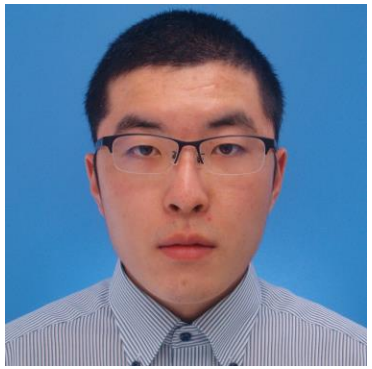
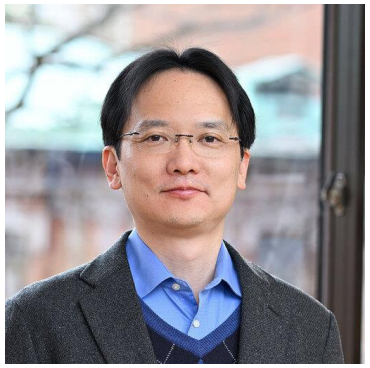
The University of Tokyo
Shinagawa Lab.
System Software & Operating Systems

BadAML:
Exploiting Legacy Firmware Interfaces to Compromise Confidential Virtual Machines





Satoru Takekoshi

Manami Mori

Takaaki Fukai

Takahiro Shinagawa

UTokyo

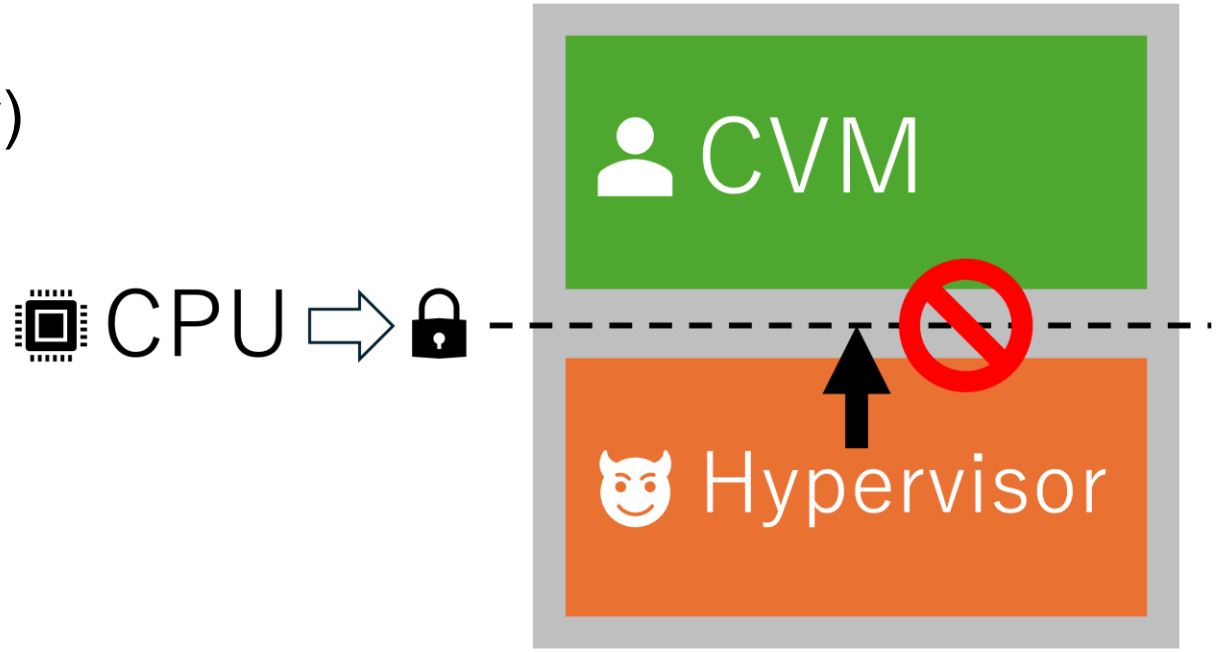
TOKYO METROPOLITAN UNIVERSITY
東京都立大学

AIST

UTokyo

Confidential Virtual Machine (CVM)

