In Containers We Trust? Building Trust in Containerized Environments

Avery Blanchard', Gheorghe Almasi*, James Bottomley*, Hubertus Franke*

Duke University', IBM Research*

Motivation

- ***Build trust in containers*** through cryptographic measurements rooted in **trusted hardware**
- ***Measurement and attestation*** of containerized workloads
- Goal: Enable container attestation through the measurement of individual container integrity

Background
Trusted Platform Module TPM

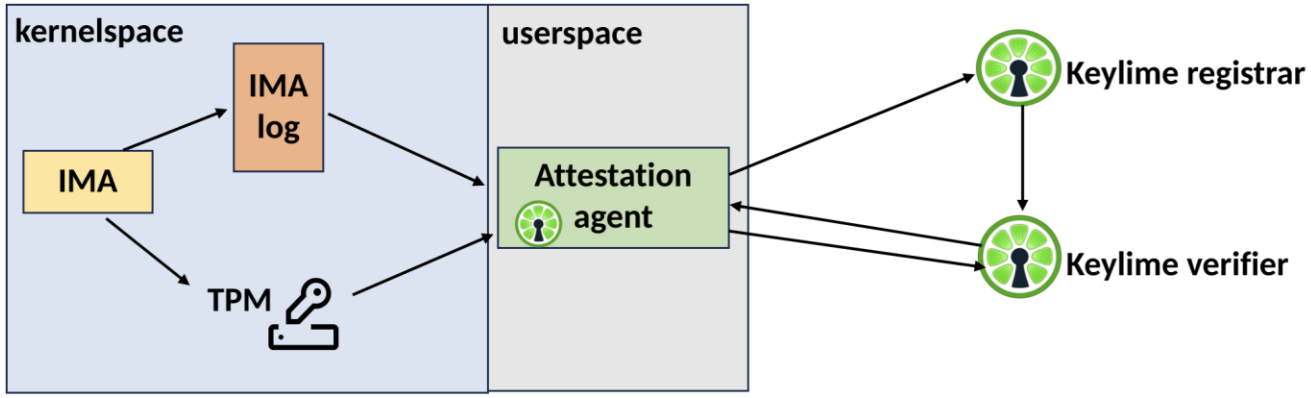
- Cryptographic coprocessor designed to secure hardware
- Components
  - Key generation
  - Secure storage
  - Unique hardware identity
- Applications
  - Secure boot
  - Disk encryption
  - Attestation and trust (Keylime)

Background
Linux Integrity Measurement Architecture IMA

- Measurement, appraisal, and storage of file integrity data
- Cryptographic hashes of file contents are stored in a TPM-based non-repudiable logs
- Files measured based on system policies
- Detect changes in file integrity due to remote/local attacks

Background
Attestation

- Verification of system integrity relying on trusted hardware
- TPM enables remote attestation of system software from boot measurements through runtime

