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

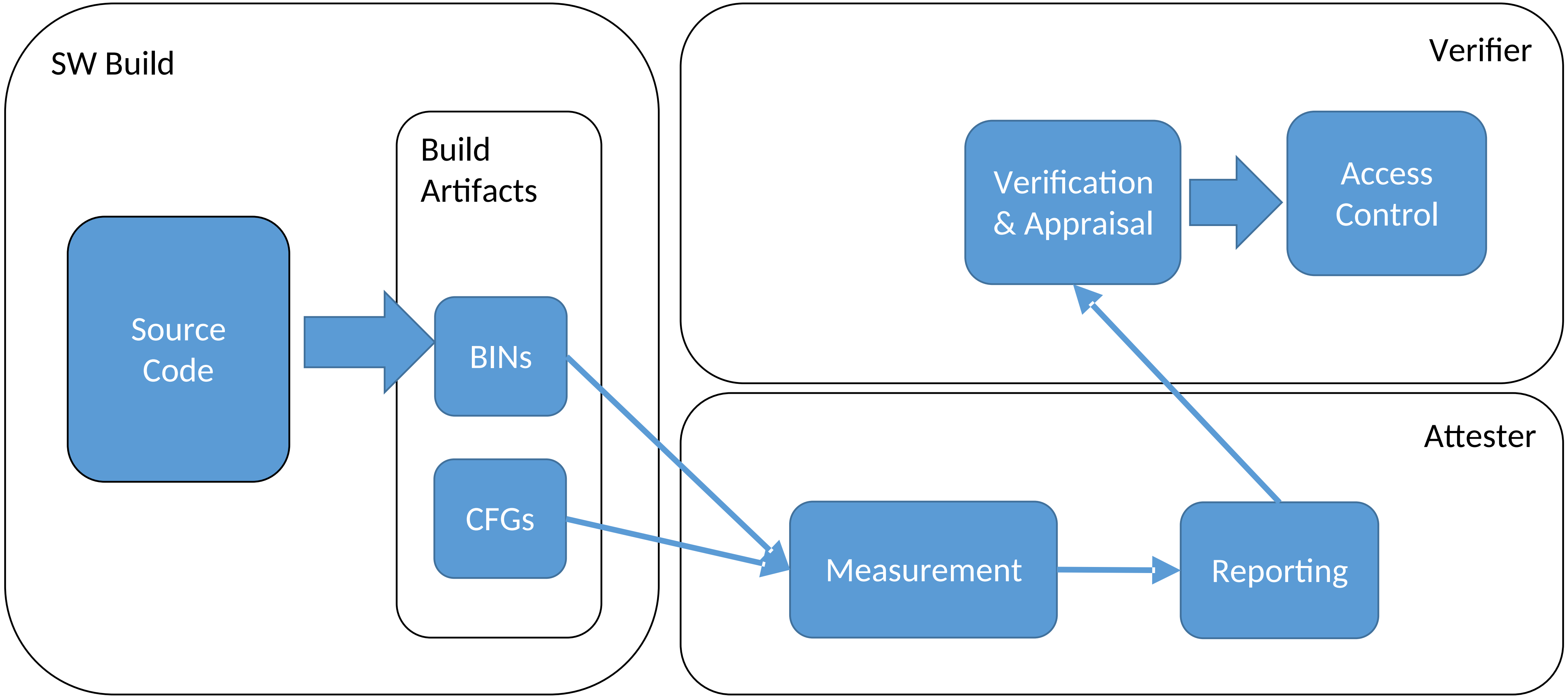
Verification of Attestation Evidence

