Project Cerberus
Hardware Security

Bryan Kelly / Principal Firmware Eng Manager
Microsoft Azure Cloud Hardware Infrastructure

Yigal Edery / Principal Program Manager
Microsoft Azure Security
Talk Outline

• Firmware Security for the Cloud
• Current State of the Industry
• Project Cerberus deconstruct
• Open Compute Security Project
OCP Security Context

“Rogues are very keen in their profession, and know already much more than we can teach them” -- Alfred Charles Hobbs, 1851

Open Compute Security Project was announced in February 2018

Microsoft and Google selected to serve as co-chairs, many companies contribute

Community focused on advancing platform hardware and firmware security
The Firmware Security Challenge...

If you Own the firmware,
...You Own the server,
And you Own the cloud...
...And no one will find you...
The Hardware Threat is real!

- Gives attackers full control
- Getting weaponized (read: easy)
- Hard to detect
- Hard to remediate
The Cloud Firmware Threat Vectors

Vectors:

- **Internet** (Compromised customer, or malicious)
  - Hypervisor compromise
  - Bare Metal Services

- **Insiders** (Compromised or malicious)
  - Logical & physical access to hardware

- **Supply Chain**
  - Source of hardware
So, what can an attacker do?

**Exploit a run-time firmware vulnerabilities**

- Firmware **quality** is critical (reduce chances of run-time exploitability)
  - Follow secure development practices & Issue security updates with a proper SLA

- Assume breach and implement **defense in depth**
  - Component compartmentalization & limit the damage & reach of possible breaches

**Persist rogue firmware**

- Modify firmware to maintain ‘hold’ of the breached hosts (Survive even formats and OS re-installs)
- Servers need the ability to protect, detect and recover from such attacks

- This is what Cerberus is all about...
### NIST 800-193: Protect, Detect, Recover

<table>
<thead>
<tr>
<th><strong>Authenticate integrity of all firmware updates</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Root(s) of trust &amp; chain(s) of trust across the platform</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Detect unauthorized access or corruption</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Generate traces &amp; events to help detect anomalies</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Restore firmware to state of integrity</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic, Automatable and manual recovery scenarios</td>
</tr>
</tbody>
</table>


The Current State of Industry Servers

- **UEFI** – limited protection
  - Secure-boot-like functionality
  - No Detect or Recover
  - Platform dependent

- **BMC** - typically unsecure
  - No protect, no detect, no recover
  - No reliable attestation
  - E.g. [BlackHat Talk](#)

- **Peripherals** - Ad-hoc, usually limited
  - No protect, detect, recover
  - Not chained to platform RoT
Introducing Project Cerberus

1. A set of **platform requirements**
   - E.g. Power sequencing while establishing trust

2. A set of **requirements** for ensuring **firmware integrity**
   - E.g. how to verify firmware integrity at boot
   - E.g. how to verify firmware signatures during updates

3. A **chip** that helps you implement the requirements
What is the Cerberus Chip?

• Dedicated security microprocessor
  • Internal Secure SRAM, Flash.
• Contains crypto acceleration blocks
  • SHA / AES / TRNG / PKA
• Interpose SPI/QSPI filter interface
• e-fuses for authentication public key hash and manifest revocation
• Hardware Physically Unclonable Function (PUF)
• Device Identifier Composition Engine (DICE)
• Tamper resistance
Interpose Interface

- All Firmware Authenticated
- All Firmware Measured
- All SPI Transactions Filtered
- All Region Access Controlled
- NIST 800-193 Enforced (Protect, Detect, Recover)
Protection

• Runtime: All flash accesses filtered through Cerberus

• Enforces region protection.

• Authenticated updates only

• Maintains Platform Firmware Manifest (PFM) for digital signature verification
Detection

- Pre-boot: Firmware integrity with digital signature enforcement
- Pre-boot attestable Firmware measurements, with freshness seed.
- Post-boot firmware integrity checks
- Post-boot attestable Firmware measurements with freshness seed.
Recovery

- Policy based Recovery
- Bare-metal recovery images locked in flash
- Flash access protected by Cerberus
- Automatic recovery workflow on detection of corruption
Platform Trust Hierarchy

- Scalable security architecture
- Motherboard contains master RoT.
- Each peripheral has a slave RoT capabilities interposed or native
- NIST 800-193 principles enforced at each level
- Master attests to platform firmware integrity
Open Compute Security Project

• Charter & doc links: https://www.opencompute.org/wiki/Security
  • In a nutshell: Creating open specs for ensuring firmware integrity of an OCP platform

• Current areas of focus (working on specs)
  • Scoping the security threats
  • SecureBoot
  • Attestation

• Haven’t started yet
  • Secure Updates
  • Recovery
  • Hardware interfaces

• We meet weekly. You’re welcome to join!
Key Takeaways

• Cerberus is Microsoft’s RoT for Hardware
  • Implementation & designs open sourced to OCP

• You’re welcome to join the OCP Security workgroups
  • We can use your help!

