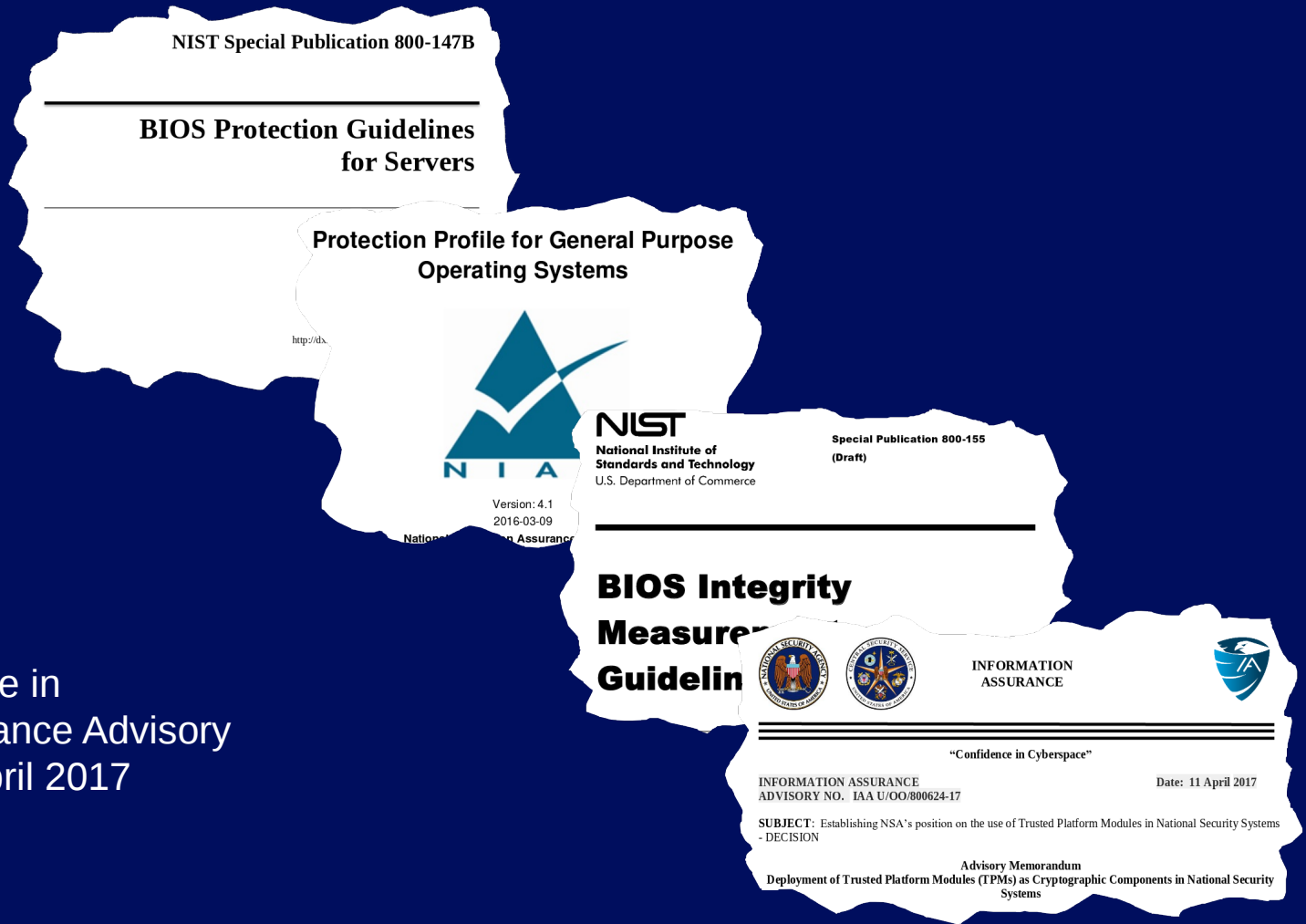
Standards

Secure Boot

NIST SP 800-147B
Common Criteria OSPP 4.1

Trusted Computing

NIST SP800-155 (Draft)
NSA Advisory Memo “Confidence in
Cyberspace”, Information Assurance Advisory
No. IAA U/OO/800624-17, 11 April 2017



Ecosystem

UEFI



Unified Extensible Firmware Interface Specification

What OpenPOWER Is

The OpenPOWER Foundation

In 2013 IBM announced the formation of the OpenPOWER Foundation

Current Members: 8 Platinum, 5 Gold, 117 Silver, 112 Associate/Academic, 73 Individual

IP sharing, access to specifications, working group participation, etc.



Platinum Level



Gold Level



More Open Hardware

CPU designs are available to members

System reference designs are available to members

Custom machines can be built based upon the reference designs

Members can shape designs via working groups; a few WGs are public



Open Firmware

OpenPOWER firmware is developed based on open source methodologies reusing as much open source code as possible

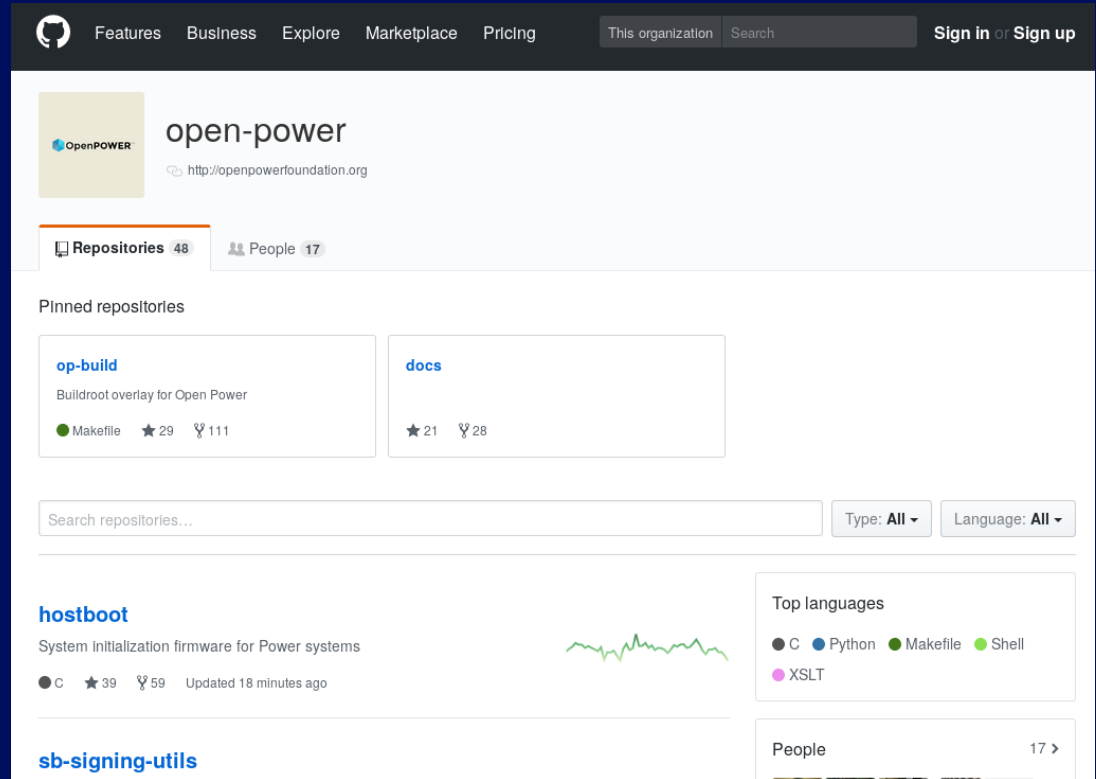
We want to maintain only OpenPOWER specific code

Code generally must be accepted upstream before it can be incorporated into the OpenPOWER build

Hosted on OpenPOWER github site

You can see what's in your firmware stack

You can rebuild and reflash your firmware



The screenshot shows the GitHub organization page for 'open-power'. The header includes navigation links: Features, Business, Explore, Marketplace, Pricing, This organization, Search, and Sign in or Sign up. The organization's profile shows the OpenPOWER logo, the name 'open-power', and the website 'http://openpowerfoundation.org'. Below this, there are tabs for 'Repositories 48' and 'People 17'. The 'Pinned repositories' section displays three repositories: 'op-build' (Buildroot overlay for Open Power, 29 stars, 111 forks), 'docs' (21 stars, 28 forks), and 'hostboot' (System initialization firmware for Power systems, 39 stars, 59 forks, updated 18 minutes ago). A search bar for repositories is located below the pinned repositories. On the right side, there is a 'Top languages' section showing C, Python, Makefile, Shell, and XSLT, and a 'People' section showing 17 members.

