Using the TPM NVRAM to Protect Secure Boot Keys in OpenPOWER

Claudio Carvalho
cclaudio@br.ibm.com
IBM Linux Technology Center
Outline

• Introduction
  OpenPOWER Secure Boot Overview

• Problem Statement

• Protecting Secure Boot Keys in OpenPOWER
  Data stored in the TPM NV
  Authorization for the TPM NV data

• Final Considerations
OpenPOWER Secure Boot Team

IBM Linux Technology Center

IBM POWER Firmware  IBM LTC Security

IBM Research
Disclaimer

- This work represents the view of the author and does not necessarily represent the view of IBM.
- 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.
- IBM is a registered trademark of International Business Machines Corporation in the United States and/or other countries.
- Linux is a registered trademark of Linus Torvalds.
- Other company, product, and service names may be trademarks or service marks of others.
What is Secure Boot for?

Secure boot aims to prevent untrusted code from loading during the platform boot

Only code signed with trusted keys are started
OpenPOWER Secure Boot

- The OpenPOWER firmware is open-source
  - https://github.com/open-power/
  - op-build

- Domains:
  - Firmware Secure Boot
  - OS Secure Boot
POWER9 Boot Flow

*Source: https://github.com/open-power/docs/blob/master/hostboot/P9_Boot_Flow_OpenPOWER.pdf
Firmware Secure Boot

- Firmware images are signed following the secure boot container layout (sb-signing-tools)
- Root of trust: hardware keys hash
- Enabled by a hardware setting in the motherboard (platform dependent)

Very Simplified IPL Flow

Self Boot Engine

Hostboot

OPAL

Skirout

Host OS

Signed Firmware Images

Processor NOR (PNOR)

Verification Code

hw-key-hash

SEEPROM

OT PROM
This is the skiroot

Firmware Secure Boot is Upstream

Secure mode disabled
Secure boot will not be enforced

This is the skiroot
OS Secure Boot

- The OS Secure Boot work is in progress
- Skiroot is a linux kernel with embedded initramfs that runs Petitboot – a kexec bootloader

Current design:

- Host OS kernel:
  - It is signed with *sign-file*, the same tool used to sign kernel modules. The signature is appended
  - It is verified by IMA-appraisal
OS Secure Boot (cont'd)

Current design:
- Reuse the kernel code that supports EFI as much as possible:
  - efivars filesystem (/sys/firmware/efi/efivars/) Prototyped

Very Simplified IPL Flow

Self Boot Engine

Hostboot

OPAL

OPAL Runtime Services

efi.get_variable()
efi.get_next_variable()
efi.set_variable()
efi.query_variable_info()
OS Secure Boot (cont'd)

Current design:
- We are in the process to request distros to build the efivar package on powerpc64le
- Secure boot variables: X.509 certificates
  - Platform Key (PK)
    - Root of trust for the OS Secure Boot
    - When PK is set, OS Secure boot policy is enforced
  - Key Exchange Key (KEK)
  - Authorized Signature Database (db)
Problem Statement

- **Firmware Secure Boot keystore:**
  - hw-key-hash → SEEPROM

- **OS Secure Boot keystore:**
  - PK, KEK and db → PNOR SECBOOT partition (~128KB)

- **PNOR is unprotected by design, attackers could have their malicious code executed, for example.**

- **Trusted Platform Module (TPM) 2.0 provides protected non-volatile (NV) memory**

- **There is no space in the TPM2 NV for all secure boot variables**
Protecting the OS Secure Boot Keys

- Integrity
- TPM2 NV authorization
- Where each variable should be stored?
- Atomic variable update
**OS Secure Boot Keys: Integrity**

- Keys might be modified in the PNOR without notice
- **Detect** keys integrity issues using a SHA512 hash
- Keys are consumed only if valid

- PK
- KEK
- db

**PNOR SECBOOT**

- sha2_hash
- sha2_hash_size

**TPM2 NV**
TPM2 NV Authorization

- Access control required for the data stored in the TPM2 NV
- NV memory allocated is write locked at boot time until next boot
- Key updates are processed during the skiroot kernel boot

**Very Simplified IPL Flow**

1. Self Boot Engine
2. Hostboot
3. OPAL
4. Linux kernel
5. Petitboot (Kexec bootloader)
6. Host OS
7. Skiroot
8. PNOR SECBOOT

- PK
- KEK
- db
- sha2_hash
- sha2_hash_size
- TPM2 NV

**Update Queue**
Where Each Variable Should be Stored?

- If PK is lost, the root of trust is lost
- PK is stored in the TPM2 NV
- No special procedure required to recover KEK and db
Atomic Secure Boot Variable Update

- Writes to the storage might be interrupted
- ActiveBankSelector bit determines which is the current active bank
- Updates are persisted in the staging bank
- Flip the ActiveBankSelector bit and reboot

Very Simplified IPL Flow

Self Boot Engine
  └── Hostboot
      └── OPAL
          └── Linux kernel
              └── Petitboot (Kexec bootloader)
                  └── Host OS

