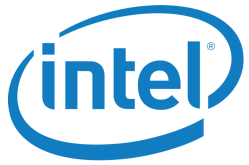
SCRUTINIZER: Towards Secure Forensics on Compromised TrustZone

Yiming Zhang, Fengwei Zhang, Xiapu Luo , Rui Hou, Xuhua Ding,
Zhenkai Liang, Shoumeng Yan, Tao Wei, Zhengyu He

Trusted Execution Environment (TEE) Protects Data in Use

TEE is a key technology in Confidential Computing; Hardware-assisted security design.



Intel SGX/TDX

Keystone

AMD SEV

Arm
TrustZone/CCA

TEE has been applied to the computing platforms and commercial products of several companies.

Microsoft
Azure Cloud

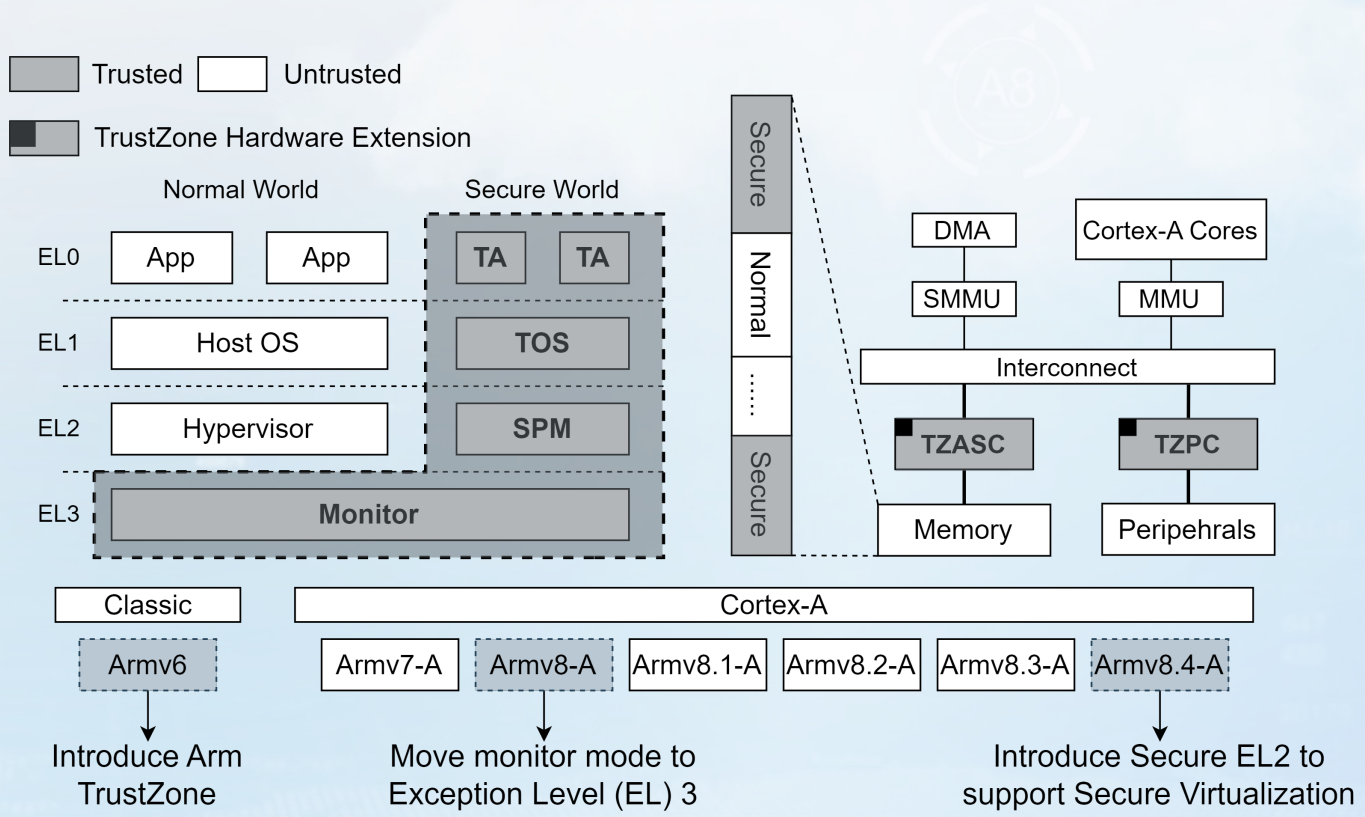
NVIDIA H100

Google
Cloud

Ant Occlum

Arm TrustZone TEE

- TrustZone was first introduced in ARMv6 and provides a hardware-based isolation of two execution environments
  - *Normal World* and *Secure World*
- TrustZone ensure isolation two words through hardware extensions (e.g. TZASC and TZPC)
- Since Armv8.4, TrustZone was extended to support virtualization **(Secure EL2)**



TrustZone System Vulnerabilities

More than 207 vulnerabilities[#] have been identified in Arm **TrustZone system** [*] over the past five years

TEE System	CVE Databases	SVE Databases	Scientific Publications	Miscellaneous Reports	Source Code	Total
Qualcomm TEE	92	-	-	7	-	99
