Towards Trustworthy Architectures for Secure Cloud Servers and End-User Devices

Jakub Szefer and Prof. Ruby B. Lee
Princeton University

http://palms.princeton.edu/
Security Challenges to Cloud Servers

- Security challenges arise from sharing resources with unknown people, no control over management software, and unknown insiders where servers are located
Security Challenges to End-User Devices

- Challenges arise due to unvetted third-party applications or loss of the physical device
Leveraging Hardware for Security

• Hardware prevents attacks “from below”, it's lowest level in the system

• Hardware security:
  – performance
  – immutability
  – secret recovery prevention

• Can leverage existing hardware or propose new to meet the threat model
Hardware System Lifecycle

• Hardware can provide protections at all points in the lifecycle of cloud servers or end-user devices

• **Hardware can measure and protect:**
  – measure code, data, and metadata
  – protect code, data, and metadata
  – store secrets (keys)

• **Hardware should be attested:**
  – ensure proper hardware is installed
  – validate configuration
  – check protections are being enforced
Hardware System Lifecycle

- Hardware can provide protections at all points in the lifecycle of cloud servers or end-user devices

**Commissioning**
- Measure the correct hardware is present.
- **Open issues:** Integrate hardware metering, PUFs and remote code integrity checking

**Runtime**
- Attest security hardware configuration, measure any trusted system software
- **Open issues:** Extend automatic power-on tests to check security features

**System Shutdown**
- Clean up protections, scrub code or data, lock secrets
- **Open issues:** Unexpected shutdown, cold boot type attacks

**System Setup**
- Protect code and data at rest
- **Open issues:** Scrubbing data before maintenance

**Maintenance**
- Scrub code and data
- **Open issues:** Secure recycling of hardware

**Decommissioning**
Hardware System Lifecycle

- Runtime protections and measurements; need to protect code, data and metadata

Measure loaded code, data or metadata; unlock them; install protections

Open issues:
Tailored attestation

Enforce protections

Open issues: Runtime attestation of software, attestation of the protections

Protect migration of code, data and metadata

Open issues:
Self-migration, migration under untrusted management software

Code & Data Execution

Local I/O

Biometrics

Measure users

Open issues:
Calibrate biometrics hardware

Scrub code and data, lock secrets

Open issues:
Secure erasure, data tracking to erase all data

Code and Data Termination

Migration
Hardware System Lifecycle

- Runtime protections and measurements; need to protect code, data and metadata

Open issues:
- Side-channel detection, side-channel prevention

Information can leak through various side-channels in the shared environment

Secure communication keys

Open issues:
- Communication through untrusted system software or network interfaces

Information Leakage

Code & Data Execution

Code and Data Termination

Data Communication
Research Questions

- How to verify that the design is correct?
- How to attest that the protections mechanisms are working?
- How to attest enforcement of the correct protections?
- How to leverage existing hardware or environment?
  - Insider threats and physical attacks
- How to protect lost or stolen devices?
Verifying Security of the Design

• Before committing to hardware, need to raise confidence in the design

• Many tools available for security protocol testing: HERMES, Casrul, AVISPA, Scyther, ProVerif, Athena, …

• Want similar tool(s) for secure architectures

• Work exists on formal verification of whole architectures, but needs large teams, long time; verifying individual parts can also give good results*

* Composition is very important unsolved problem.
Verifying Security of the Design

• Proposition: verify critical interactions and mechanisms

• Goal to leverage existing tools that architects know how to use

• Augment functional verification with checks for security problems

• Design heuristics to generate security invariants that check for attacks based on attacker capabilities
Attesting Protection Mechanisms

- Hardware security mechanisms need to be checked that indeed they offer the advertised protections
- Power-on testing is used for error checking, expand for security checks
- **Challenge-response** tests

![Diagram of functional units and circuits](image)
Attesting Protection Mechanisms

- Hardware security mechanisms need to be checked that indeed they offer the advertised protections.

- Power-on testing is used for error checking, expand for security checks.

- **Time-based** tests.
Attesting Enforcement of the Protections

- At startup, measure and attest correct code, data, and protection specifications were loaded.
- At runtime, measures and attest enforcement of the protections.
- HyperWall example.
Leveraging Existing Hardware

• Existing hardware can also be used if it can be applied correctly

• Hardware manufacturers offer many **performance** enhancing features in commodity microprocessors, e.g.
  - Ring-based hierarchical protection
  - Memory translation hardware
  - Data structures for controlling world switches
  - Hardware virtualized devices

• **Leverage the hardware for security**
Leveraging Existing Hardware

- Existing hardware can also be used if it can be applied correctly

- NoHype example
Leveraging Existing Environment

- Environment of the cloud server or end-user device is usually assumed neutral or adversarial.

- **Can we use the environment to our benefit?**

- E.g. end-user devices are often mobile, connect from different locations, etc.

- E.g. cloud servers, are often used in data centers, many redundant network connections, many other servers near by, many sensors, etc.
Preventing Insider Threats

- Data centers are becoming backbone of computing, they have their unique threats, e.g. insider threats
- But also unique opportunities as in the available sensors that measure environment around the server
- Can leverage sensors for physical insider threat prevention

Event detection

\[ T_{detect} \]

Estimated time of attack

\[ T_{attack} \]
Lost or Stolen End-User Devices

• Devices are portable, can be lost or stolen while users are logged in

• One-time user authentication is not sufficient, can we leverage behavior to **continuously authenticate users through hardware sensors**?

• Software behavior will be different when new person picks up the device; can we leverage **performance counters and hardware events to detect malicious behavior**?
Research Questions

- How to verify that the design is correct?
- How to attest that the protections mechanisms are working?
- How to attest enforcement of the correct protections?
- How to leverage existing hardware or environment?
  - Insider threats and physical attacks
- How to protect lost or stolen devices?
Thank you!

Towards Trustworthy Architectures for Secure Cloud Servers and End-User Devices

Jakub Szefer and Prof. Ruby B. Lee
Princeton University

http://palms.princeton.edu/