ActiveBankSelector
  sha2_bank_hash_size

Bank[0]
  - KEK
  - db

Bank[1]
  - PK
  - sha2_hash

Update Queue
  PNOR SECBOOT

TPM2 NV
OS Secure Boot NV Indices

Define the OS NV indices

```
[root@localhost utils]$ ./nvdefinespace -ha 01c10191 -hi p -hia p
  -sz 6  -at ppr +at ar +at wst -pwdn ""
  nvdefinespace: success
[root@localhost utils]$ ./nvdefinespace -ha 01c10192 -hi p -hia p
  -sz 1088 -at ppr +at ar +at wst -pwdn ""
  nvdefinespace: success
[root@localhost utils]$ ./nvdefinespace -ha 01c10193 -hi p -hia p
  -sz 1088 -at ppr +at ar +at wst -pwdn ""
  nvdefinespace: success
```

Read the os-nv-header index public info

```
[root@localhost utils]$ ./nvreadpublic -ha 01c10191
  nvreadpublic: name algorithm 000b
  nvreadpublic: data size 6
  nvreadpublic: attributes 42044005
  TPMA_NV_PPWRITE
  TPMA_NV_AUTHWRITE
  TPMA_NV_WRITE_STCLEAR
  TPMA_NV_AUTHREAD
  TPMA_NV_NO_DA
  TPMA_NV_PLATFORMCREATE
  nvreadpublic: policy length 0

  nvreadpublic: name length 34
  00 0b 27 35 82 6b 0f 3e f1 de 4c 00 b2 f1 c6 41
  2b 68 95 b4 1a 1c f4 aa f4 7d e9 3c 5c ec 16 f8
  81 67
[root@localhost utils]$  
```

- IBM’s TPM 2.0 TSS* is open-source
- Max NV Index size = 2048 bytes
- Same attributes, but different sizes
- Write-locked at boot time until next boot

```
<table>
<thead>
<tr>
<th>ActiveBankSelector</th>
<th>(2 bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>sha2_bank_hash_size</td>
<td>(4 bytes)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bank[0]</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>- PK</td>
<td>(1024 bytes)</td>
</tr>
<tr>
<td>- sha2_bank_hash</td>
<td>(64 bytes)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bank[1]</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>- PK</td>
<td>(1024 bytes)</td>
</tr>
<tr>
<td>- sha2_bank_hash</td>
<td>(64 bytes)</td>
</tr>
</tbody>
</table>

NV data for OS Secure Boot
Total size = ~2182 bytes

* TCG Software Stack (TSS)
Firmware Secure Boot NV Index

Define the Firmware NV index

```
[root@localhost utils]$ ./nvdefinespace -ha 01c10190 -hi p 'hia p \
> -sz 64 -at ppr +at ar +at wst -pwdn ""
nvdefinespace: success
[root@localhost utils]$ 
```

Read the os-nv-header index public info

```
[root@localhost utils]$ ./nvreadpublic -ha 01c10190
nvreadpublic: name algorithm 000b
nvreadpublic: data size 64
nvreadpublic: attributes 42044005
TPMA_NV_PPWRITE
TPMA_NV_AUTHWRITE
TPM_NT_ORDINARY
TPM_NV_WRITE_STCLEAR
TPM_NV_AUTHREAD
TPM_NV_NO_DA
TPMA_NV_PLATFORMCREATE
nvreadpublic: policy length 0

nvreadpublic: name length 34
00 0b 59 bc 8f a6 03 9d c8 66 0a 27 68 90 ab 43
95 73 5c 29 a7 f3 2d 03 c1 c2 10 17 6c 7e bf 9f
ee d8
[root@localhost utils]$ 
```

- Hardware Key Hash* (64 bytes)

NV data for Firmware Secure Boot

Total size = 64 bytes

* The OS platform key is invalidated when the underlying hardware keys change
Other TPM2 NV Commands

Read and write to the NV index

```
[root@localhost utils]$ ./nvwrite -ha 01c10192 -hia p -pwdn "" -ic "LinuxSecuritySummit"
[root@localhost utils]$ ./nvread -ha 01c10192 -pwdn "" -sz 30 -of lss.txt
```

Write lock the NV Index until the next TPM Reset or TPM Restart

```
[root@localhost utils]$ ./nvwritelock -ha 01c10192 -hia p -pwdn "" 
[root@localhost utils]$ ./nvwrite -ha 01c10192 -hia p -pwdn "" -ic "foobar"
```

Undefine the NV Index

```
[root@localhost utils]$ ./nvundefinespace -ha 01c10192 -hi p
[root@localhost utils]$ ./nvreadpublic -ha 01c10192
```

Set the platform authorization default password to “pass4lss”