Boot Flow Components

Self Boot Engine (SBE): Pre-firmware bootloader; initializes pervasive bus, PNOR

OTPROM: Stores early immutable code and data, including signature verification routines

SEEPROM Stores early updatable code and data, including HW key hash; lockable once

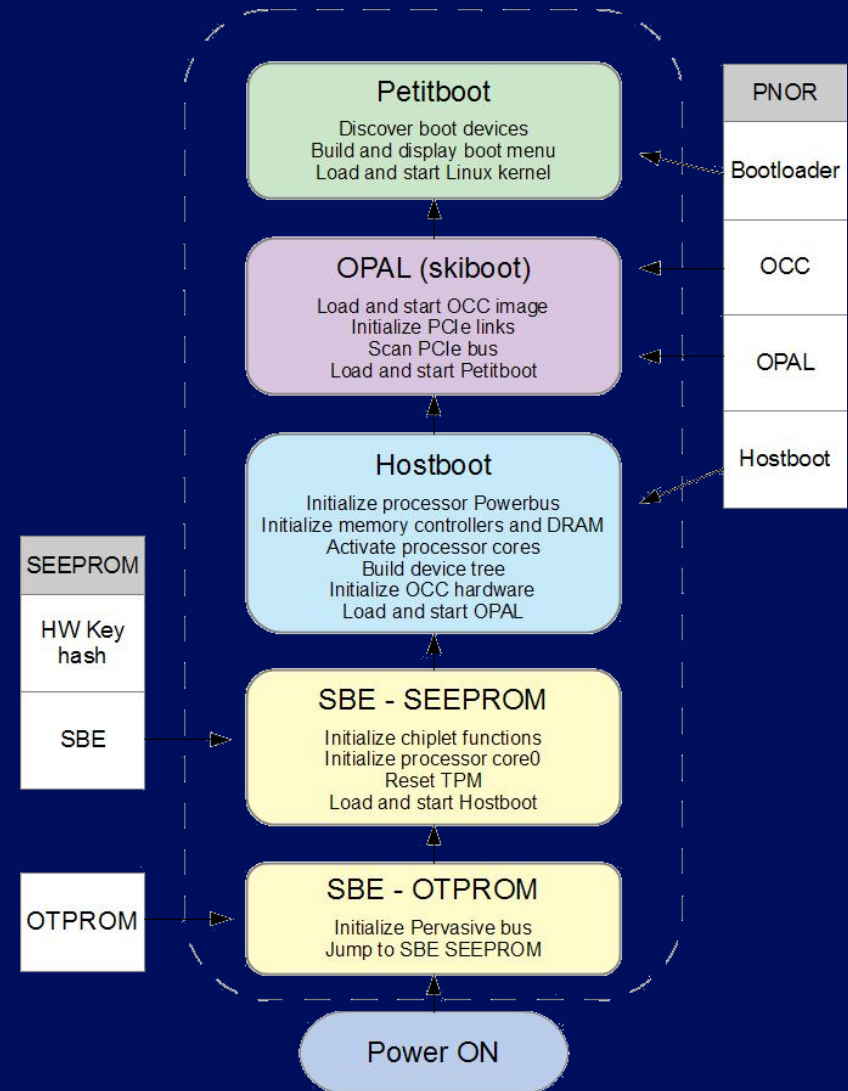
Processor NOR (PNOR): Holds images/data

Hostboot: First stage firmware brings up internal buses, DRAM, etc.

Skiboot: Brings up PCIe, provides OPAL

Skiroot: Linux kernel with embedded initramfs that runs Petitboot application

Petitboot: OS bootloader – uses kexec



Trusted Computing in OpenPOWER

Firmware

TCG Server Specification largely defers to PC Client Specification – there's room for new TCG work on this

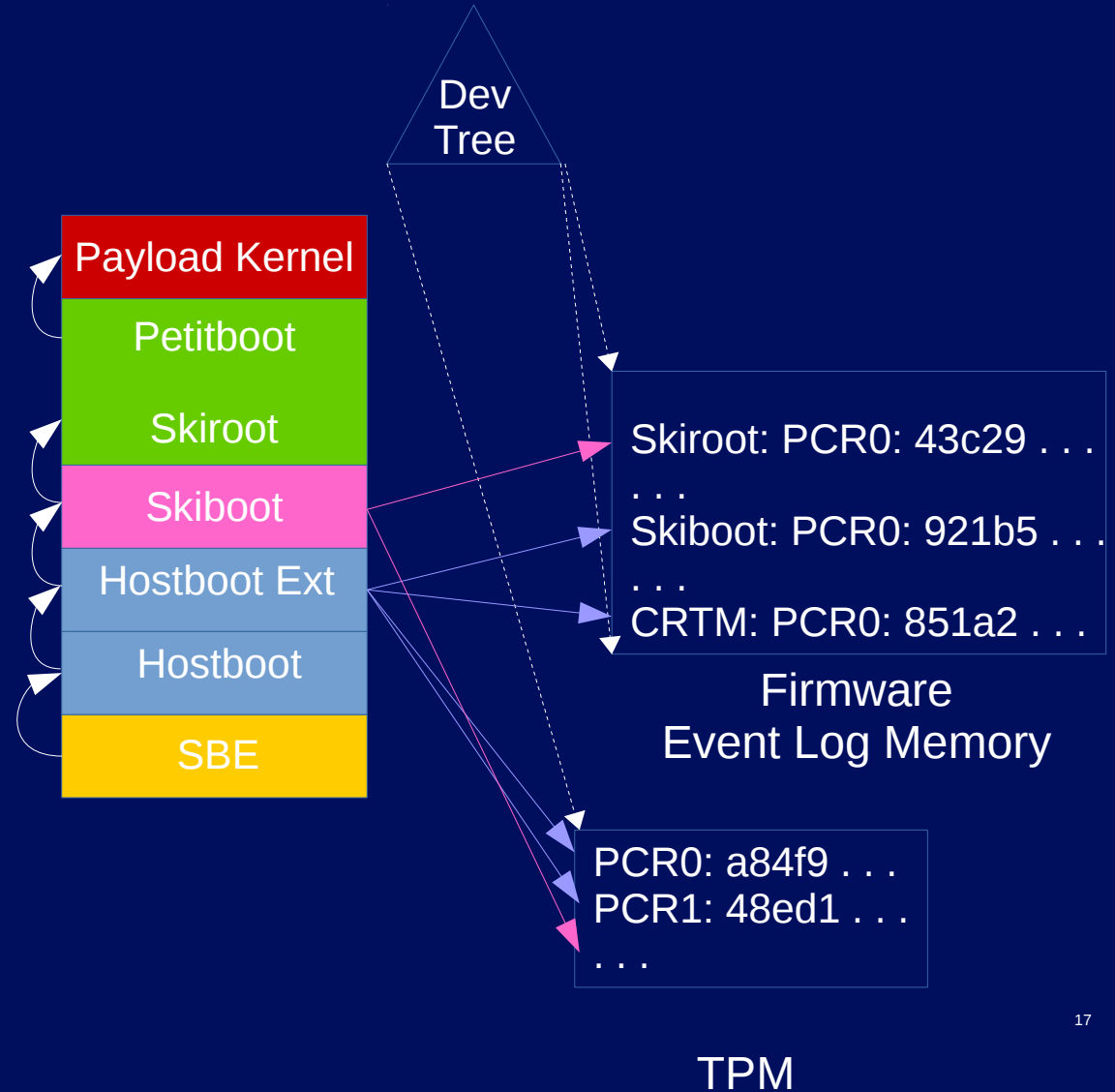
Applied TCG PC Client and UEFI Specifications to the extent possible for OpenPOWER