- A virtual machine (VM) acting as a trusted execution environment (TEE)
- CPU-enforced protection from malicious hypervisors
  - Attestation (verified initial state)
  - Memory encryption (confidentiality)



Related Work

- SEVurity [IEEE S&P ’20]
  - Break encrypted-memory integrity on a specific processor (AMD SEV-ES)
- One Glitch To Rule Them All [CCS ‘21]
  - Hardware-based attack against a specific processor (AMD-SP)
- HECKLER [USENIX Security ‘24]
  - Interrupt injection attack targeting a specific OS (Linux)

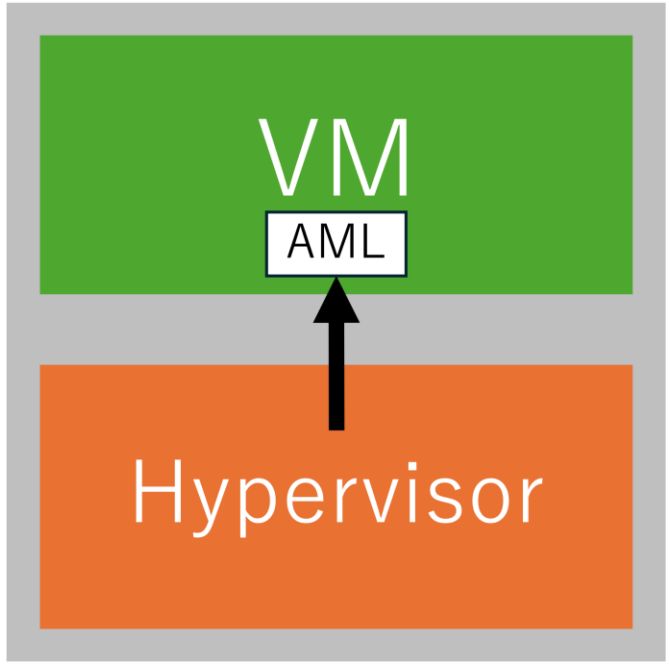
→ *Existing attacks depend on specific hardware (HW) or OS designs*

BadAML (AML Injection Attack)

- OS- and HW-independent attack against CVMs
- Malicious cloud host can execute arbitrary code inside CVMs
- Exploits AML as an attack vector
  - An industry-standard firmware interface
  - Widely supported by OSs and hypervisors

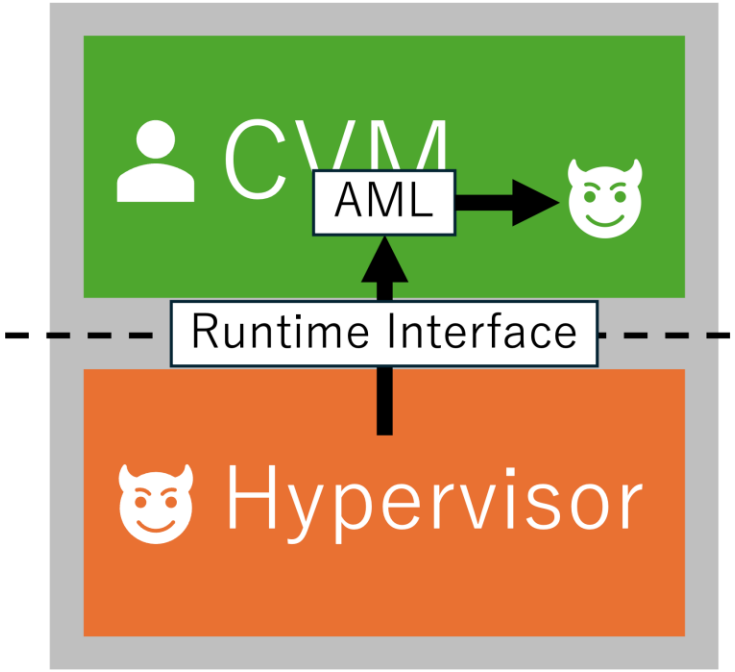
ACPI / AML

- ACPI = Advanced Configuration and Power Interface
  - Interface for HW configuration and power management from the OS
- AML = ACPI Machine Language
  - Bytecode used to configure HW through ACPI
  - Has memory access capability
  - Executed inside the OS
- AML is passed from hypervisor to the VM
  - During boot