- 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

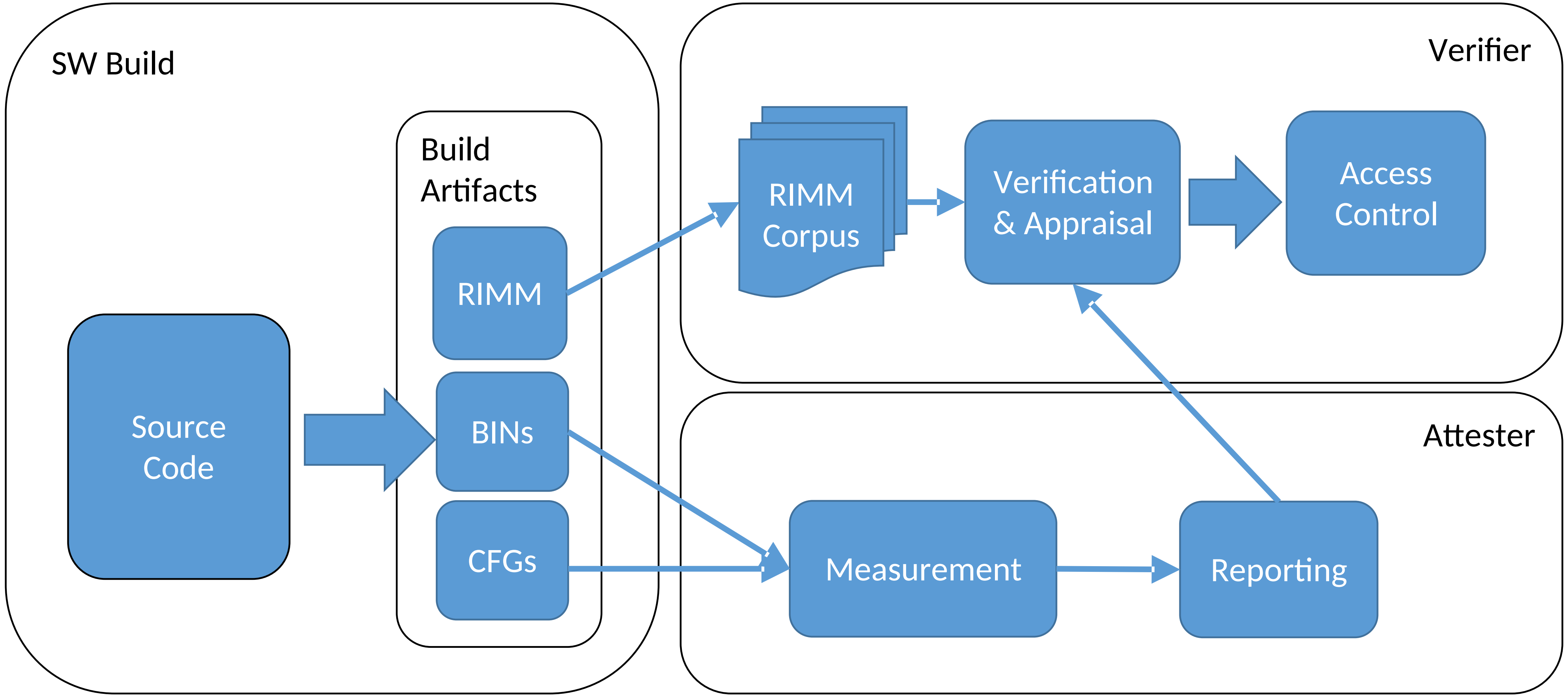
Entities & Relationships



Reference Integrity Measurement (RIM)