Attempted to mirror logging and PCR semantics

Device Tree based – another area ripe for new TCG work

The TPM and event log memory region references are maintained in the Device Tree

SRTM only presently



Firmware (cont'd)

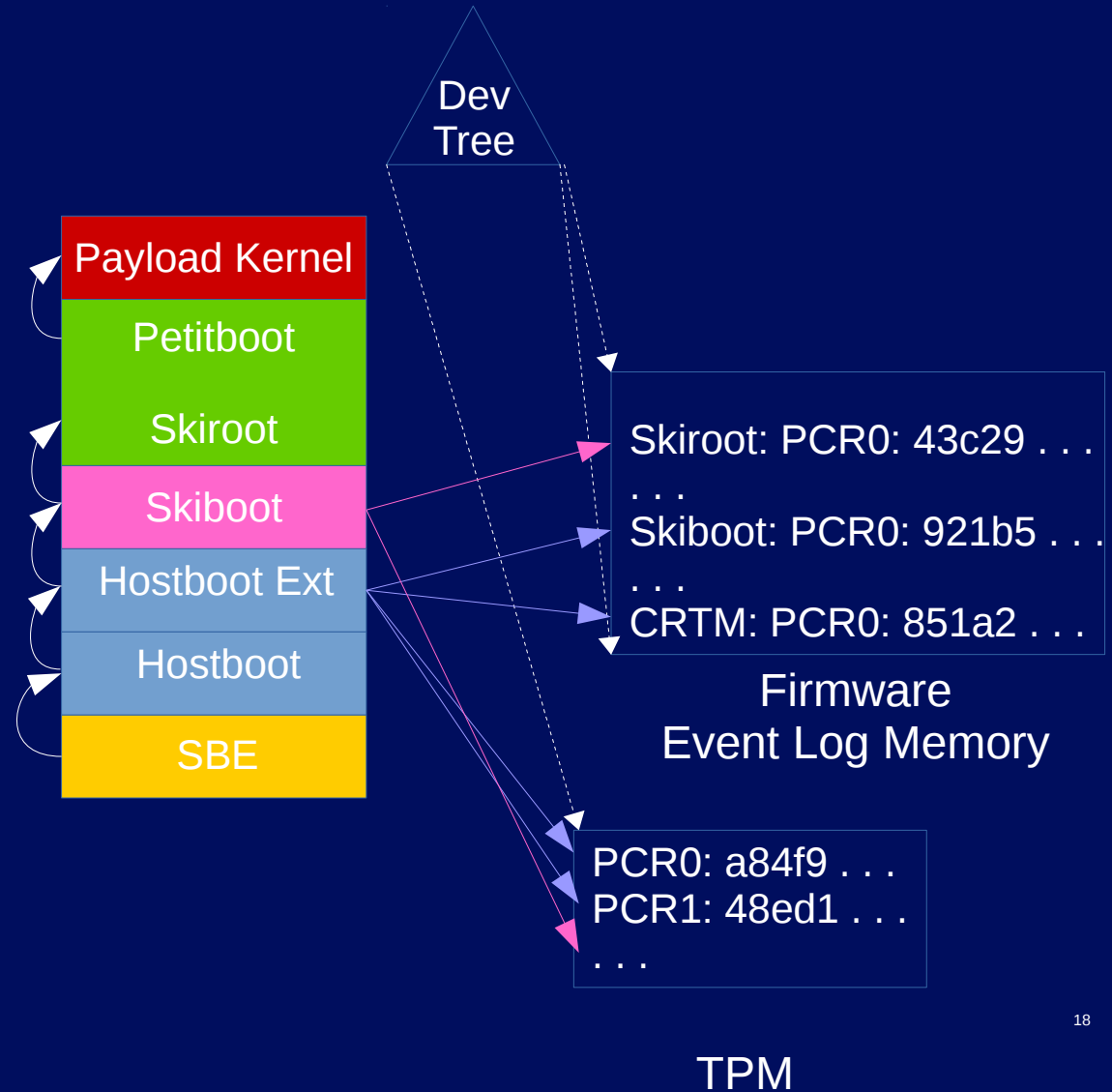
Measurement starts with SBE measuring Hostboot

First measurements are logged in Hostboot Extended

CRTM comprises SBE, ROM code, and all of Hostboot

Secure Boot is necessary to ensure that CRTM is intact

Firmware measurement chain is complete after Skiboot measures Skiroot



Software

Skiroot is the transition point between firmware and software measurements

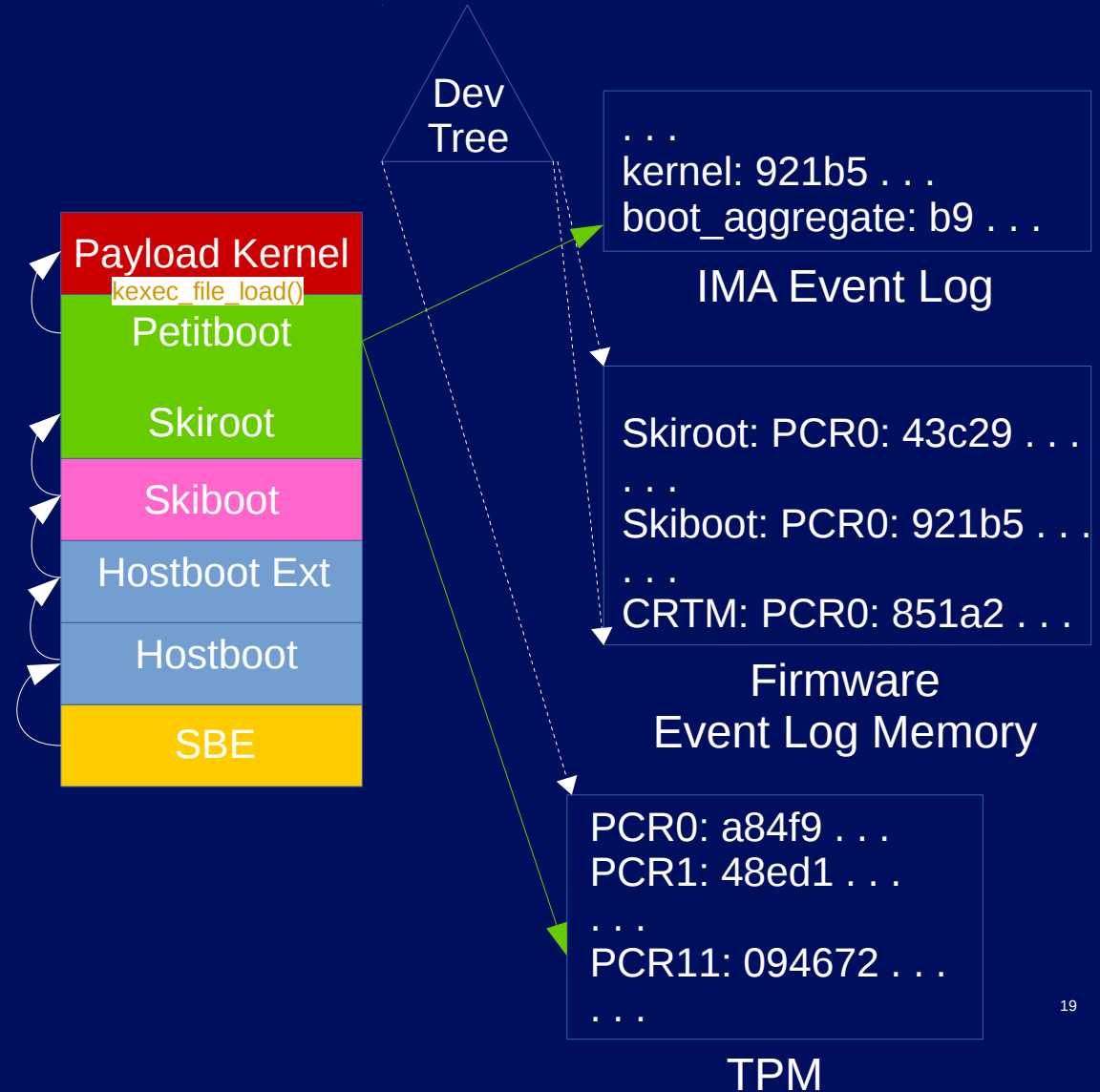
Skiroot is a Linux kernel that contains an embedded intramfs that runs the Petitboot application from init