Trustonic TEE	5	17	-	4	-	26
Huawei TEE	3	-	-	1	-	4
Nvidia TEE	10	-	-	-	-	10
Linaro TEE	3	-	-	1	36	40
Other	11	-	15	2	-	28
Total	124	17	15	15	36	207

[#]Sok: Understanding the prevailing security vulnerabilities in trustzone-assisted tee systems. In IEEE SP, 2020

[*] We use 'TrustZone systems' to refer to the software in Secure World, including trusted apps, trusted OS and secure hypervisor (S.EL0 – S.EL2)

Motivation: TrustZone Inspection

More than 207 vulnerabilities have been identified in Arm TrustZone system over the past five years

It's crucial to add extra security *forensics** to check TEE systems

TEE System	CVE Databases	SVE Databases	Scientific Publications	Miscellaneous Reports	Source Code	Total
Qualcomm TEE	92	-	-	7	-	99
Trustonic TEE	5	17	-	4	-	26
Huawei TEE	3	-	-	1	-	4
Nvidia TEE	10	-	-	-	-	10
Linaro TEE	3	-	-	1	36	40
Other	11	-	15	2	-	28
Total	124	17	15	15	36	207

#Sok: Understanding the prevailing security vulnerabilities in trustzone-assisted tee systems. In IEEE SP, 2020

*Capture snapshots of target for backend analysis,
letting the platform owner for incident response or periodic security scans

Existing Approaches Presents Limitations

1. Out-TEE

External methods for TrustZone are hindered by TrustZone’s protection.

