Defeating Invisible Enemies: Firmware 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 1991

Introduced in the RS/6000 line

Usage presently spans embedded to server

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

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

OpenPOWER runs open source firmware and KVM with Linux guests

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

“OpenPOWER needs security!”
The Case for Firmware Security
Leaks

Wikileaks Vault 7 Year 0 Dump

NSA ANT Catalog
Industry Surveys

UEFI Firmware Rootkits: Myths and Reality

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

- It is quite hard to provide really good analysis of datasets related to security issues:

- 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.
More Open Hardware

CPU designs are available to members

System reference designs are available to members; some more widely

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.
Boot Flow Components

**Self Boot Engine (SBE):** Pre-firmware bootloader

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

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

**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
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 rather than ACPI 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

---

**Payload Kernel**

- Petitboot
- Skiroot
- Skiboot
- Hostboot Ext
- Hostboot
- SBE

**Dev Tree**

- Skiroot: PCR2: 43c29 . . .
- Skiboot: PCR0: 921b5 . . .
- CRTM: PCR0: 851a2 . . .

**Firmware Event Log Memory**

- PCR0: a84f9 . . .
- PCR1: 48ed1 . . .

---

Linux Security Summit / Defeating Invisible Enemies / September 15, 2017 / © 2017 IBM Corporation
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 as init

Petitboot launches the host OS payload kernel via kexec_file_load()

Petitboot 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 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.

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

But it lacks a resource manager
tpmd-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.
Secure Boot in OpenPOWER
Major Design Parameters

Learn from UEFI

Device tree

No secure, 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 key transitions.

IBM signing keys are maintained on a 4767 HSM.
Firmware Verification

Hostboot is protected by a hash kept in SEEPROM

Verification begins with SBE checking Hostboot

Actual check is performed by immutable code in ROM

Hash of Hostboot 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

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

Keys must be loaded and checked against TPM hashes before use
Software Key Management (cont’d)

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

Key management commands are signed by PK private and placed into a command queue.

All updates require a system reboot.

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

Firmware can race against the BMC but we can detect corruption.

---

Linux Security Summit / Defeating Invisible Enemies / September 15, 2017 / © 2017 IBM Corporation
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
User Interface and Controls

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.

Updates are signed with PK private key.

Signing of key updates will typically be performed offline.

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 public key.
Future, Status, Challenges
Host OS and Beyond

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

Subsequent kexe'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 MPIPL foundation which hasn’t been implemented for OpenPOWER as it hasn’t been required
Current Status

Secure Boot and Trusted Boot up to Skiroot 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, OPAL, and Power Firmware teams.

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
Skiroot/Petitboot/Kernel

Already a full Linux environment

IMA merely needs to be switched on for the majority

ESL key import

Signature verification in kernel with X.509 certs
Benefits to Open Source Software
Trusted Computing

TPM device driver performance
- 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
Secure Boot

Kernel

- Appended signature
- kexec_file_load() verification
- Additional keyrings
- 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

Secure boot is hard to do

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


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-tpp-specification/

**UEFI**
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


Questions?
Thank you!