

# Session 08

## System Isolation

Security of Information Systems (SIS)

Computer Science and Engineering Department

November 26, 2024

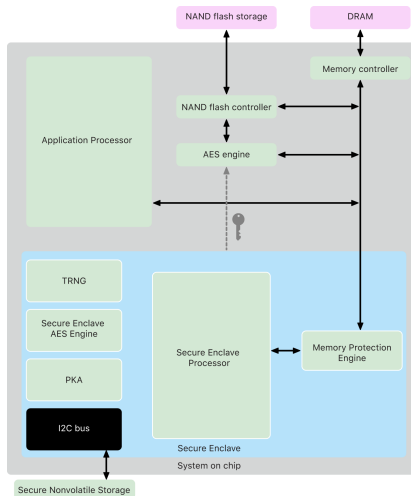
# Papers

- ▶ Application and analysis of the virtual machine approach to information system security and isolation
- ▶ My VM is Lighter (and Safer) than your Container

# Apple FaceID, TouchID, SEP

- ▶ Application Processor (AP) vs Secure Enclave Processor (SEP)
- ▶ *Secure Enclave* - similar to Arm TrustZone
- ▶ hardware-based isolation
- ▶ biometrics, keys are only handled by SEP
- ▶ specific interface between AP and SEP

# Apple SEP (2)



<https://support.apple.com/en-ke/guide/security/sec59b0b31ff/web>

# Run Untrusted Code

- ▶ apps, plugins, codecs
- ▶ software not written by you, not-verified
- ▶ damage control
- ▶ kill it if it misbehaves
- ▶ ensure misbehaving app does not alter the system

# Confinement Types

- ▶ hardware: different hardware systems, air gap
- ▶ virtual machine: isolate OSes in a single machine
- ▶ process: sandboxing, jailing
- ▶ application: software fault isolation

# Software Fault Isolation

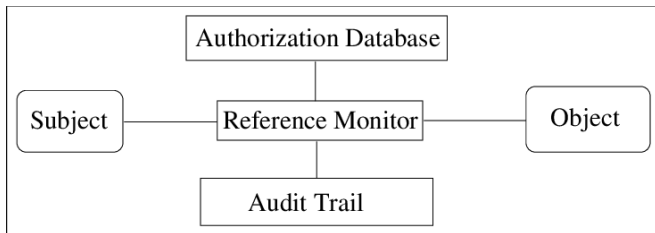
- ▶ isolate components in their *fault domain*
- ▶ part of the same address space
- ▶ requires some OS/hardware support to separate addresses
- ▶ Mogoşanu et al.: MicroStache: A Lightweight Execution Context for In-Process Safe Region Isolation

# Reference Monitor

- ▶ mediates requests, implements policy, enforces isolation and confinement
- ▶ must always be invoked
- ▶ tamperproof
- ▶ validated



## Reference Monitor (2)



[https://www.researchgate.net/publication/2390175\\_Secure\\_Information\\_Flow\\_in\\_Mobile\\_](https://www.researchgate.net/publication/2390175_Secure_Information_Flow_in_Mobile_)

Bootstrapping\_Process

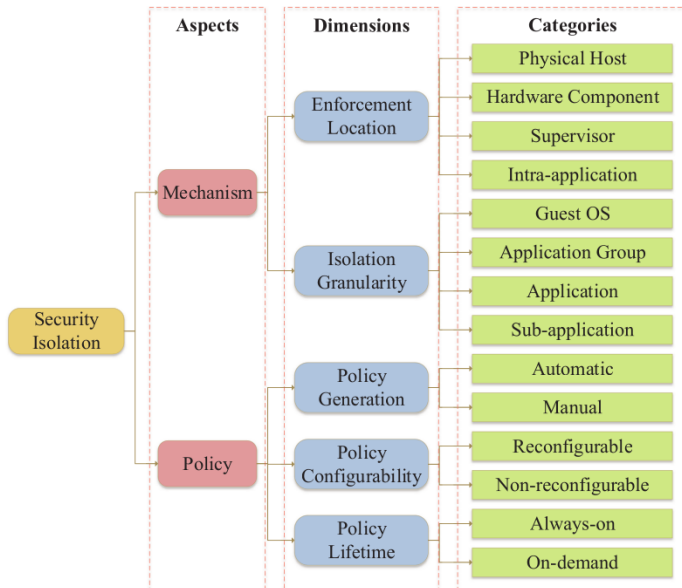
# Principles and Goals

- ▶ least privilege
- ▶ privilege separation
- ▶ safely execute a non-trusted program
- ▶ harden a system that runs programs that increase its attack surface
- ▶ isolate what can happen if a vulnerability is exploited

