Reference Integrity Measurements & Grub2

Philip Tricca, Intel
philip.b.tricca@intel.com
Background

• Assume audience understands TPMs, what they’re for & attestation process
• mjg’s attestation talk on Monday is great background
• Terminology in this space is unfortunate & inconsistent
  • Measurement – synonymous with “hash” or “digest”
  • Attester – entity provide evidence to verifier for appraisal
  • Verifier – entity requesting attestations from clients
  • Verification – process by which integrity of attestation evidence is established
  • Appraisal – process by which verifier establishes trust in attestation evidence
• Acronyms
  • RIM – reference integrity measurement
  • RIMM – RIM Manifest
Verification of Attestation Evidence

- Verifier receives evidence
  - Signed manifest of PCR values (“quote”)
  - Event log

- Verifier must first verify
  - Signature over quote: trusts signing key
  - Integrity of evidence: reconstruct PCR values from event log

- Verification
  - Necessary precondition to appraisal
  - Alone it doesn’t provide much
Appraisal of Attestation Evidence

- Platform identity: hardware, software & configuration
- Events from event log tell us about software & config
- Appraisal process results in a trust decision
  - Can events from the event log be identified?
  - Are these values what you expect / something known (good or bad)
- This assumes
  - Verifier has sufficient *knowledge* to identify components
Entities & Relationships

- SW Build
  - Source Code
  - Build Artifacts
    - BINs
    - CFGs

- Attester
  - Measurement
  - Reporting

- Verifier
  - Verification & Appraisal
  - Access Control
Reference Integrity Measurement (RIM)

- Measurement == Hash
- RIM include a hash over a piece of software or config + metadata
- RIM Manifest (RIMM) is a collection of RIMs
- RIMM is input into the appraisal process
- Collection of RIMMs is effectively the appraisal corpus
Updated Entities & Relationships
POC w/ Grub2

- Goal: generate data for RIMM
  - Calculate identity (hash) of Grub2 components
    - Independent of event log
    - Method must integrate into build
  - Associate events from event log with Grub2 components
  - Motivate future work, frame discussions with distros

- Required work
  - Borrow [test script from tpm2-tcti-uefi](https://github.com/flihp/grub2/tree/tpmcmd) (swtpm + qemu + ovmf)
  - Port [example from tpm2-tcti-uefi](https://github.com/flihp/grub2/tree/tpmcmd) to dump event log from grub2 shell
    - [https://github.com/flihp/grub2/tree/tpmcmd](https://github.com/flihp/grub2/tree/tpmcmd) - not a permalink
  - Tools to hash Grub2 components
Firmware happens
- Firmware
  - Measures bootloader (grub2)
  - Records event in event log
  - Executes grub2
- Grub2
  - Measures normal module
  - Records event in Eventlog
  - Loads normal module
- Load module dependencies
  - Measure module
  - Record event in Eventlog
  - Load module
  - Repeat
- grub.cfg processing
  - Eventually execute kernel
Calculate has of grub.efi / bootx64.efi

• Digest from Eventlog

```
AlgorithmId: EFI_TCG2_BOOT_HASH_ALG_SHA256 (0xb)
Digest: 32 bytes
00000006 a9 49 e9 c9 88 a7 83 75 3f 22 be 23 e1 ef 66 08 |...I.....u?"#.##f.|
00000016 dc 90 37 a6 ed bf 04 67 1e d8 7e 18 5c 53 0e 95 |...7....g..\S..|
```

• pehasher – just sbsign with PKCS#7 bits hacked off

```
$ src/pehasher /tmp/test-img/EFI/BOOT/BOOTX64.EFI
a949e9c988a783753f22be23e1ef6608dc9037a6edbf04671ed87e185c530e95
```
Calculate has of grub module

• Digest from Eventlog

```
AlgorithmId: EFI_TCG2_BOOTHASH_ALG_SHA256 (0xb)
Digest: 32 bytes
00000006  87 5e b1 0c 0a 84 41 ad c6 47 de b9 69 fa 56 19 |.^....A...G..i.V.|
00000016  17 14 09 d9 d4 b7 36 52 0d eb fc b7 53 73 24 3c |......6R....Ss$<|
```

• Simple sha256 hash

```
$ sha256sum ./grub-core/tpmcmd.mod
875eb10c0a8441adc647deb969fa5619171409d9d4b736520debfcb75373243c ./grub-core/tpmcmd.mod
```
POC output

• Ability to calculate hash of grub executable & modules @ build time
  • Not always as simple as sha*sum, hopefully UEFI PEs are a “worst case”
  • Cannibalizing sbsign isn’t sustainable
  • Ignores hard problems like grub.cfg

• More questions than answers
• How deep can the appraisal process go?
  • Identifying binary is good, but trust is implicit: limit of closed source
  • Need process to trace RIM back to source (reproducible builds)

• Tools needed
  • Need a tools to generate RIMs in standard format
  • Must integrate with build process

• Need infrastructure for distributing RIMs to verifiers
Thank You!