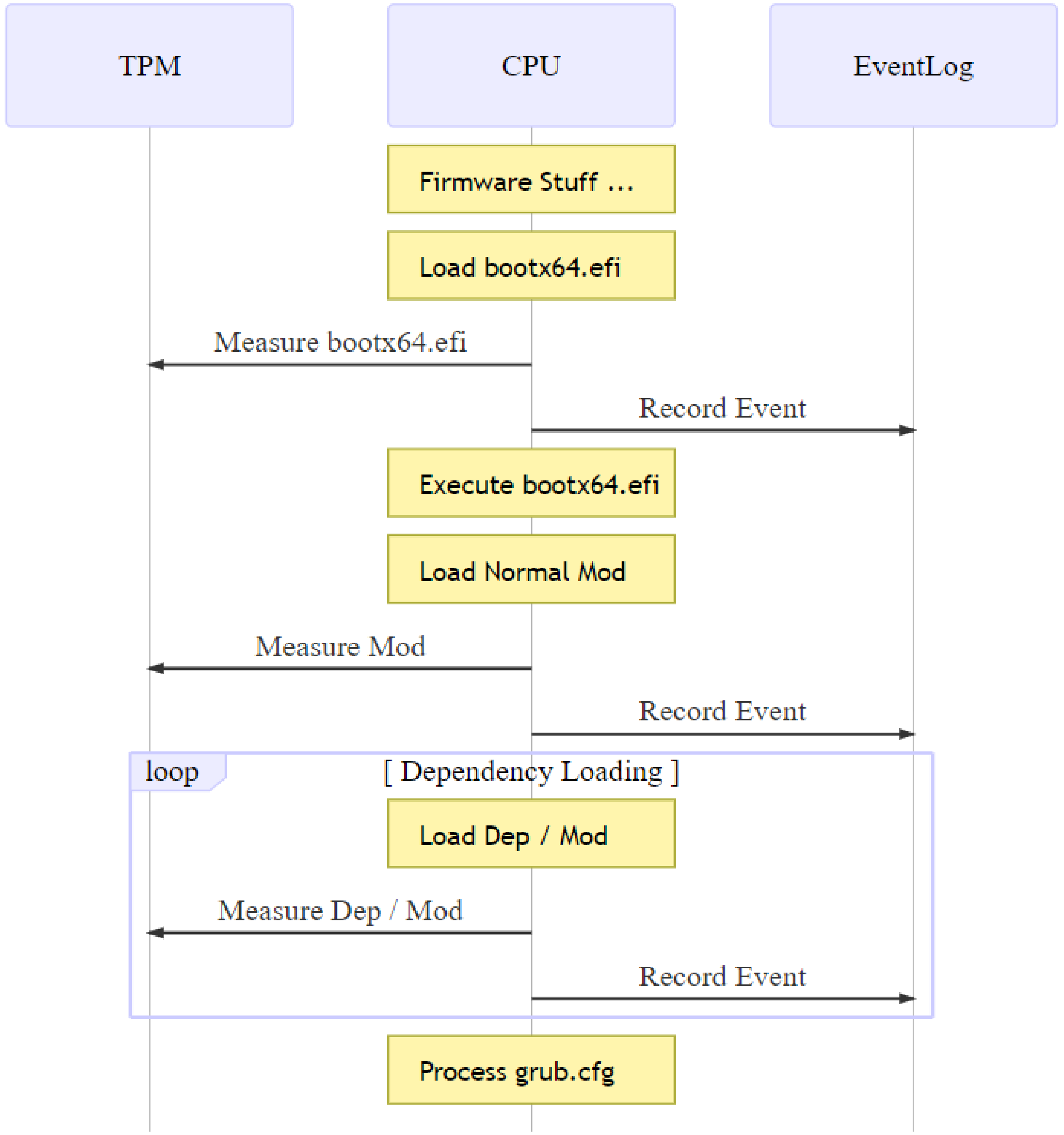
- Measurement == Hash
- RIM include a hash over a piece of software or config + metadata
- RIM Manifest (RIMM) is a collection of RIMs
- RIMM is input into the appraisal process
- Collection of RIMMs is effectively the appraisal corpus