Petitboot launches the host OS payload kernel via `kexec_file_load()`

`kexec_file_load()` measures the host kernel

TPM device data and firmware event log references are passed to the payload kernel via the device tree

Skiroot relies on IMA for measurements



Software (cont'd)

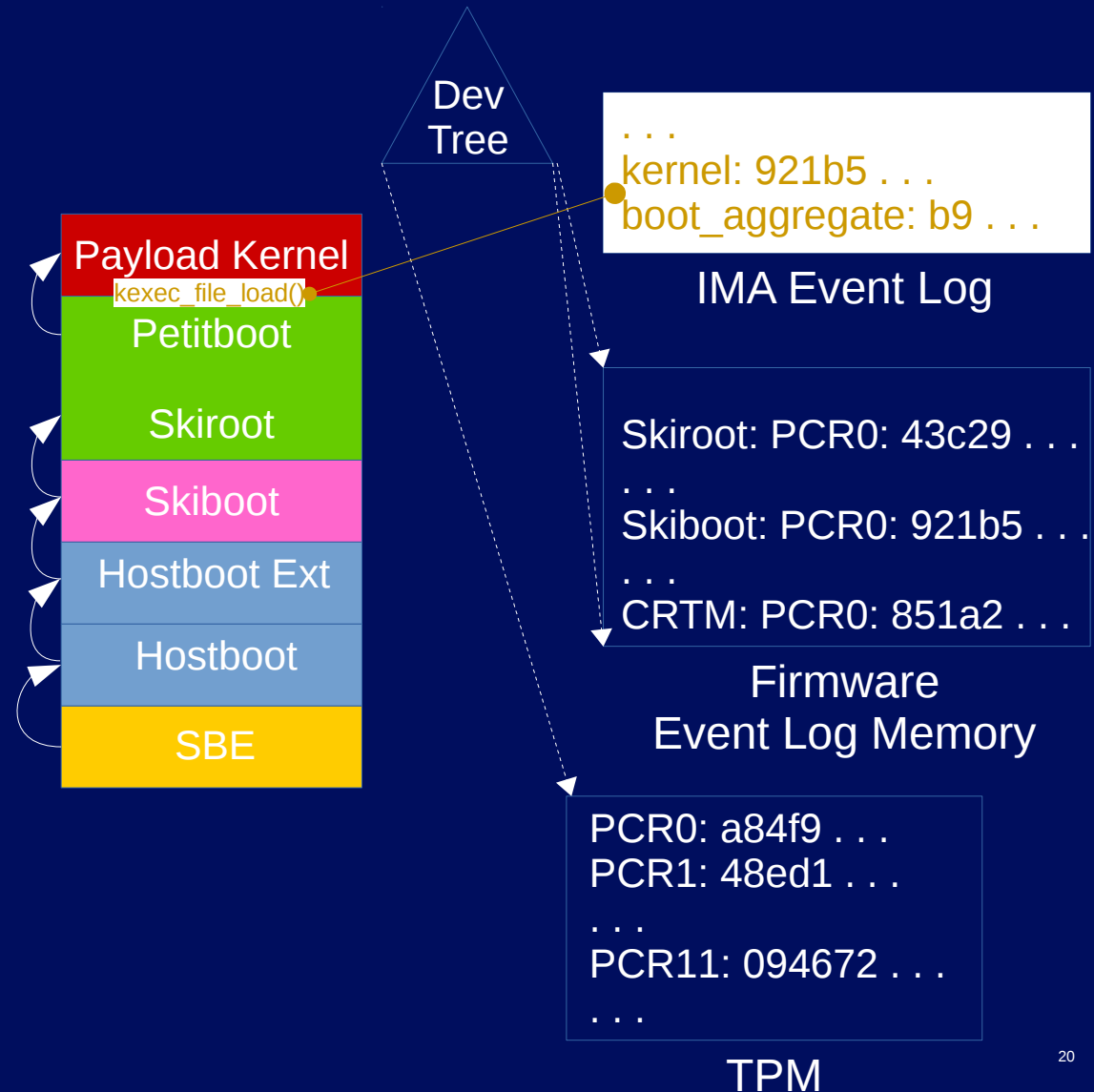
The IMA event log is maintained separately from the firmware event log

`kexec_file_load()` passes the IMA event log to the payload kernel

If the payload kernel supports `kexec_file_load()` event log passing, kernels chained by `kexec` can pass the event log to the next kernel as well

There may be other cases where passing state to the next kernel is helpful, perhaps the kernel keyring for self-encrypting drives

Extending trusted computing to guests requires a vTPM integration with QEMU and guest firmware support



Control and Attestation

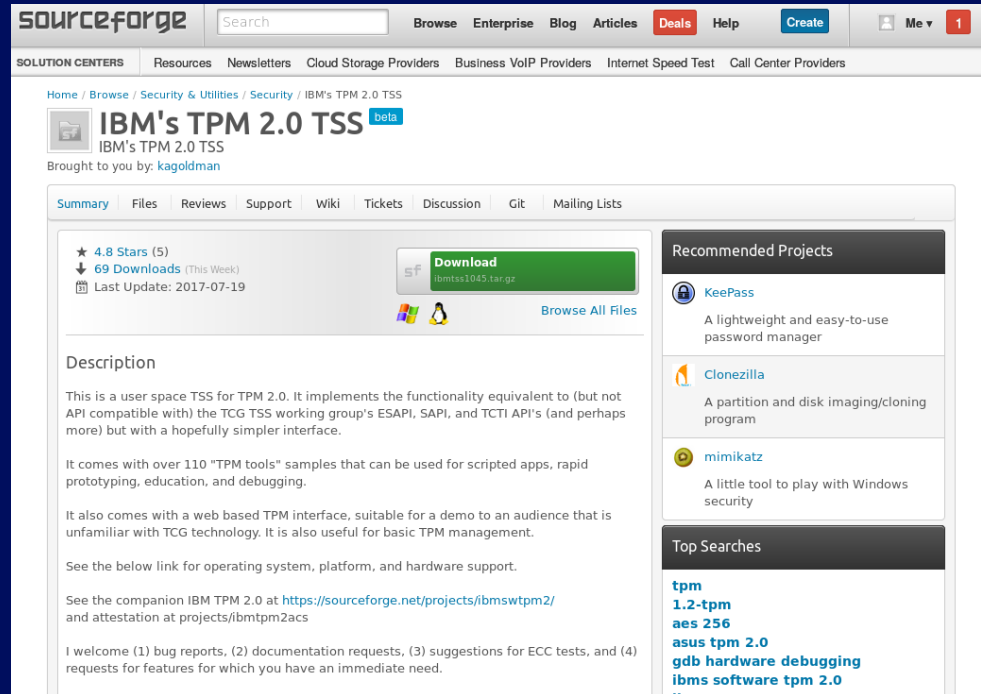
The TCG TSS 2.0 WG specification and reference implementation have taken a long time to materialize