```
[root@localhost utils]$ ./hierarchychangeauth -hi p -pwda "" -pwdn "pass4lss"
```
OS Secure Boot Architecture

**PNOR SECBOOT**

- **Bank[0]**
  - KEK (~800KB each entry)
  - db (~800KB each entry)
- **Bank[1]**
  - KEK (~800KB each entry)
  - db (~800KB each entry)
- **UpdateQueue**
  (~2KB each update)

**TPM2 NV**

- **Bank[0]**
  - PK (1024 bytes)
  - sha2_bank_hash (64 bytes)
- **Bank[1]**
  - PK (1024 bytes)
  - sha2_bank_hash (64 bytes)

**ActiveBankSelector** (bit)

**sha2_bank_hash_size** (4 bytes)

**Very Simplified IPL Flow**

1. **Self Boot Engine**
2. **Hostboot**
3. **OPAL**
4. **OPAL Runtime Services**
   - efi.get_variable()
   - efi.get_next_variable()
   - efi.set_variable()
   - efi.query_variable_info()

**Linux kernel**

**Petitboot** (Kexec bootloader)

**Userspace**
Final Considerations

- TPM2 NV has shown a secure and valuable storage to protect secure boot variables
- In POWER9, OpenPOWER OS Secure Boot depends on TPM 2.0
- Sharing TSS code throughout the firmware stack is challenging
- Verbose mode in the IBM's TSS
References

OpenPOWER Foundation
https://openpowerfoundation.org

OpenPOWER Firmware
https://github.com/open-power

POWER9 Boot Flow
https://github.com/open-power/docs/blob/master/hostboot/P9_Boot_Flow_OpenPOWER.pdf

Protecting System Firmware with OpenPOWER Secure Boot

Trusted Platform Module TCG Working Group
https://trustedcomputinggroup.org/work-groups/trusted-platform-module/

IBM's TPM 2.0 TSS
https://sourceforge.net/projects/ibmtpm20tss/
Questions?

Thank you! Obrigado!

Claudio Carvalho
cclaudio@br.ibm.com
IBM Linux Technology Center
Backup Slides
Creating, Using and Installing Your Own Keys

Create at least three sets of certificates: one for PK, one for KEK and one for db

```bash
>$ openssl req -new -x509 -newkey rsa:2048 -subj "/CN=DB/" \
-keyout db.key -out db.crt -days 3650 -nodes -sha256
```

Sign the UEFI images with your db key

```bash
>$ sbsign --key db.key --cert db.crt --output \ HeloWorld-signed.efi HeloWorld.efi
```

Create authorized variable updates. Repeat for KEK and PK

```bash
>$ cert-to-sig-list db.crt db.esl
>$ sign-efi-sig-list -k KEK.key -c KEK.crt db db.esl db.auth
```

Update the variables on your platform, remembering to do PK last.

```bash
>$ sudo efivar -n 8be4df61-93ca-11d2-aa0d-00e098032b8c-PK -w -f PK.auth
>$ sudo efivar -n 8be4df61-93ca-11d2-aa0d-00e098032b8c-KEK -w -f KEK.auth
>$ sudo efivar -n d719b2cb-3d3a-4596-a3bc-dad00e67656f-db -w -f DB.auth
```

OR

```bash
>$ efi-updatevar -f db.auth db
>$ efi-updatevar -f KEK.auth KEK
>$ efi-updatevar -f PK.auth PK
```

* Source: https://git.kernel.org/pub/scm/linux/kernel/git/jejb/efitools.git/tree/README
Detecting if the NV index wasn't written yet (TPM 2.0)

```
[root@localhost utils]$ ./nvread -ha 01c10190 -pwn "" -sz 30
nvread: failed, rc 00000014a
TPM_RC_NV_UNINITIALIZED - an NV Index is used before being initialized
[root@localhost utils]$ ./nvwrite -ha 01c10190 -hia p -pwn "pass4lss" -ic "LinuxSecuritySummit"
[root@localhost utils]$ ./nvread -ha 01c10190 -pwn "" -sz 30 -of lss.txt
nvread: data length 30
4c 69 6e 75 78 53 65 63 75 72 69 74 79 53 75 76 6d
6d 69 74 00 00 00 00 00 00 00 00 00 00 00
[root@localhost utils]$ hexdump -C lss.txt
00000000  4c 69 6e 75 78 53 65 63 75 72 69 74 79 53 75 76 6d |LinuxSecuritySummit|
00000010  6d 69 74 00 00 00 00 00 00 00 00 00 00 00 00 00
0000001e
[root@localhost utils]$./nvreadpublic -ha 01c10190
nvreadpublic: name algorithm 000b
nvreadpublic: data size 1024
nvreadpublic: attributes 62054001
TPMA_NV_PPWRITE
TPM_NT_ORDINARY
TPMA_NV_WRITE_STCLEAR
TPMA_NV_PPREAD
TPMA_NV.AUTHREAD
TPMA_NV_NO_DA
TPMA_NV_WRITEN
TPMA_NV_PLATFORMCREATE
nvreadpublic: policy length 0
```

```
vreadpublic: name length 34
00 0b 38 fa 00 5d e0 7d 8b c3 80 a1 74 9e ae 3f
4a 50 c9 20 35 61 56 87 24 f9 90 be 80 95 ad fb
45 87
[root@localhost utils]$ ```
Verbose mode (-v) can be used to inspect TSS commands, specially the byte stream sent and received from the TPM2.