Updated Entities & Relationships

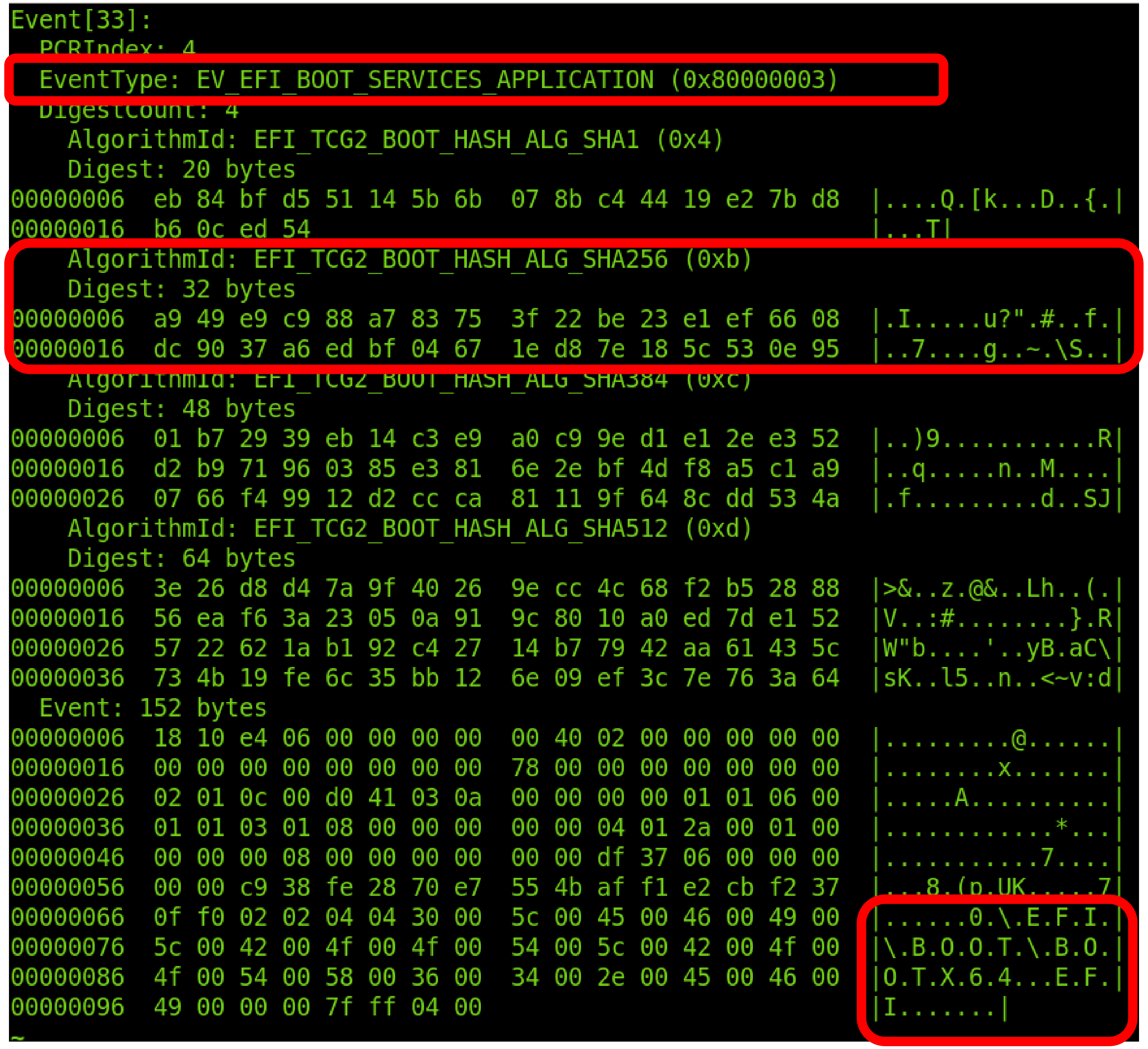


POC w/ Grub2

- Goal: generate data for RIMM
  - Calculate identity (hash) of Grub2 components
    - Indepnedent 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 (swtpm + qemu + ovmf)
  - Port example from tpm2-tcti-uefi to dump event log from grub2 shell
    - https://github.com/flihp/grub2/tree/tpmcmd - not a permalink
  - Tools to hash Grub2 components