# Mechanism and Policy

- ▶ mechanism: how goals are achieved
- ▶ policy: rules that achieve isolation goals
- ▶ mechanism: mostly implementation
- ▶ policy: mostly configuration

# Mechanisms and Policies



# System Isolation

- ▶ isolate app, group apps or entire OS
- ▶ prevent it from hurting other components
- ▶ virtual machines, library OS, containers
- ▶ we consider sandboxing, mandatory access control, software fault isolation (SFI) to be app-centric mechanisms (not system-centric)

# Trusted Computing Base (TCB)

- ▶ trusted system components (by the reference monitor)
- ▶ critical parts of the system
- ▶ if exploited, might jeopardize the security of the entire system
- ▶ aimed to be small (reduced attack surface)

# Hardware Protection

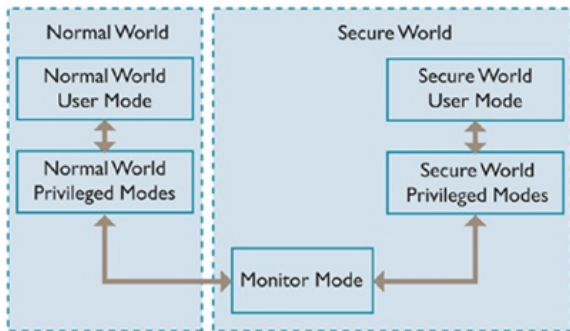
- ▶ provide security isolation for shared resources
- ▶ passive components: TPM (*Trusted Platform Module*)
- ▶ active components: control critical system operations

# Trusted Execution Environment (TEE)

- ▶ secure area on CPU
- ▶ code run is secure: confidentiality and integrity
- ▶ runs in parallel with OS



## TEE (2)



<https://resources.infosecinstitute.com/topic/understanding-ios-security-part-1/>

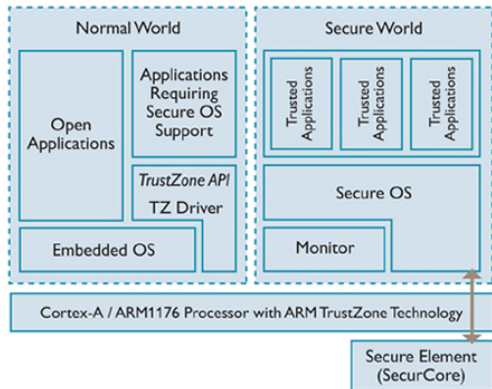
# Intel TXT

- ▶ Trusted eXecution Technology
- ▶ attest platform/operating system
- ▶ uses TPM and cryptography to validate/measure code that can be trusted

# Arm TrustZone

- ▶ Arm TZ
- ▶ two worlds: secure and non-secure
- ▶ rich OS runs in non-secure worlds, security-specialized code in secure world
- ▶ aim to reduce attack surface

# Arm TrustZone



<https://blog.quarkslab.com/introduction-to-trusted-execution-environment-arms-trustzone.html>

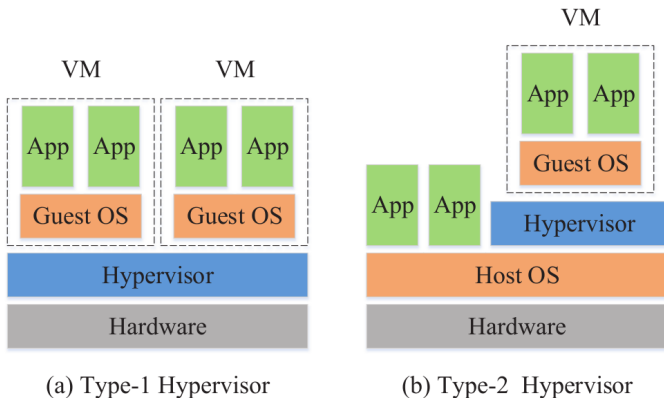
# Intel SGX

- ▶ Software Guard eXtensions
- ▶ specialized instructions
- ▶ user-level code allocates enclaves
- ▶ protected from higher privilege level components
- ▶ secure remote computation
- ▶ cache DRAM side-channel attack