We're currently relying on the IBM Research TSS 2.0

It lacks a resource manager but is sufficient for our purposes

tpmdd-devel discussion makes a good case for the resource manager to reside in the kernel rather than in userspace

We will consider moving to a spec-compliant TSS 2.0 in the future



Control and Attestation (cont'd)
























We've also needed a way to make use of measurements

We're using the IBM Research Attestation Client/Server

It's currently a proof of concept but allows basic remote attestation

We may consider using the Intel OpenAttestation project at a later time

Eventually, we plan to publish reference manifests in the TCG prescribed format for each firmware level

TPM 2.0 Attestation Machines					
Ken Goldman, IBM Research, kgoldman@us.ibm.com Attestation Server: cainl.watson.ibm.com					
Machines Reports					
Enrolled Machines					
Machine	TPM Vendor	Enrolled	EK Certificate	AK Certificate	Boot Time
cainlec.watson.ibm.com	IBM	2017-03-30 17:36:49			2016-03-21 09:08:24
cainl.watson.ibm.com	IBM	2017-03-30 17:36:37			2016-03-21 09:08:24
carolineec	NTC	2017-03-29 11:32:39			2017-03-28 16:22:07
caroline	NTC	2017-03-29 11:32:03			2017-03-28 16:22:07
dogbertec	IFX	2017-03-27 18:41:40			2017-03-20 16:35:16
dogbert	IFX	2017-03-27 18:40:50			2017-03-20 16:35:16
netsec24ec	STM	2017-02-01 11:10:19			2017-01-11 17:22:44
netsec24	STM	2017-02-01 11:09:36			2017-01-11 17:22:44
systemz	IBM	2017-01-17 13:23:51			2016-03-21 09:08:24
abell	IBM	2016-08-24 13:49:37			2016-08-24 13:25:40
cit1012.rtp.raleigh.ibm.com	IBM	2016-08-23 17:29:23			2016-08-19 18:25:03
sbct-3.watson.ibm.com	IBM	2016-08-15 13:13:19			2016-06-04 21:39:34
hab20	NTC	2016-05-31 15:02:27			2016-05-31 13:47:13
Copyright © IBM 2016					

Secure Boot in OpenPOWER

Major Design Parameters

Learn from UEFI

Device tree

No secure, dynamically lockable storage

No SMM

There is a TPM

Two separate domains with separate
authorities: firmware and software

Firmware Signing

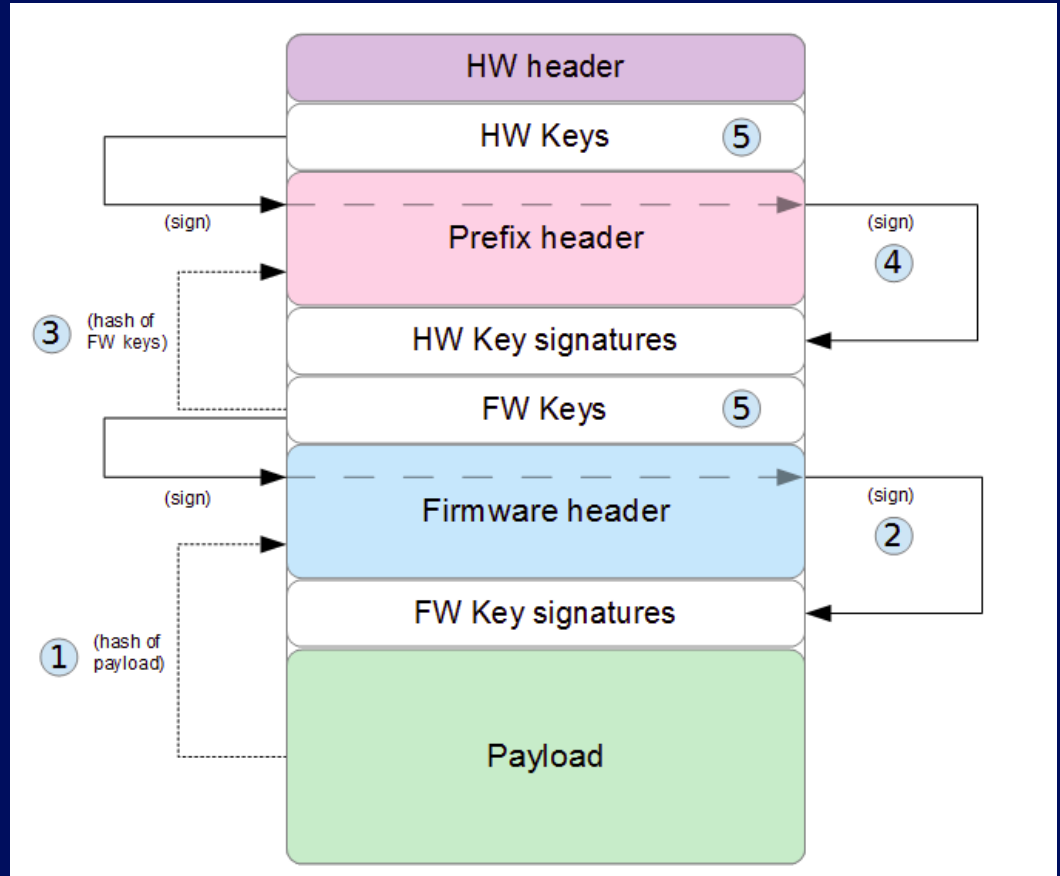
Each firmware component is encapsulated in a Secure Boot “container” structure that is signed by an 512-bit ECDSA key generated by the firmware supplier

Containers actually contain 3 keys to accommodate privilege separation

IBM signing keys are maintained on a 4767 HSM

Hardware keys sign firmware keys

Hash of hardware keys is kept in SEEPROM



Firmware Verification

Verification begins with SBE checking Hostboot

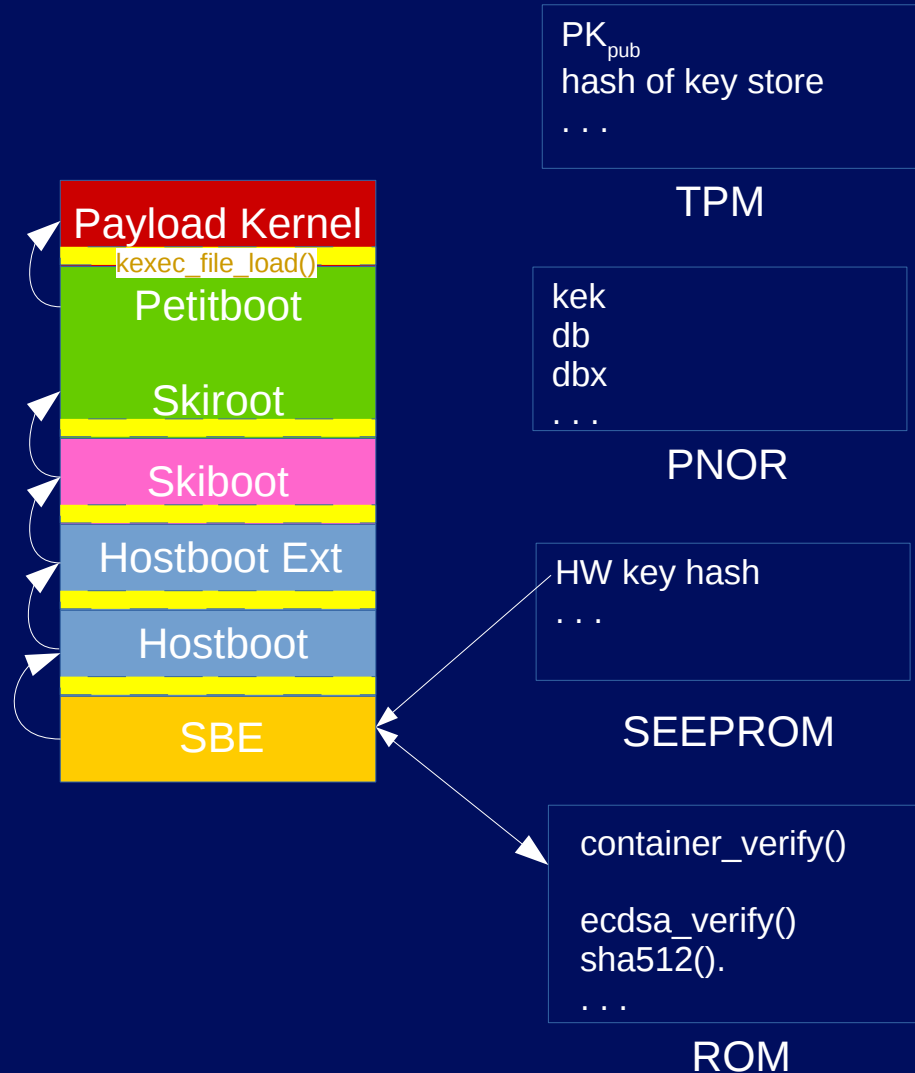
Chain begins in immutable code in ROM

The container verification comprises:

- Verify payload using FW keys
- Verify FW keys using HW keys
- Verify HW keys with hash in SEEPROM

Hash of HW keys is stored in locked SEEPROM that can't be updated without authentication unless security jumper is added

Firmware verification is complete with verification of Skiroot



Software Signing

The OS kernel is signed with sign-file – the same tool used to sign kernel modules

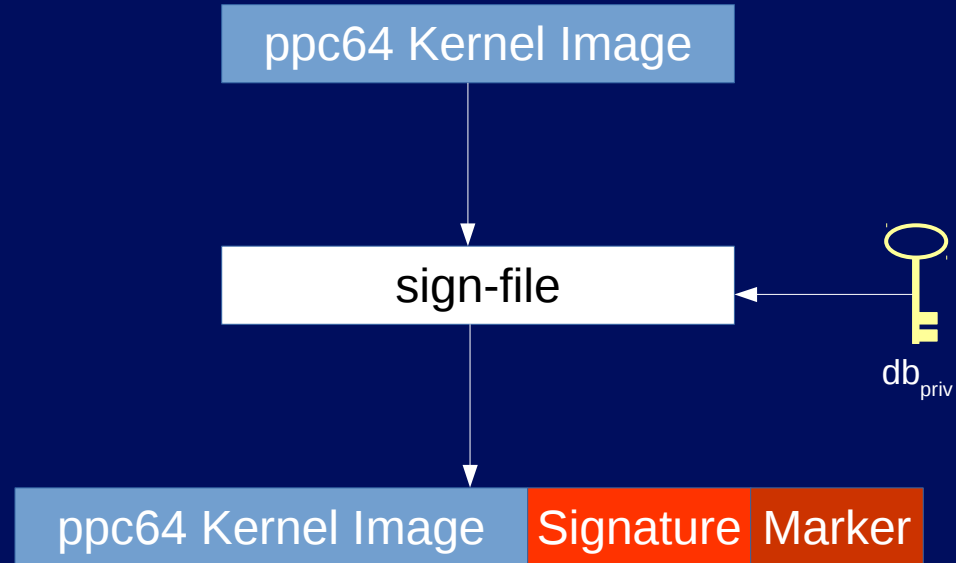
Appended signature provides backwards compatibility with unsigned legacy kernels

Sticking with UEFI RSA 2048 with SHA-256

There's potential to reuse UEFI shim keys

The initramfs is not signed because it is volatile – there is a use case for IMA-appraisal here but it requires kernel cpio support for xattrs – or overlay intramfs

Host OS signature needs to be compatible with guest signatures – both QEMU and PowerVM



Software Key Management

No central CA and no shim – admin is in full control for better or worse

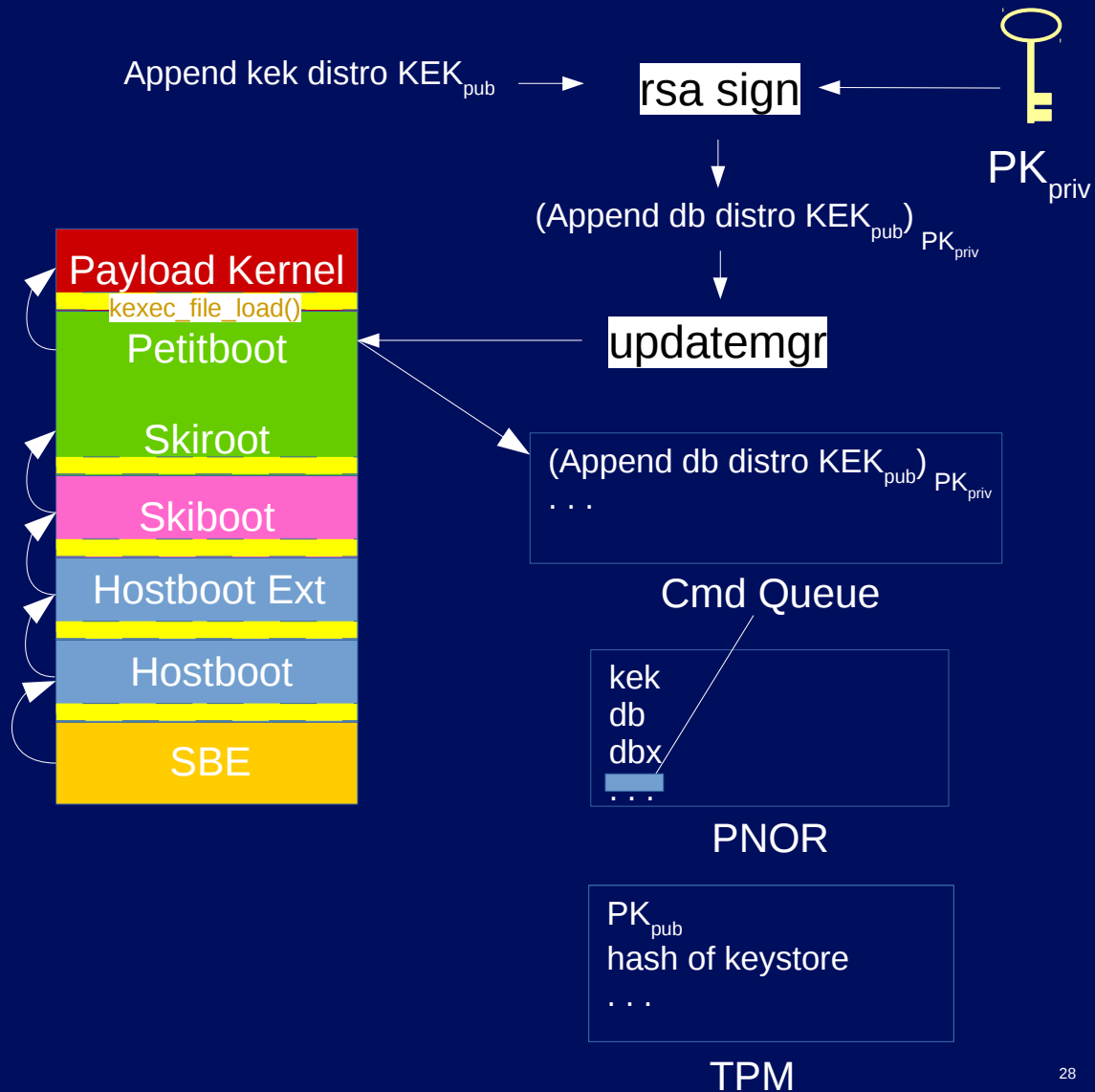
Intent is to support 3 scenarios:

- Admin built and signed OS
- Admin installation of distro keys
- Manufacturing installation of distro keys

Keystore modeled on UEFI secure boot variables and stored in ESL format: pk, kek, db, dbx