2. In-TEE

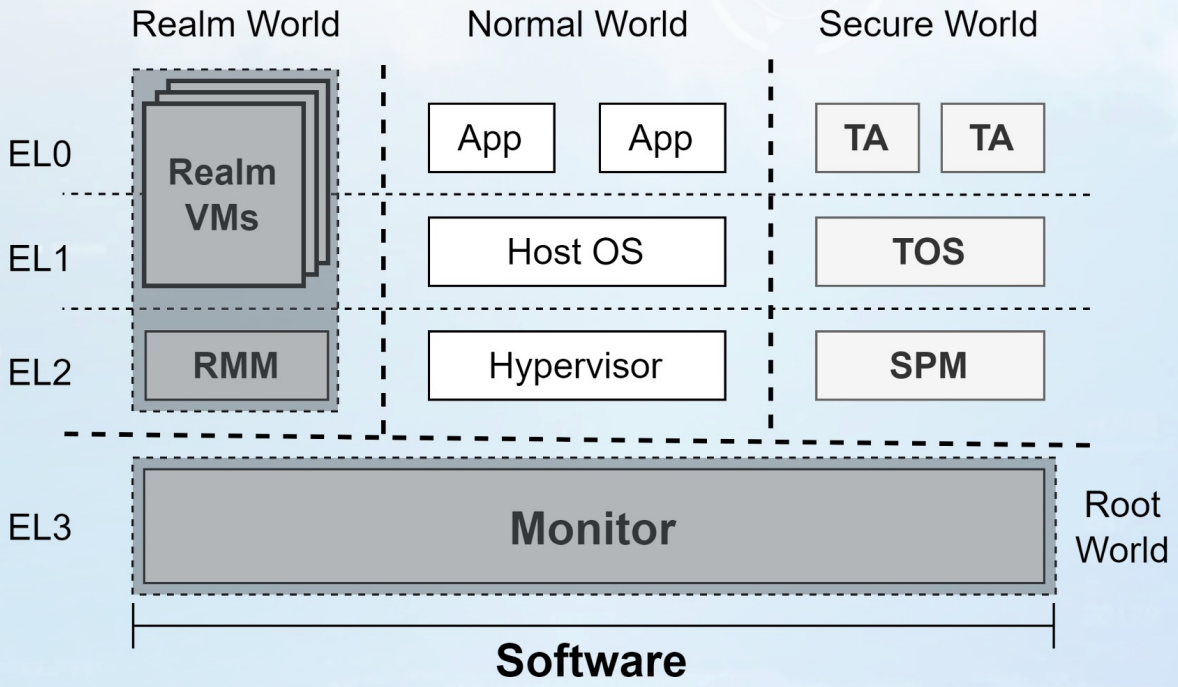
Internal solutions within Secure World cannot be isolated from the compromised TrustZone

3. TZASC

TrustZone hardware features (e.g., TZASC) are insufficient to protect a inspection system

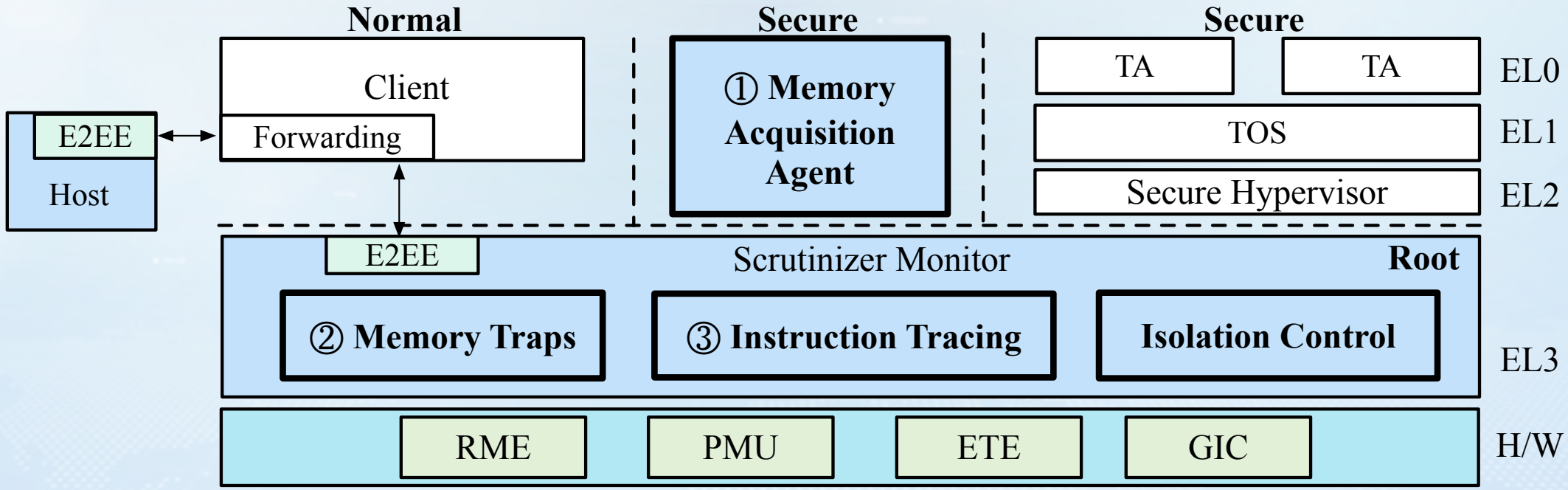
Arm Confidential Compute Architecture (CCA)