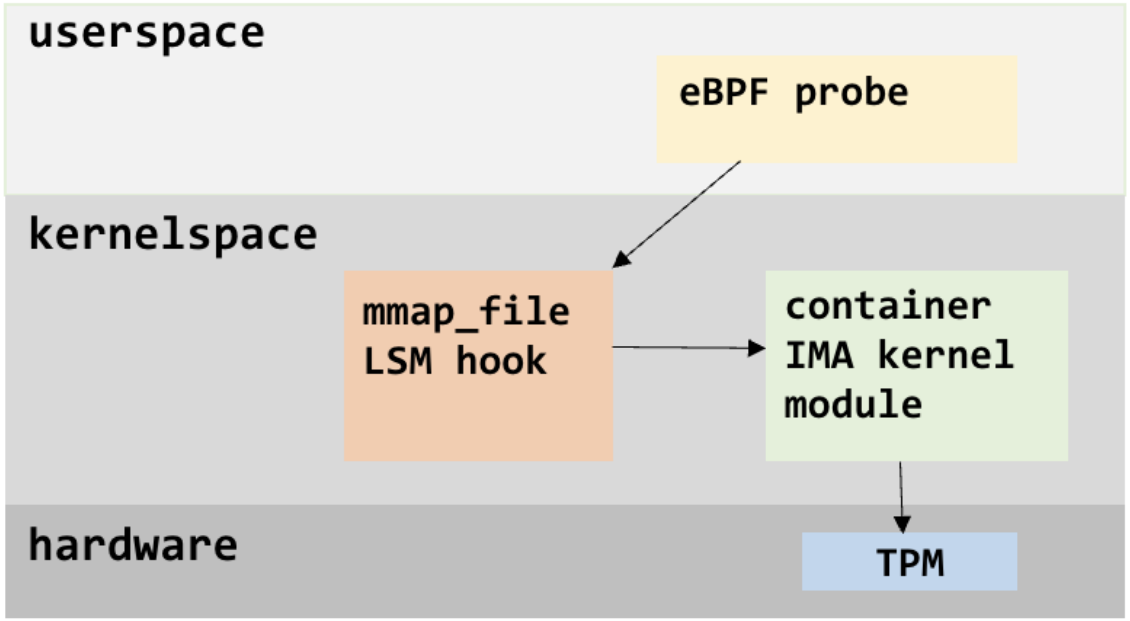


Background: Kernel Extensions

- User-defined programs loaded into the OS kernel
- Kernel Modules
  - Programs that can be loaded into the OS (device drivers, file systems, etc)
- eBPF
  - Mechanism allowing user-define programs to run sandboxed in the privileged kernel context
  - Wide variety of hooks located across subsystems

Extending IMA to Containers using eBPF

- IMA currently does not have namespace support
  - Cannot be used to verify the integrity of individual containers
- Through leveraging the kernel's support of eBPF, we can add namespace support to IMA without requiring changes to the kernel



https://github.com/avery-blanchard/container-ima

Extending IMA to Containers using eBPF

- eBPF
  - Provides visibility into a container's executable content without changes to the OS
  - Sleepable eBPF program hooking into **`mmap_file`** LSM
    - Same LSM hook used by IMA to provoke measurements in the kernel
  - Provokes measurement through calling kernel module exported function
- Kernel module
  - Measures and stores integrity data in the host IMA log
  - Namespaced measurements are stored

    `HASH(FILE HASH | NS)`

Container Integrity Measurement

- With the eBPF extension of IMA, container file integrity is measurement throughout runtime
- Building a policy for this system ***introduces more and more complexity to do attestation at this scale***
  - Whitelist of file hashes for every container
- Where can we go from here?

Container Image Measurement

- From the operating system level, visibility into container creation is limited
- **`Unshare`** system call
  - Disassociate parts of a process' execution context that are currently being shared
- Through filtering calls to **`unshare`** based on policy, we have visibility into container images through the file system of the new namespace

Provoking Container Image Measurements

- Add an LSM hook into the **unshare** system call to provoke a measurement based on policy
- The introduction of this hooks allows for future work on image appraisal and access control from the OS-level

Image Measurement

- Single measurement for the image
- Traverse the file system, concatenating after each measurement