# Secure Enclave

- ▶ on Apple iOS / watchOS devices
- ▶ fingerprint data completely walled from the OS
- ▶ uses a SEP (*Secure Enclave Processor*), SEP OS
- ▶ based on Arm TZ

# Virtualization



Rui Shu et al.: A Study of Security Isolation Techniques

# Virtual Machine

- ▶ run an isolated OS instance on top of a supervisor component (hypervisor)
- ▶ hypervisor or VMM (*Virtual Machine Monitor*)
- ▶ malware in a VM cannot infect host OS or other VMs



# Covert Channels

- ▶ side channels
- ▶ use CPU, memory, cache information from one VM to determine what's happening on the other VM

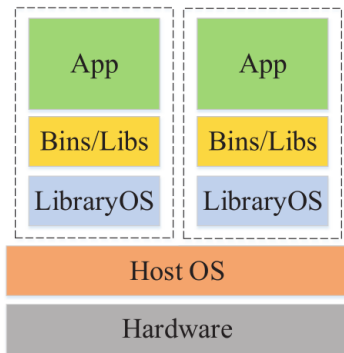
# VMM Detection

- ▶ VM platforms emulate simple hardware
- ▶ VMM introduces time latency variances
- ▶ VMM shares TLB (*Translation Lookaside Buffers*)

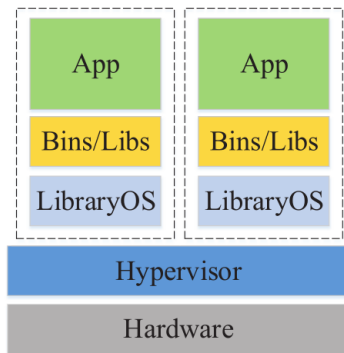
## Type-1 vs Type-2

- ▶ reduced TCB vs additional flexibility
- ▶ efficiency for Type-1

# Library OS



(a) LibraryOS on Host OS



(b) LibraryOS on Hypervisor

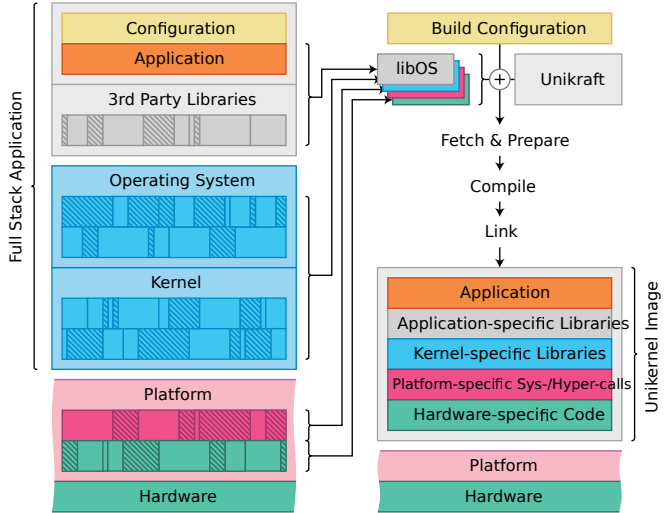
Rui Shu et al.: A Study of Security Isolation Techniques

# Library OS Characteristics

- ▶ unikernel
- ▶ OS functionality as user library/libraries
- ▶ single-image app, can run on top of hypervisor or hardware
- ▶ no need for user-level/kernel-level transitions
- ▶ difficult to run multiple instances: use a hypervisor
- ▶ reduce the attack surface

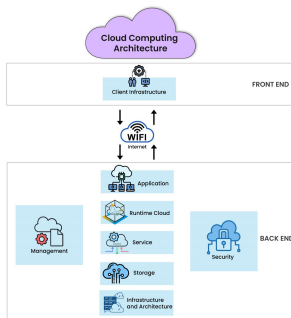
# Implementations

- ▶ ClickOS: virtualized software middle box
- ▶ LKL (*Linux Kernel Library*)
- ▶ My VM is Lighter (and Safer) than Your Container:  
<http://cnp.neclab.eu/projects/lightvm/lightvm.pdf>
- ▶ <https://awesomeopensource.com/projects/unikernel>



<https://unikraft.org/docs/concepts/build-process/>

# Cloud Computing



<https://www.infosectrain.com/blog/what-is-cloud-computing-architecture-and-its-components/>



# Security of Cloud Computing

- ▶ isolate the VM from the host
- ▶ prevent the VM from tampering with the host
- ▶ what about the other way around?

