Applications of TPM 2.0

Doing useful things with a TPM

Russell Doty
Javier Martinez Canillas
Last Year

- Introduced TPM 2.0
- Showed Clevis with TPM in Fedora
- Talked about a great future

Hardware Root of Trust

Trust me, it’s real this time...

Russell Doty
Javier Martinez Canillas
This Year

✓ NBDE with TPM2 shipped in RHEL 7.6
  ○ Included in next version of RHEL
✓ TPM2 measurement of boot path added to grub2 & shim
✓ Customers using TPM2 and Sealing to protect secrets
✓ Code to encrypt/decrypt data with TPM2 available
✓ Signing with TPM2
✓ And more in the pipeline!
HW RoT Strategy Update

- All Red Hat TPM activities built on TPM2
  - TPM 1.2 is available but deprecated
- There are two TPM2 userspace libraries included in Linux:
  - TCG (Trusted Computing Group)
  - IBM
  - Red Hat development efforts based on the TCG specification.
- The HW RoT focus at Red Hat is TPM2 and TCG userspace
TPM2 and PKCS #11

- PKCS #11: platform independent interface to security tokens
  - TPM2, Yubikey, SmartCard, HSM, etc.
- Create, modify & delete keys and tokens
  - RSA, X.509, etc.
- Widely used by certificate authorities
- Proposed interface to TPM2
  - Subset of TPM2 capabilities
  - Implemented on top of TCG interface (TPM2 Software Stack)
PKCS #11 in GnuTLS
TPM2 Cheat Sheet:

- Clevis is a crypto framework with many capabilities
- Many useful TPM operations have been implemented in Clevis
- The Clevis code is a good starting point for developing TPM2 applications
  - As well as Jose, JWE, Shamir Secret Sharing, UDisks2
- https://github.com/latchset/clevis
TPM2 building blocks

<table>
<thead>
<tr>
<th>Crypto processing</th>
<th>Volatile memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNG</td>
<td>Transient objects</td>
</tr>
<tr>
<td>Key generator</td>
<td>PCR banks</td>
</tr>
<tr>
<td>Hash generator</td>
<td>Non-volatile memory</td>
</tr>
<tr>
<td>Encryption engine</td>
<td>Persistent objects</td>
</tr>
<tr>
<td>Decryption engine</td>
<td>Hierarchy seeds</td>
</tr>
</tbody>
</table>

Secured I/O
TPM2 features

- Random number generation
- Secure key generation and storage
- Encryption and decryption
- Machine identification
- NVRAM storage
- Integrity measurement
- System health attestation
TPM2 Implementations

- Hardware chips

- Firmware TPM (Intel & AMD)

- **NEW!** Software TPM in Virtualization (KVM/QEMU)
Secrets and TPM2

- TPM can encrypt/decrypt arbitrary secrets
  - Secret: a blob of data
  - Stored encrypted outside of TPM
    - “Unlimited” storage space
  - Operations such as *signing* entirely inside TPM
  - Sealing of Data: preconditions must be met before data is operated on
  - Note: TPM is too slow for bulk crypto; use for keys and other secrets
Secrets and TPM2

$ tpm2_createprimary -H o -g sha256 -G ecc -C primary.context

$ tpm2_create -g sha256 -G rsa -u key.pub -r key.priv -c primary.context

$ tpm2_load -c primary.context -u key.pub -r key.priv -n key.name -C key.context

$ echo foobar > plaintext.txt

$ tpm2_rsaencrypt -c key.context -o secret.dat plaintext.txt

$ rm plaintext.txt

$ tpm2_rsadecrypt -c key.context -I secret.dat -o plaintext.txt

$ cat plaintext.txt

foobar
Clevis

- Pluggable framework for automated decryption
- Contains a number of “pins” that implements decryption policies
- CLI to decrypt data using these pins
- Allow to bind a pin to a LUKS volume
- Have different LUKS unlockers:
  - Dracut
  - Udisk2
  - clevis luks unlock
Clevis and TPM2

Encryption
- Pin
- Config
- Data

Clevis encrypt
- JWK
- JWE

TPM 2.0

Decryption
- TPM 2.0

JWE

JWK

Clevis decrypt

Data
Clevis and TPM2

$ echo foobar | clevis encrypt tpm2 '{},' > secret.jwe

$ clevis decrypt < secret.jwe
foobar
Clevis and LUKS

$ dnf install clevis-dracut

$ dracut -f

$ UUID=$(lsblk | grep luks | sed 's/^.*luks-//' | cut -d ' ' -f1)

$ DEV=$(blkid --uuid $UUID)

$ clevis luks bind -d $DEV tpm2 '{}'
TPM Measurement

- Items of interest, like UEFI, are hashed and hash is stored in TPM PCR register
- Additional values are hashed and extend the TPM PCR - all work done inside TPM
  - Single PCR contains results of a series of measurements
- Final PCR value measures the state of system
- The final measurement depends on each item measured and order of measurements.
- TPM measurements into PCR difficult to spoof and can’t be modified.
# Platform Configuration Registers