BadAML Overview

- Injects AML into a CVM via the “Runtime Interface” **after boot**
  - Bypasses CVM attestation at boot time
- AML rewrites CVM memory
  - Leads to arbitrary code execution in guest OSes
    - Proof-of concept demonstrated on Linux and Windows



Runtime Interface

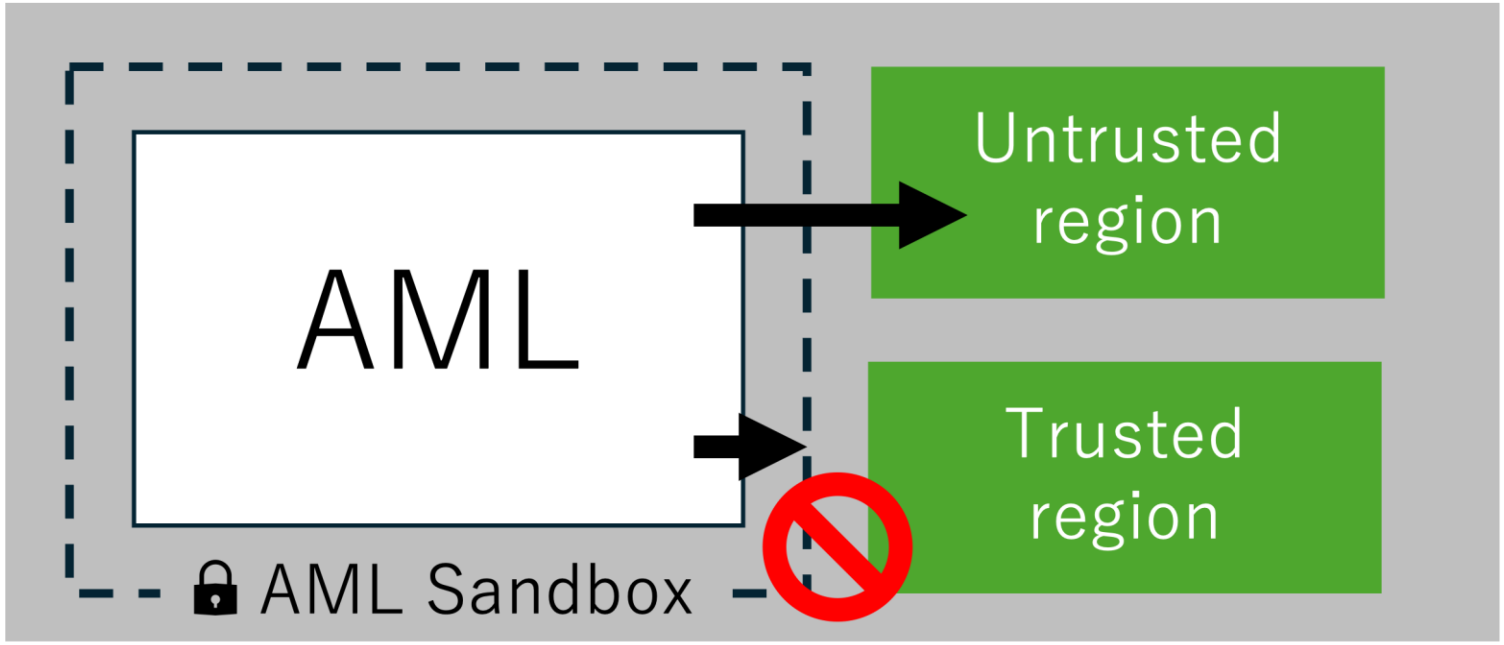
- Has existed since the era of traditional VMs
  - Not considered a security issue in the traditional threat model
- Essential for flexible VM configuration
  - Not easily disabled
- Widely used in read-world environments
  - “fw_cfg” (QEMU)
  - “UEFI configuration blob” / “PCAT BIOS Helper” (Azure)