# Confidential (Cloud) Computing

## Current encryption technologies



### Data at rest

Stored and inactive data is encrypted on servers in databases and is not moving through networks



### Data in transit

Data is encrypted prior to transit on public and private networks

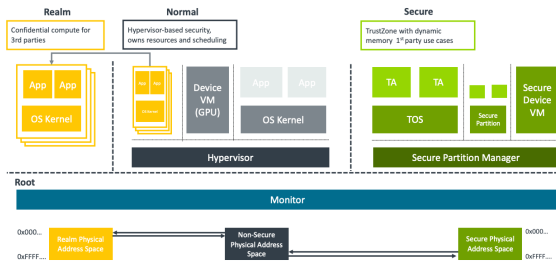
## Confidential computing



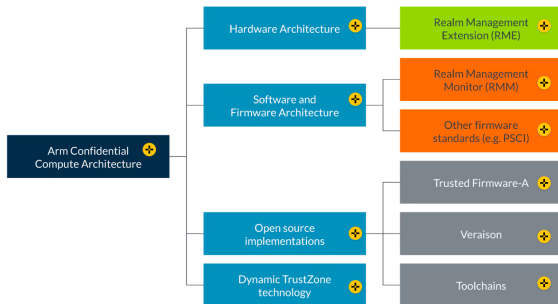
### Protect data in use

Data is protected in use by RAM encryption and hardware-based technologies that protect data during computation

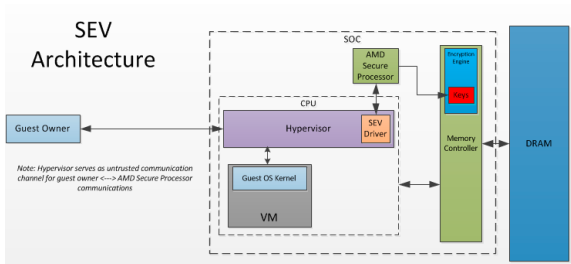
# Arm CCA



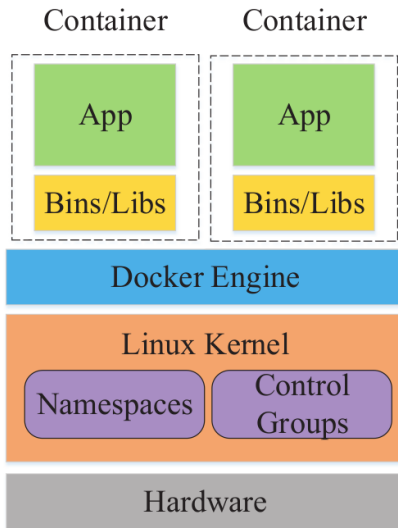
# Arm CCA Technologies



# AMD SEV



# Containers



# Container

- ▶ restricted environment
- ▶ applications or application groups
- ▶ sandboxing only provides a certain set of privileges
- ▶ containers provide a dedicated isolated environment

- ▶ Linux Containers
- ▶ use Linux namespaces: PID, network, IPC, mount, user, UTS
- ▶ Linux control groups (cgroups): limits, accounts, isolates resource usage



# OS vs. Application Containers

- ▶ OS: provided an entire distro, similar to a virtual machine (LXC)
- ▶ app: provide an environment for running a single service (Docker)

# Containers vs. hypervisors

- ▶ containers are faster to create, deploy, run
- ▶ containers are lighter (reduced overhead)
- ▶ hypervisors are more secure: reduced TCB, no common kernel

# Keywords

- ▶ confinement
- ▶ isolation
- ▶ resource monitor
- ▶ TCB
- ▶ TEE
- ▶ Intel TXT
- ▶ Intel SGX
- ▶ Arm TZ
- ▶ VMM
- ▶ hypervisor
- ▶ library OS
- ▶ unikernel
- ▶ Arm CCA
- ▶ AMD SEV
- ▶ container
- ▶ LXC
- ▶ Docker

# Resources

- ▶ A Study of Security Isolation Techniques
- ▶ CS155: Computer and Network Security: Isolation and Sandboxing
- ▶ <https://blog.risingstack.com/operating-system-containers-vs-application-containers/>
- ▶ <https://www.arm.com/architecture/security-features/arm-confidential-compute-architecture> Arm CCA
- ▶ <https://www.amd.com/en/developer/sev.html> AMD SEV