Because there is no secure lockable storage other than the TPM, we place the keystore in PNOR with check hashes in TPM NVRAM

Keystore must be loaded and checked against TPM hash before use



Software Key Management (cont'd)

Design corresponds to scenario 3 in NIST SP 800-147B

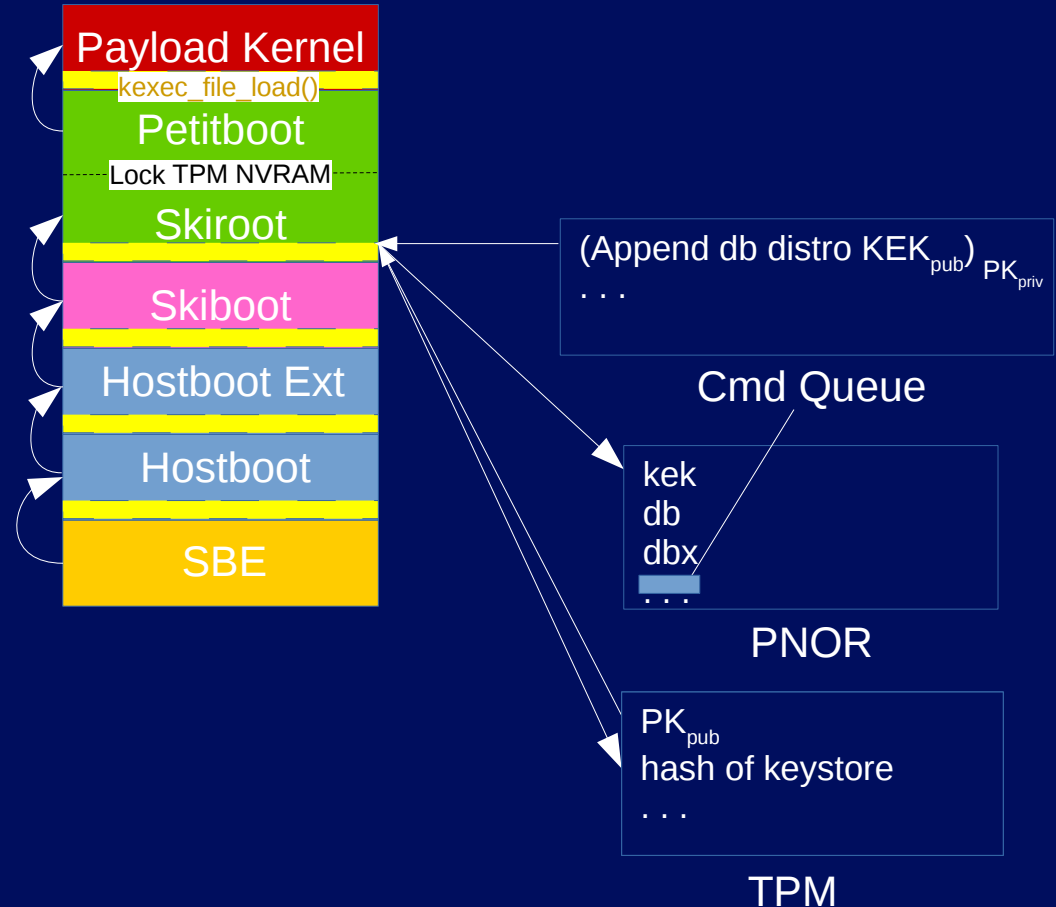
Key management commands are signed by PK or KEK private key and placed into a command queue

All updates require a system reboot

Updates are verified using the PK public and processed by early skiroot before Petitboot is executed, then the TPM NVRAM is locked from further changes

Firmware can race against the BMC but we can detect corruption

Software keys are invalidated when underlying hardware keys change



Software Verification

Software verification starts with Petitboot

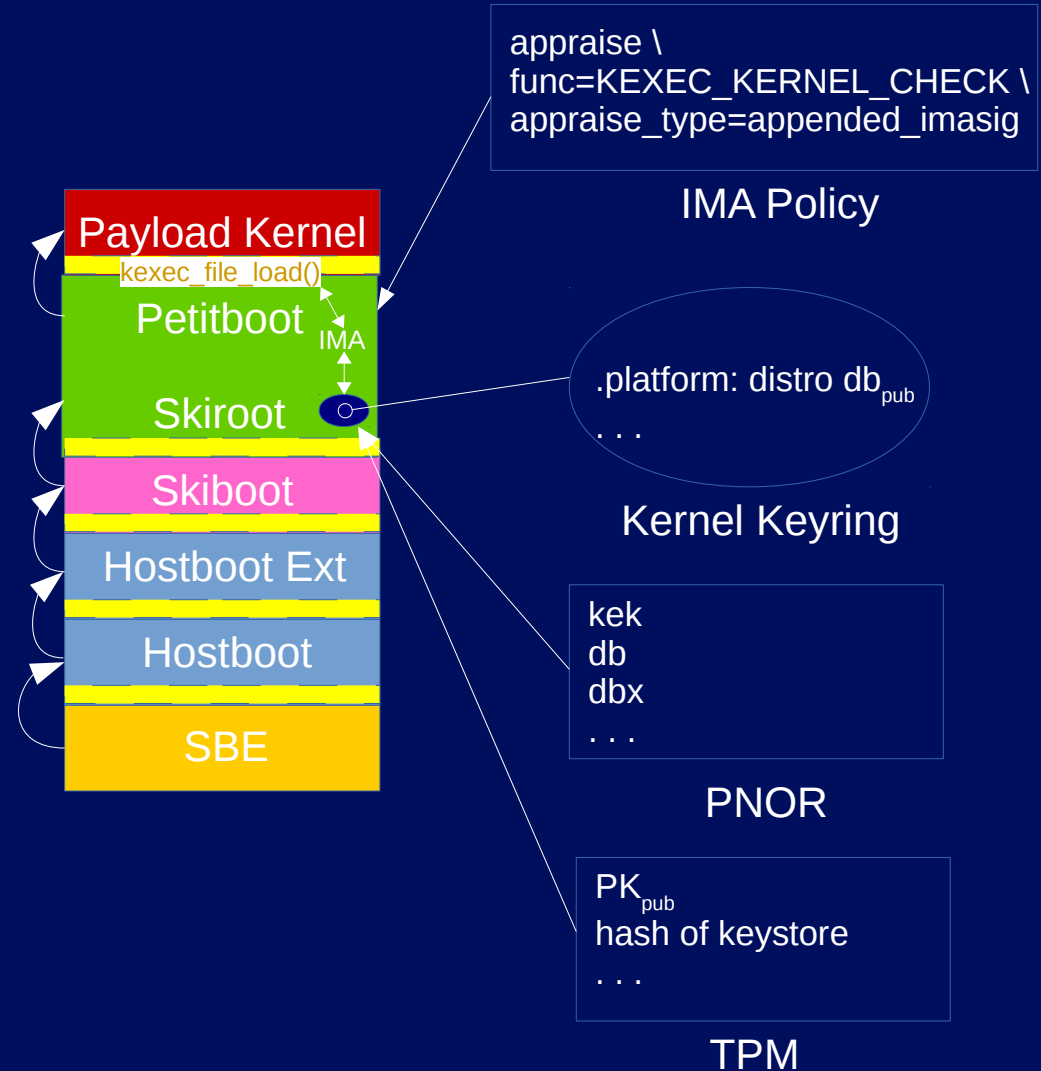
Petitboot now uses `kexec_file_load` to load host OS kernel

`kexec_file_load` must be able to verify the host OS kernel

IMA-appraisal can verify the kernel but requires extended attributes

IMA-appraisal patches extend IMA to verify the kernel using an appended signature

Future mechanism will implement kernel verification independent of IMA per ppc tree maintainer request



UI and Key Updates

We need the ability to toggle the secure boot state so admins can boot unsigned kernels or simply decide they don't want secure boot