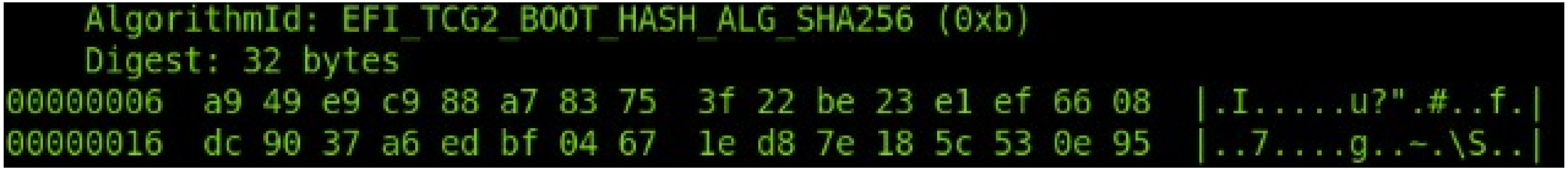
- Firmware happens
- Firmware
  - Measures bootloader (grub2)
  - Records event in event log
  - Executes grub2
- Grub2
  - Measures normal module
  - Records event in Eventlog
  - Loads normal module
- Load module dependencies
  - Measure module
  - Record event in Eventlog
  - Load module
  - Repeat
- grub.cfg processing
  - Eventually execute kernel



```
Event[33]:
  PCRIndex: 4
  EventType: EV_EFI_BOOT_SERVICES_APPLICATION (0x80000003)
  DigestCount: 4
    AlgorithmId: EFI_TCG2_BOOT_HASH_ALG_SHA1 (0x4)
    Digest: 20 bytes
00000006  eb 84 bf d5 51 14 5b 6b  07 8b c4 44 19 e2 7b d8  |....Q.[k...D..{.|
00000016  b6 0c ed 54                                       |...T|
    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..|
    AlgorithmId: EFI_TCG2_BOOT_HASH_ALG_SHA384 (0xc)
    Digest: 48 bytes
00000006  01 b7 29 39 eb 14 c3 e9  a0 c9 9e d1 e1 2e e3 52  |..)9...........R|
00000016  d2 b9 71 96 03 85 e3 81  6e 2e bf 4d f8 a5 c1 a9  |..q.....n..M....|
00000026  07 66 f4 99 12 d2 cc ca  81 11 9f 64 8c dd 53 4a  |.f.........d..SJ|
    AlgorithmId: EFI_TCG2_BOOT_HASH_ALG_SHA512 (0xd)
    Digest: 64 bytes
00000006  3e 26 d8 d4 7a 9f 40 26  9e cc 4c 68 f2 b5 28 88  |>&..z.@&..Lh..(.|
00000016  56 ea f6 3a 23 05 0a 91  9c 80 10 a0 ed 7d e1 52  |V..:#........}.R|
00000026  57 22 62 1a b1 92 c4 27  14 b7 79 42 aa 61 43 5c  |W"b....'..yB.aC\|
00000036  73 4b 19 fe 6c 35 bb 12  6e 09 ef 3c 7e 76 3a 64  |sK..l5..n..<~v:d|
  Event: 152 bytes
00000006  18 10 e4 06 00 00 00 00  00 40 02 00 00 00 00 00  |.........@......|
00000016  00 00 00 00 00 00 00 00  78 00 00 00 00 00 00 00  |........x.......|
00000026  02 01 0c 00 d0 41 03 0a  00 00 00 00 01 01 06 00  |.....A..........|
00000036  01 01 03 01 08 00 00 00  00 00 04 01 2a 00 01 00  |............*...|
00000046  00 00 00 08 00 00 00 00  00 00 df 37 06 00 00 00  |...........7....|
00000056  00 00 c9 38 fe 28 70 e7  55 4b af f1 e2 cb f2 37  |...8.(p.UK.....7|
00000066  0f f0 02 02 04 04 30 00  5c 00 45 00 46 00 49 00  |......0.\.E.F.I.|
00000076  5c 00 42 00 4f 00 4f 00  54 00 5c 00 42 00 4f 00  |\.B.O.O.T.\.B.O.|
00000086  4f 00 54 00 58 00 36 00  34 00 2e 00 45 00 46 00  |O.T.X.6.4...E.F.|
00000096  49 00 00 00 7f ff 04 00                           |I.......|
```