- CCA was announced in March 2021
  - Introduced as supplement to Armv9.2-A
- CCA introduces a set of new hardware features
  - New isolation boundaries for third party confidential computing (*Root and Realm Worlds*)
  - Dynamic assignment of memory to different worlds (*Granule Protection Check, GPC*)



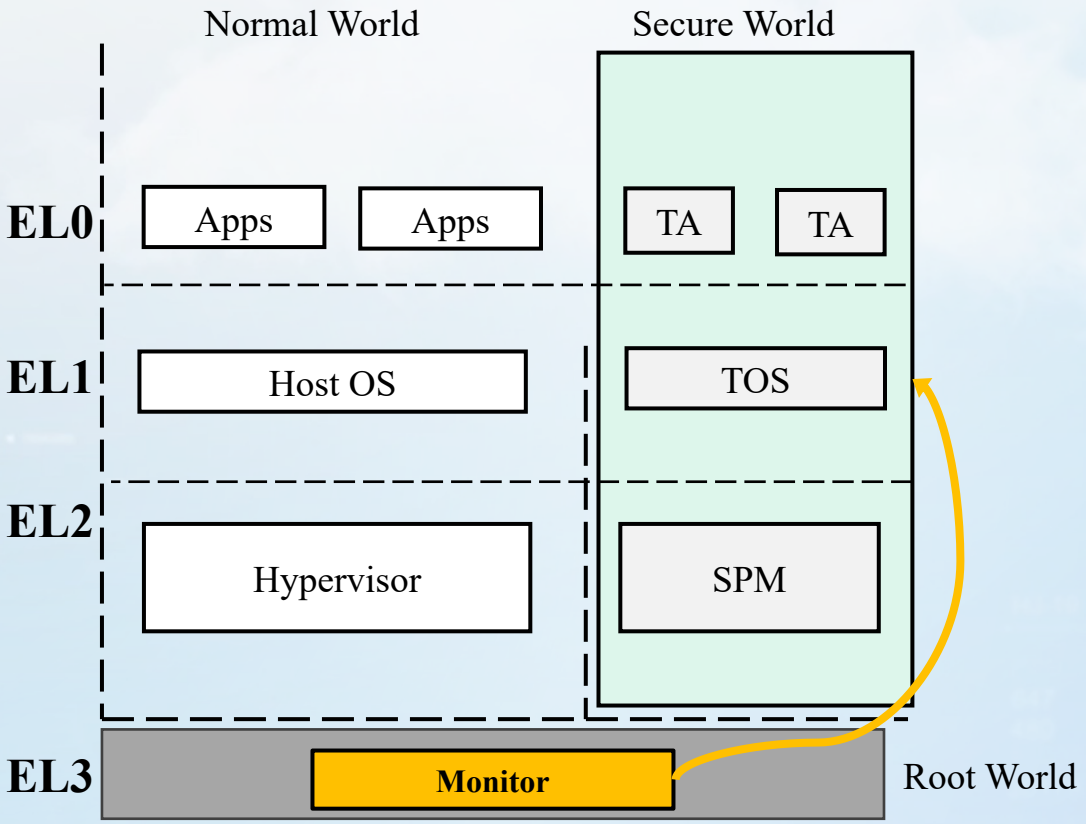
Towards Secure Forensics on Compromised TrustZone

- **SCRUTINIZER** based on Arm CCA Platform
  - Targeting the TrustZone software in Secure World
- SCRUTINIZER *Monitor* in the EL3 *Root World* protecting the components from compromised TrustZone