• Don’t forget that firmware integrity is not the whole story
  • Firmware quality & defense in depth are critical too
Cerberus Deconstruct

- Power on sequence
- Cerberus boot
- Cerberus attestation
- Platform boot
- Platform attestation
- Cerberus Provisioning
Power-On Sequence

• When power is applied, Cerberus powers first!
• Cerberus holds CPU and BMC in reset, they cannot access flash so holding them in reset is just a graceful thing to do.
• After Cerberus completes BMC flash digital signature verification, it allows flash access and releases reset.
• Cerberus communicates with peripherals for attestation measurements.
• After Cerberus completes BMC flash digital signature verification, it allows flash access and releases reset.
Cerberus Boot

• Immutable ROM authenticates key manifest (mutable code).
• ROM selects key from Manifest and authenticates RIoT Core layer.
• ROM calculates CDI of Manifest and RIoT Core
• ROM launches and passes control to RIoT Core, passing CDI.
• RIoT Core selects key from Manifest and authenticates Cerberus application firmware.
Cerberus Boot Continued (RIoT Core)

- ROM calculates CDI using Manifest and RIoT Core measurement
- ROM passes CDI to RIoT Core
- RIoT Core generates asymmetric Device ID key from CDI and Alias key from CDI and digest form FSD of Cerberus FW.
- Device Id Public key is provided to Cerberus Application Firmware
- Device Id Certificate and Alias Certificates are generated
- Alias Certificate is signed by Device Id key.
- If Device Id Certificate has not been signed, CSR is generated resulting in Device Id Certificate being CA signed during provisioning.
Cerberus - Platform Firmware Manifest (PFM)

• Describes firmware attributes of protected device:
  • RO flash descriptor offsets
  • RW flash descriptor offsets
  • Versions and version index
  • Digest of code blocks
  • Public Certificate for verification

• Cerberus uses PFM to verify flash content
• PFM describes supported firmware images.
• PFM is a signed update to Cerberus application firmware.
Cerberus - BMC Verification

- PFM describes FW digest structure
- PFM declares read-only, read-write regions
- Cerberus verifies digital signature of image on flash matches PFM
- Measurement extended for attestation.
- Cerberus SPI filter enforces runtime region protection
- Flatten Image Tree (FIT) permitted for secure boot extending.
- UEFI and other components protected in the same way.
Cerberus - BMC Golden Recovery

• Cerberus detects image integrity corruption
• If BMC restored from backup flash
• Always recoverable, Cerberus can internally stores BMC recovery uboot.
• Golden uboot performs NFS FIT secured Linux boot
Revocation

• Platform Firmware Manifest (PFM) are forward only.
  • PFM ID defined as uint64, does not permit rollback.
• PFM enforcement is function of the Cerberus firmware.
  • Soft revocation
• PFM is signed with key from key manifest.
• Key manifest can be hard revoked, using OTP memory and manifest ID
Cerberus Attestation Policies

- In-band attestation
  - SPI to Cerberus
  - KCS bridged through BMC to Cerberus

- Out Of Band (OOB) attestation
  - Ethernet bridged through BMC to Cerberus

- Policies can be set on Cerberus for remediation
  - Recovery
  - Power Control
Cerberus – Security Controller

Guided by NIST, enforces firmware integrity Protection, Detection and Recovery.

Cryptographic microcontroller enforces digital signatures on all platform firmware modules.

Hierarchical Root-of-Trust topology, provides attestation for all firmware modules

Opened the design

Security Project Committee in OCP
Open Compute Project

• Founded in April 2011 by Facebook
• Microsoft joined in January 2014
  • Shared collateral for Open Cloud Server
  • Optimized for cloud scale and density
  • Contributed design was fully complete 😞
• October 2016 Microsoft announced Project Olympus
  • Design was 70% complete
  • Attempting Open Source Server Hardware
• Community contributed to the design taking it to 100% complete – something never attempted before in OCP.
Project Olympus

- Community feedback incorporated into the solution
- System and Rack design
- Manufacturing collateral schematics and board files open sourced
- Open Firmware: Open EDKII, OpenBMC and Open PDU and Rack Manager
Open Source Momentum...
More Open Building Blocks Followed

Flash Storage

HDD Storage

GPU / PCIe