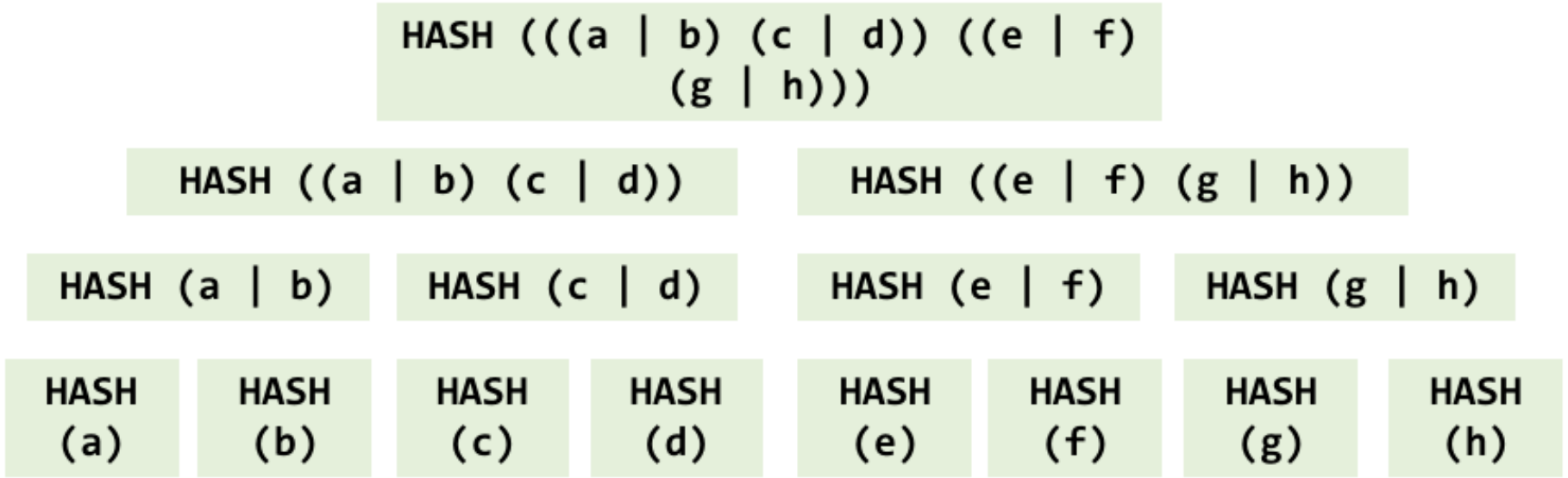


Image Measurement Storage

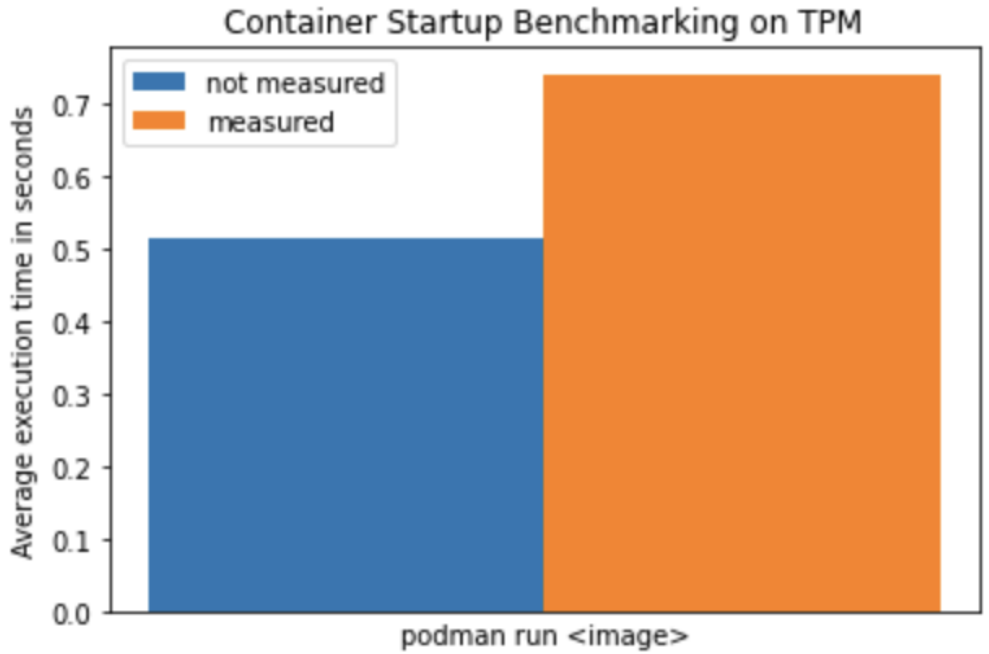
- Image digests are stored as a single entry in the host IMA log
- Digests are logged with their namespace as an identifier
- Digests are extended to PCR on a TPM

```
[root@ima-fedora1 ~]# sudo tail /sys/kernel/security/ima/ascii_runtime_measurements
11 2cad4f3c210e2100f8de33533051f8f21e177d4f ima-ng sha256:525dd35d139c1bfa5c74faecbd0a6812032649c4ab022f786de2406b6037ae47 4026532367:/usr/lib64/libpcre2-8.so.0.11.0
11 ffeccc4ba58d619611ca57a3f0e906ad6ded9df8 ima-ng sha256:32f081813e52019823645d2592e779700d9a0328109b13775b2ce0b26df0affe 4026532367:/usr/lib64/libcap-ng.so.0.0.0
10 71cf2fa458c60d0f53b16a15f94758f84bbbeff9 ima-ng sha256:b39a71b04f7ff3ae6e91fd69b54e08d36e9c16cafe4a109867bc121eade42444 /usr/bin/containerd-shim-runc-v2
11 adfcdcc1ee5dd95e7a3fe2bae43b704aa0039f58 ima-ng sha256:6f3671bd20ed4b121ee0f8175a3b312300a33a79444239371863bfb5c6ea8e97 4026532559
10 5c4f7bddfc3988228ca9d50629d524b4dacb1454 ima-ng sha256:d3f7d10e296e5c626ea78539cb38cd8dffd043bea3627da35ce3b20c0ac68014 /hello
11 5d88c02290271a3949c899f6ca85d88e7e695781 ima-ng sha256:aa15dcbe503ee00f9f72924e8bea3b0c9bd42ca5205c40e70dde9d7c963e56e0 4026532559:/hello
```

Policy Enforcement

- Image measurements are enforced based on a system policy
- This policy determines what flags passed to **`unshare`** warrant a measurement
  - Container runtimes affect which flags should provoke a measurement and should be reflected in the policy

Evaluation



- Benchmarking container startup time when image is measured on a machine with a hardware TPM

Current State of Image Digests

- Current image digests are dependent on images layers, manifest files, image ids, ...
- From the operating system, the only thing visible in the final image
- A digest of the image itself is needed to be provided to extend the chain of trust from hardware up to each container instance
- What does the path to kernel-verifiable measurement of the container look like?

Future Work

- Improve policy enforcement
- Container attestation with Keylime

Thank you!

Avery Blanchard: avery.blanchard@duke.edu

We will be presenting this work at LPC 2023 in November