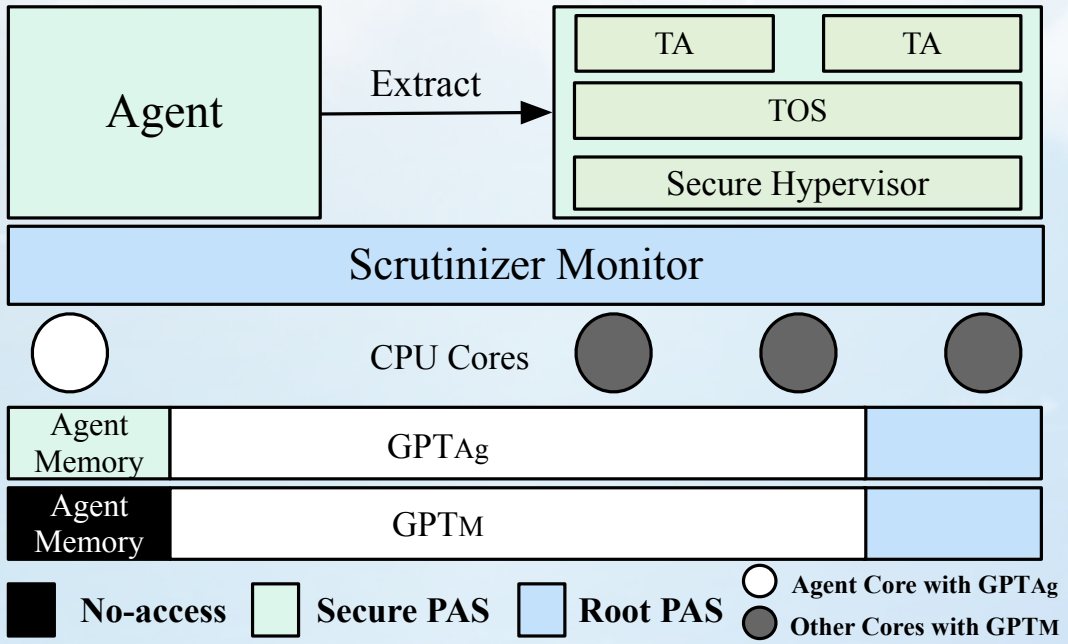
Memory Acquisition in Root World?

- Challenge 1: Memory acquisition in the Root World *enlarges* the Root World *codebase.*
- Challenge 2: Acquiring Secure World memory from the Root World is *slower than native access.*