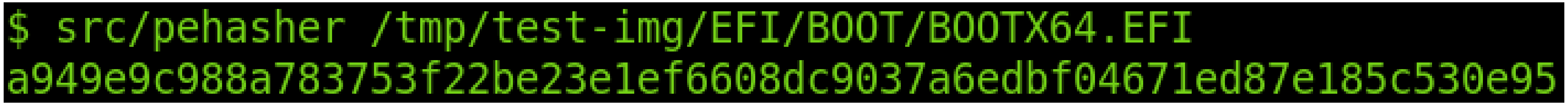
Calculate has of grub.efi / bootx64.efi

- Digest from Eventlog

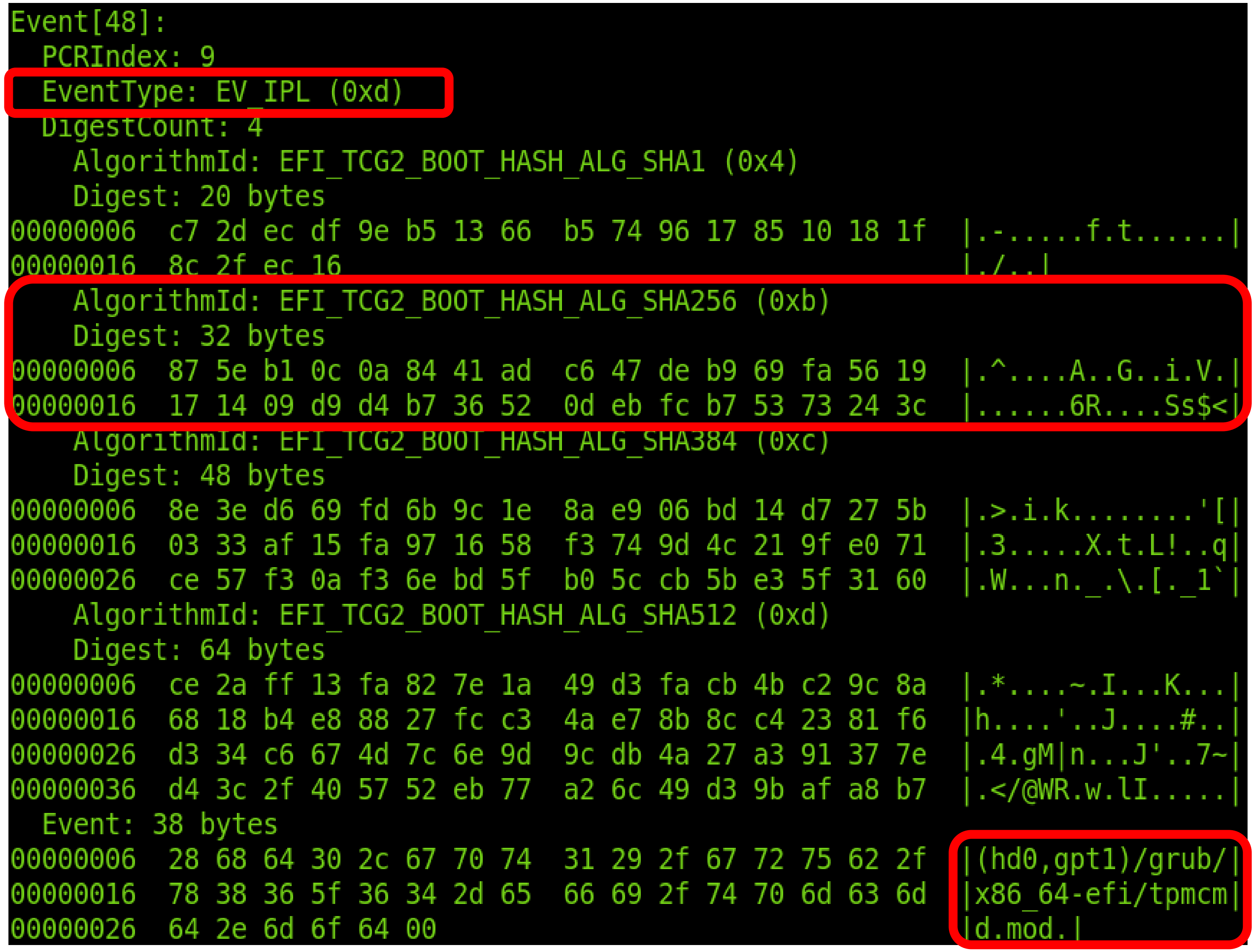


```
    AlgorithmId: EFI_TCG2_BOOT_HASH_ALG_SHA256 (0xb)
    Digest: 32 bytes
00000006  a9 49 e9 c9 88 a7 83 75  3f 22 be 23 e1 ef 66 08  |.I.....u?".#..f.|
00000016  dc 90 37 a6 ed bf 04 67  1e d8 7e 18 5c 53 0e 95  |..7....g..~.\S..|
```