<table>
<thead>
<tr>
<th>PCR #</th>
<th>Allocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>BIOS</td>
</tr>
<tr>
<td>1</td>
<td>BIOS Configuration</td>
</tr>
<tr>
<td>2</td>
<td>Option ROMs</td>
</tr>
<tr>
<td>3</td>
<td>Option ROM Configuration</td>
</tr>
<tr>
<td>4</td>
<td>MBR (Master Boot Record)</td>
</tr>
<tr>
<td>5</td>
<td>MBR Configuration</td>
</tr>
<tr>
<td>6</td>
<td>State Transitions and Wake Events</td>
</tr>
<tr>
<td>7</td>
<td>Platform Manufacturer specific measurements</td>
</tr>
<tr>
<td>8-15</td>
<td>Static Operating System</td>
</tr>
<tr>
<td>16</td>
<td>Debug</td>
</tr>
<tr>
<td>23</td>
<td>Application Support</td>
</tr>
</tbody>
</table>
TPM: grub & shim

• Following PCRs are extended by shim:
  o PCR 4: Authenticode hash of binaries
  o PCR 7: Certificates
  o PCR 8: Kernel & grub commands
  o PCR 9: kernel, initramfs, & multiboot modules
  o PCR 14: MOKList, MOKListX, and MOKSBstate
  o Note: PCR 0-3 extended by HW & TPM

• See https://github.com/rhboot/shim for details
TPM Sealing and Secrets

- **UnSealing**: a PCR check is done before performing operations
- **Sealed Secrets** can only be decrypted with valid PCR measurements
  - Sealing operation uses specified PCRs
  - TPM checks current PCR values
- **Sealing operations** occur inside TPM
TPM2 Sealing and Secrets

$ echo foobar | clevis encrypt tpm2 \{"pcr_ids":"7"\} > tpm2.jwe

$ clevis decrypt < tpm2.jwe
foobar
Attestation

- Compare TPM PCRs to known good values
- **Static Attestation**: system contains list of valid values.
  - e.g. PCR values for vendor UEFI or RHEL bootloader
- **Dynamic Attestation**: system sends PCR values to an external service to determine validity
  - Examples: KeyLime, Intel OpenCIT, StrongSwan
Signing with TPM2

• Hash the data to be signed.
• Hash is encrypted using TPM private key.
• Hash is decrypted using TPM public key.
• Can prove that data is unmodified and that it was processed by the TPM.
Signing with TPM2

$ tpm2_createprimary -H o -g sha256 -G ecc -C primary.context

$ tpm2_create -g sha256 -G rsa -u key.pub -r key.priv -c primary.context

$ tpm2_load -c primary.context -u key.pub -r key.priv -n key.name -C key.context

$ echo foobar > data

$ tpm2_sign -c key.context -g sha256 -m data -s signature

$ tpm2_loadexternal -H o -u key.pub -C external_key.context

$ tpm2_verifysignature -c external_key.context -t ticket -g sha256 -m data -s signature
Interesting Upstream Projects

● TPM2 and LUKS (clevis)
● TPM2 and OpenSSL (tpm2-tss-engine)
● TPM2 and PCKS#11 (tpm2-pkcs11)
Example use cases & demos

● Automatically unlock LUKS using Udisk2
● Protect a server certificate private key
● Protect a client OpenSSH private key
Udisks2 to unlock LUKS dev

$ clevis luks bind -d /dev/sdb1 tpm2 '{}'
Protect a LUKS master key

Sealing against system and peripheral FW, EFI drivers and Secure Boot policy

$ clevis luks bind -d $DEV tpm2 '{"pcr_ids":"0,2,7"}'}
Protect a cert private key

$ tpm2tss-genkey -a rsa server.tss

$ openssl req -new -key server.tss -engine tpm2tss -keyform engine -out server.csr

$ openssl x509 -req -in server.csr -CA ca-root.pem -CAkey ca-root.key -CAcreateserial -out server-cert.pem -days 365

$ openssl s_server -cert server-cert.pem -key server.tss -keyform engine -engine tpm2tss -CAfile ca-root.pem -accept 8443
Protect a ssh private key

$ ssh-keygen -D /usr/lib64/pkcs11/libtpm2_pkcs11.so > tpm2_key.pub

$ ssh-copy-id -f -i tpm2_key user@host

$ ssh -l /usr/lib64/pkcs11/libtpm2_pkcs11.so user@host
Potential uses of TPM

- Secured keyring
- Secured certificates
- Secured SSH
- Secured VPN
- Trusted Identity
- Git signing
- Secure Host/Secure VM
- Secure Container
- Software Licensing & authorization
- Multi-factor Authentication (including “m of n”)
- IoT
- Blockchain
Resources

- A Practical Guide to TPM 2.0
- Trusted Computing Group
  - [https://trustedcomputinggroup.org](https://trustedcomputinggroup.org)
- IBM TSS
  - [https://sourceforge.net/projects/ibmtpm20tss](https://sourceforge.net/projects/ibmtpm20tss)
- TCG TSS
  - [https://github.com/tpm2-software](https://github.com/tpm2-software)
- Clevis
  - [https://github.com/latchset/clevis](https://github.com/latchset/clevis)
- OpenSSL engine
  - [https://github.com/tpm2-software/tpm2-tss-engine](https://github.com/tpm2-software/tpm2-tss-engine)
- PKCS#11 support
  - [https://github.com/tpm2-software/tpm2-pkcs11](https://github.com/tpm2-software/tpm2-pkcs11)
Thank-you!