Solution 1: Memory Acquisition with TCB Optimization

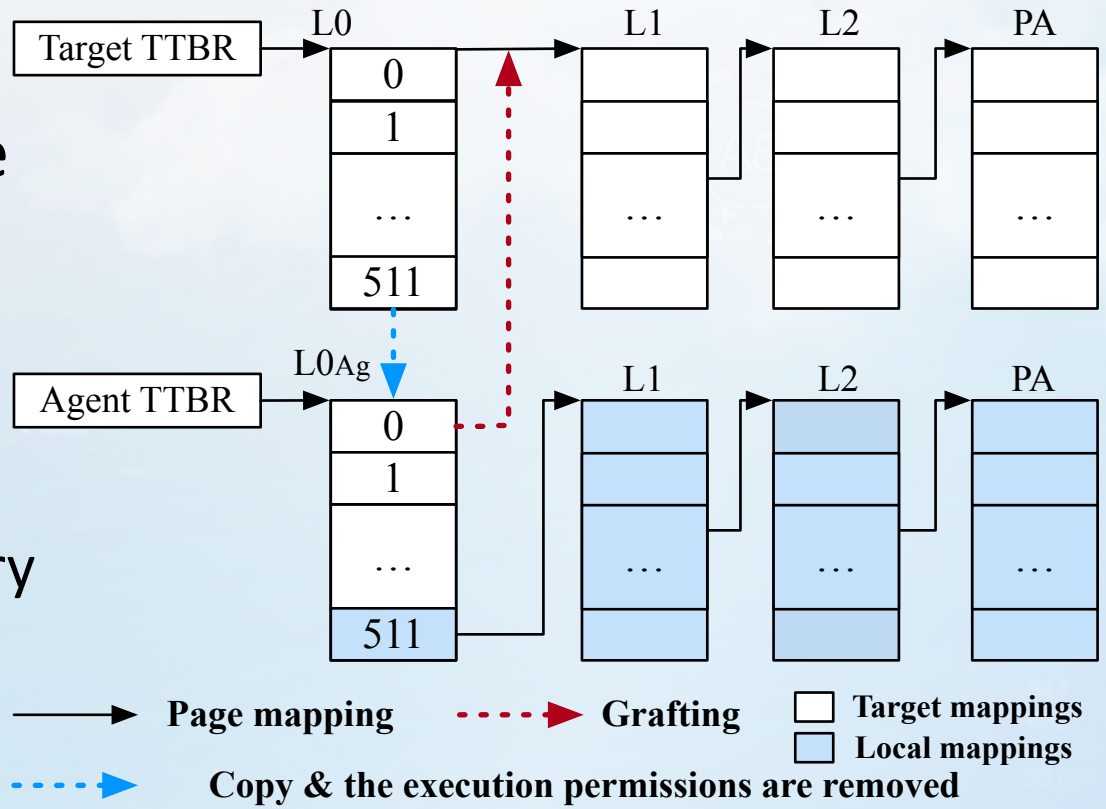
- **Codebase Reduction:** Decouple the memory acquisition functionality from the Monitor and integrate it into an agent
  - Reduce the expanded codebase of Monitor
  - Ensure that the Root World's size does not grow with the agent's code
- **Isolation Control for Agent:** Establish an agent execution domain within Secure World via dual-GPTs isolation
  - Ensure that the agent executes within Secure World yet remains isolated from compromised TrustZone systems



Solution 2: Memory Acquisition with Performance Optimization

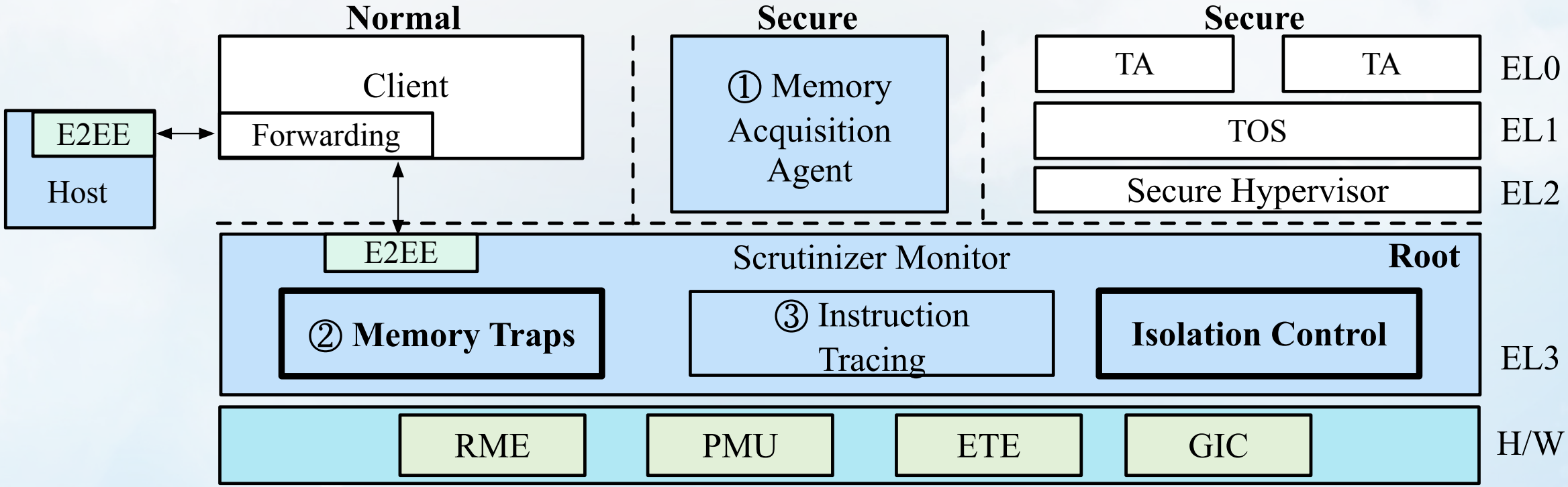
Grafting Mechanism: Copy the first-level page table, i.e., the target's L0 table, to the agent's memory, and directly graft the remaining levels