Possible Mitigations and Limitations

- Disable ACPI
  - Requires abandoning compatibility
- Disable Runtime Interfaces
  - Eliminates VM's flexible configurability
- Measure ACPI tables
  - Requires separate auditing of ACPI tables
    - Who and how to audit ACPI tables from untrusted cloud?

AML Sandbox

- Enforces memory access policy on AML
  - Use CVM trust boundary as policy
    - Deny access to trusted regions
    - Allow access to untrusted regions

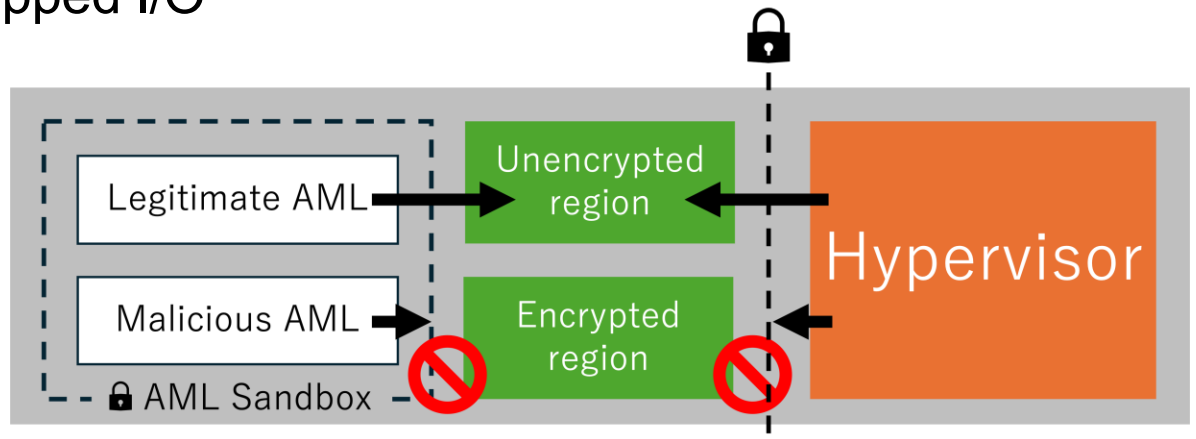


Policy Implementation

- Uses the encryption bit to determine the trust boundary
  - Trusted regions = encrypted for protection
    - Used for guest OS code and data
    - Contain no device information

    → **Access is denied**
  - Untrusted regions = unencrypted for communication
    - Used for I/O buffers and memory-mapped I/O
    - Needed to control hardware devices

    → **Access is allowed**



Implementation

- Linux
  - Modify AML interpreter source code
    - 396 LoC in C
- Windows
  - Patch AML interpreter binary
    - Hook a function to call our AML Sandbox module
    - 361 LoC in C + 5-bytes binary patch

Evaluation Setup

- Tested on 18 configurations of cloud CVMs
  - Vendor: AWS, Azure, GCP, Oracle
  - OS: Linux, Windows
  - CPU: AMD SEV-SNP, Intel TDX

Evaluation Results

- Successfully prevented BadAML attacks
  - AML sandbox detected and stopped malicious memory access
- Preserved normal AML functionality
  - Legitimate AML code accessed only unencrypted regions
- Practically negligible overhead
  - 14.7% on Linux and 75.8% on Windows per AML memory access
  - Overhead mostly occurs at boot time and with small access counts

Conclusion

- BadAML enables arbitrary code execution inside CVMs
  - OS- and HW-independent
    - Exploits an industry-standard firmware interface (AML/ACPI)
- AML Sandbox successfully prevents BadAML attacks
  - AML can access only unencrypted memory

Thank you!