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



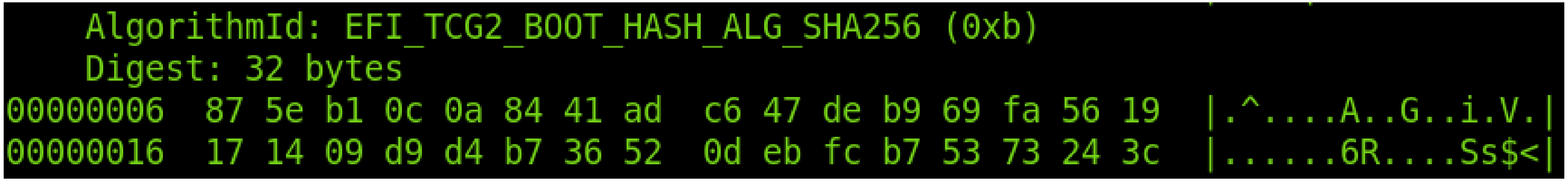
```
$ src/pehasher /tmp/test-img/EFI/BOOT/BOOTX64.EFI
a949e9c988a783753f22be23e1ef6608dc9037a6edbf04671ed87e185c530e95
```



```
Event[48]:
  PCRIndex: 9
  EventType: EV_IPL (0xd)
  DigestCount: 4
    AlgorithmId: EFI_TCG2_BOOT_HASH_ALG_SHA1 (0x4)
    Digest: 20 bytes
00000006  c7 2d ec df 9e b5 13 66  b5 74 96 17 85 10 18 1f  |.-.....f.t......|
00000016  8c 2f ec 16                                      |./..|
    AlgorithmId: EFI_TCG2_BOOT_HASH_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$<|
    AlgorithmId: EFI_TCG2_BOOT_HASH_ALG_SHA384 (0xc)
    Digest: 48 bytes
00000006  8e 3e d6 69 fd 6b 9c 1e  8a e9 06 bd 14 d7 27 5b  |.>.i.k........'[|
00000016  03 33 af 15 fa 97 16 58  f3 74 9d 4c 21 9f e0 71  |.3.....X.t.L!..q|
00000026  ce 57 f3 0a f3 6e bd 5f  b0 5c cb 5b e3 5f 31 60  |.W...n._.\.[._1`|
    AlgorithmId: EFI_TCG2_BOOT_HASH_ALG_SHA512 (0xd)
    Digest: 64 bytes
00000006  ce 2a ff 13 fa 82 7e 1a  49 d3 fa cb 4b c2 9c 8a  |.*....~.I...K...|
00000016  68 18 b4 e8 88 27 fc c3  4a e7 8b 8c c4 23 81 f6  |h....'..J....#..|
00000026  d3 34 c6 67 4d 7c 6e 9d  9c db 4a 27 a3 91 37 7e  |.4.gM|n...J'..7~|
00000036  d4 3c 2f 40 57 52 eb 77  a2 6c 49 d3 9b af a8 b7  |.</@WR.w.lI.....|
  Event: 38 bytes
00000006  28 68 64 30 2c 67 70 74  31 29 2f 67 72 75 62 2f  |(hd0,gpt1)/grub/|
00000016  78 38 36 5f 36 34 2d 65  66 69 2f 74 70 6d 63 6d  |x86_64-efi/tpmcm|
00000026  64 2e 6d 6f 64 00                                |d.mod.|
```

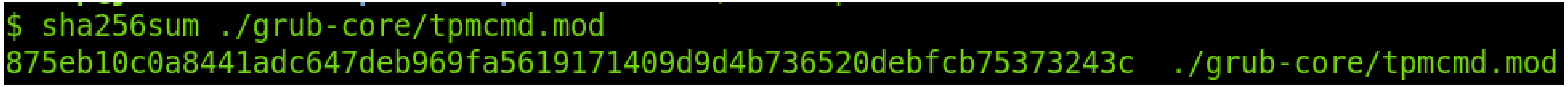
Calculate has of grub module

- Digest from Eventlog



```
    AlgorithmId: EFI_TCG2_BOOT_HASH_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!