A Formal Architecture for Trustworthy Remote Attestation

Grant Jurgensen[1] Adam Petz[2] Perry Alexander[2]

[1]Kestrel Institute
grant@kestrel.edu

[2]Institute for Information Sciences
The University of Kansas
{ampetz, palexand}@ku.edu

Outline

Definitions

- Attestation: the process in which some party provides evidence of its state and/or identity to another party to establish trust
- Appraisal: the analysis of attestation evidence resulting in a trust judgment
- Kinds of attestation
  1. Local attestation
  2. Remote attestation

Attestation and Verification

- Verification produces stronger results, but may be prohibitively time-consuming
- Attestation frequently relies on probabilitic evidence, but is more easily applied
- Attestation and verification are complimentary approaches.

Goals

1. Design and prototype implementation of a system architecture for trustworthy remote attestation
2. Formal specification and verification of the attestation architecture

Copland Definition

- Copland[1] is a domain-specific language (DSL) for attestation protocols
- Protocols are executed with respect to some initial evidence, and produce further evidence

$$t ::= a \mid \texttt{@}p\ \texttt{[}t\texttt{]} \mid t\ \texttt{->}\ t \mid t\ \texttt{\textasciitilde}\ t \mid \ldots$$
$$a ::= name_{asp}\ p\ name_{targ} \mid \texttt{!} \mid \ldots$$

Figure: An abridged grammar of the Copland protocol language. The t rule defines a top-level protocol term. p represents a place. $name_X$ is an identifier corresponding to an external measurement service ($X = asp$) or target of measurement($X = targ$).

[1] https://ku-sldg.github.io/copland/

Copland Examples

Remote attestation

```
@p1 [(hashFile p1 "/etc/passwd") -> !]
```

Cross-domain attestation

```
@p2 [(hashFile p1 "/etc/passwd") -> !]
```

Mixed attestation

```
@p1 [ @p2 [(attest p1 hashFile) -> !] ->
      (hashFile p1 "/etc/passwd") -> !]
```

Design Goals

1. Guarantee confidentiality of private keys and integrity of measurements
2. Accommodate a variety of existing systems

Challenges

A separation kernel would provide optimal integrity. However, it would not accommodate as much existing software as would a general purpose OS.

Design

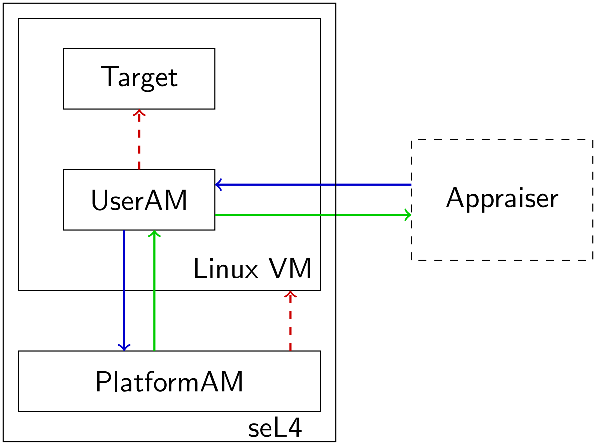

Figure: The attestation architecture. Red dashed arrows represent measurement. Blue arrows represent incoming Copland attestation requests. Green arrows represent outgoing Copland evidence.

Implementation Details

- Verified components: CakeML compiler, HACL* cryptographic primitives, formal Copland specification
- Communication
  - Inter-VM communication occurs over a static CAmkES dataport. A kernel module provides a file-like interface to Linux userspace.
- Privileges
  - CAmkES components are assigned static privileges over communication channels.
  - Communication from the Linux environment is protected only by regular file access controls.

Key Release

- Keys are released to AM over the course of system boot to avoid starting a compromised AM with the confidential key.
- Lower layers release keys to higher layers after sufficient measurement.
- The lowest level layer either starts with its key, or it may be derived from some hardware root-of-trust.

Verification

- We formally model[2] the abstract, component-level design of the system in Coq
- We embed CTL[3] and prove temporal properties

[2] https://github.com/ku-sldg/attarch-model/tree/thesis
[3] https://github.com/ku-sldg/CTL/tree/thesis

Conclusion

- Future work
  - Incorporate robust measurements into PlatformAM
  - Apply architecture to system with a hardware root of trust
  - Improve CTL proof automation