Petitboot currently has no authentication – we're considering using TPM authentication of an empty NVRAM index for this purpose

We'd eventually like to further lock down the Petitboot environment with rsh or sudo, IMA-appraisal, and SELinux – would be useful controls for dedicated KVM host OS as well

Signing of key updates will be performed offline

```
/ # updatemgr
Available Commands:
  SetVariable
  ClearQueue
  GetCommand
  GetCommandInfo
  GetQueueLen
  GetQueue

/ # updatemgr SetVariable
Usage: SetVariable [-a] -t <timestamp> -v <var> -s <sign file> -f <var file>
Submit an Authenticated Variable to Update Queue
Options:
  -a          Perform an append rather than replacement
  -t <timestamp> Use <timestamp> as the timestamp of the timed variable update
  -v <var name>  Secure Variable Name
  -f <var file>  File containing the ESL or data values
  -s <sign file> File containing the Signature from the Authenticated Variable data
Error: Success
```

UI and Key Updates (cont'd)

With networking setup in Petitboot, admin can fetch signed key updates using wget from the Petitboot shell

Petitboot will accept key update commands, place them in the command queue, and facilitate viewing errors

Upon reboot, early skiroot validates updates using PK or a KEK public key

```
[ 17.692234] Key type encrypted registered
[ 17.695572] rtc-generic rtc-generic: setting system clock to 2017-09-06 20:07:42 U
[ 17.778379] Freeing unused kernel memory: 4288K
[ 17.778486] This architecture does not have kernel memory protection.
==> Running (trusted init)
==> Creating PK trust store keyring
==> Adding PK certificate </etc/keys/PK/PK-der.crt> to PK keyring
==> Adding KEK certificate </etc/keys/KEK/KEK-0-der.crt> to KEK keyring
==> Processing Update queue entry 1
==> Using key-id 273982420 to validate signature ... OK
==> Processing Update queue entry 2
==> Using key-id 273982420 to validate signature ... FAILED
==> Using key-id 1009726545 to validate signature ... OK
==> Performing cleanups
keyring 'pk-truststore-certs'
purged 1 keys
keyring 'kek-truststore-certs'
purged 1 keys
==> Rebooting for Trust Store changes to take effect
==> Process update Queue completed
[ 20.009572] udevd[159]: starting version 3.2.2
[ 20.023330] random: udevd: uninitialized urandom read (16 bytes read)
[ 20.026479] random: udevd: uninitialized urandom read (16 bytes read)
[ 20.029666] random: udevd: uninitialized urandom read (16 bytes read)
[ 20.030301] random: udevd: uninitialized urandom read (16 bytes read)
```


Future, Status, Challenges

Host OS and Beyond

The Host OS should be locked down to prevent loading unsigned kernel modules

Subsequent kexec'd kernels need to be verified – required at least for crash dump kernels

Insecure kexec_load() should be prohibited

IMA-appraisal can extend secure boot chain of trust into Host OS – this would be easier if distros shipped signatures in package metadata and laid them down during install

Passing IMA event log to next level kexec'd kernels can provide full Trusted Computing chain

Guest secure boot and trusted boot should be considered

DRTM could be useful – it, however, requires MPIOPL foundation, which hasn't been implemented for OpenPOWER as it hasn't been required

Current Status

Secure Boot and Trusted Boot up to Skirroot shipped in December 2016 on select models

Will replicate POWER 8 work for POWER 9

Still designing Secure Boot in detail

Socializing design with distro partners, OpenPOWER WGs, and open source communities

Working towards a tentative 2Q18 release

Challenges

None of us have designed and implemented Trusted Boot or Secure Boot before – working closely with IBM Research Secure Processor team, OPAL, and Power Firmware teams – decisions from PCR semantics to how to sign and verify the ppc kernel

Interleaving design and development

OpenPOWER design

Early hardware limitations

Coordination with Test, Documentation, and Manufacturing

Coordination with distros

Schedule

Benefits of Open Source Software

Firmware

Apache licensed

Can utilize existing compatibly licensed open source packages, such as OpenSSL

Skirroot/Petitboot/Kernel

Already a full Linux environment

IMA does the bulk of the work for both measurement and enforcement

Reuse of module signing mechanism for kernel

ESL key import

Kernel crypto API including recent asymmetric algorithm support in 4.14 and keyctl

OpenSSL

Benefits to Open Source Software

Trusted Computing

TPM device driver

- msleep() vs usleep_range()

- Only sleep on retry

- Burstcount

- Drivers built into distro kernels

- Extensions for TPM 2.0

IMA

- Stress testing

- Extensions for TPM 2.0

- Passing of IMA event log across kexec

Control and Attestation

- TPM 2.0 TSS

- TPM 2.0 attestation client/server

QEMU/libvirt

- Virtual TPM and virtual BIOS measurements

Secure Boot

Kernel

- Appended signature
- kexec_file_load() verification
- Platform keyring
- Kernel lockdown

IMA-appraisal

- IMA signature verification
- Kernel cpio xattr support

Petitboot

- Authentication
- Key management interface
- Additional locked down configuration

Host OS

- Additional locked down configuration

QEMU

- Virtual secure boot

Conclusion

There is strong rationale for firmware security

We think the machine owner should truly be the machine owner

Developing secure boot from scratch isn't easy

Learning from other firmware implementations is valuable

Reusing as many (hopefully) well tested components as possible should generally be better than developing them from scratch

Having fully open source firmware makes life easier all the way around

Arch- or problem-specific features can help the general art progress

References

OpenPOWER

General

OpenPOWER Foundation: <https://openpowerfoundation.org/>

OpenPOWER Github: <https://github.com/open-power>

Secure and Trusted Boot

OpenPOWER secure and trusted boot, Part 1 – Using trusted boot on IBM OpenPOWER servers:
<https://www.ibm.com/developerworks/library/l-trusted-boot-openPOWER-trs/index.html>

OpenPOWER secure and trusted boot, Part 2 –Protecting system firmware with OpenPOWER secure boot: <https://www.ibm.com/developerworks/library/l-protect-system-firmware-openpower/index.html>

Trusted Computing

IBM's TPM 2.0 Attestation Client/Server: <https://sourceforge.net/projects/ibmtpm20acs/>

IBM's TPM 2.0 TSS: <https://sourceforge.net/projects/ibmtpm20tss/>

Standards

Government

Common Criteria OSPP 4.1: <https://www.niap-ccevs.org/Profile/Info.cfm?id=400>

NIST SP 800-147B: <http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-147B.pdf>

TCG

TCG PC Client Platform Firmware Profile Specification:

<https://trustedcomputinggroup.org/pc-client-specific-platform-firmware-profile-specification>

PC Client Platform TPM Profile (PTP) Specification:

<https://trustedcomputinggroup.org/pc-client-platform-tpm-profile-ptp-specification/>

TCG EFI Protocol Specification: <https://trustedcomputinggroup.org/tcg-efi-protocol-specification/>

UEFI

UEFI Specification Version 2.7: http://www.uefi.org/sites/default/files/resources/UEFI_Spec_2_7.pdf

Firmware Threats

Leaks

NSA ANT Catalog: https://www.eff.org/files/2014/01/06/20131230-appelbaum-nsa_ant_catalog.pdf

Wikileaks Year 0 Vault 7: <https://wikileaks.org/ciav7p1/>

Surveys

Firmware Is the New Black – Analyzing Past Three Years of BIOS/UEFI Security Vulnerabilities:
<https://github.com/rrbranco/BlackHat2017/blob/master/BlackHat2017-BlackBIOS-v0.13-Published.pdf>

UEFI Firmware Rootkits: Myths and Reality:
<https://www.blackhat.com/docs/asia-17/materials/asia-17-Matrosov-The-UEFI-Firmware-Rootkits-Myths-And-Reality.pdf>

Questions?

Thank you!

George Wilson
<gcwilson@us.ibm.com>
IBM Linux Technology Center