- Agent's mappings are inaccessible to TrustZone systems
- Enable efficient access to the target memory without building additional operations (VA_TZ → PA_TZ → VA_Ag)



Since agent run in the Secure EL1/EL2, enabling it to have the capability to directly use the TrustZone virtual address space (VA_TZ) for reading memory (infeasible at EL3 Monitor)

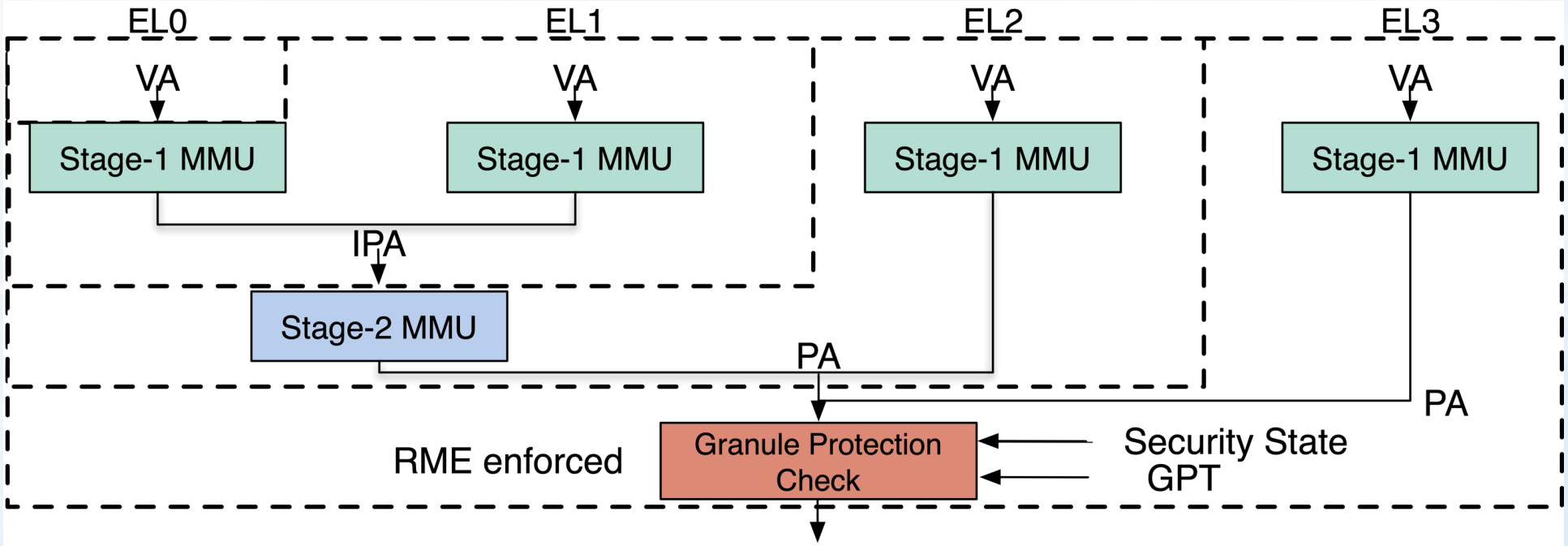
Memory Access Traps



Three secure forensic functions with several standard hardware features (RME, PMU, ETE, GIC):

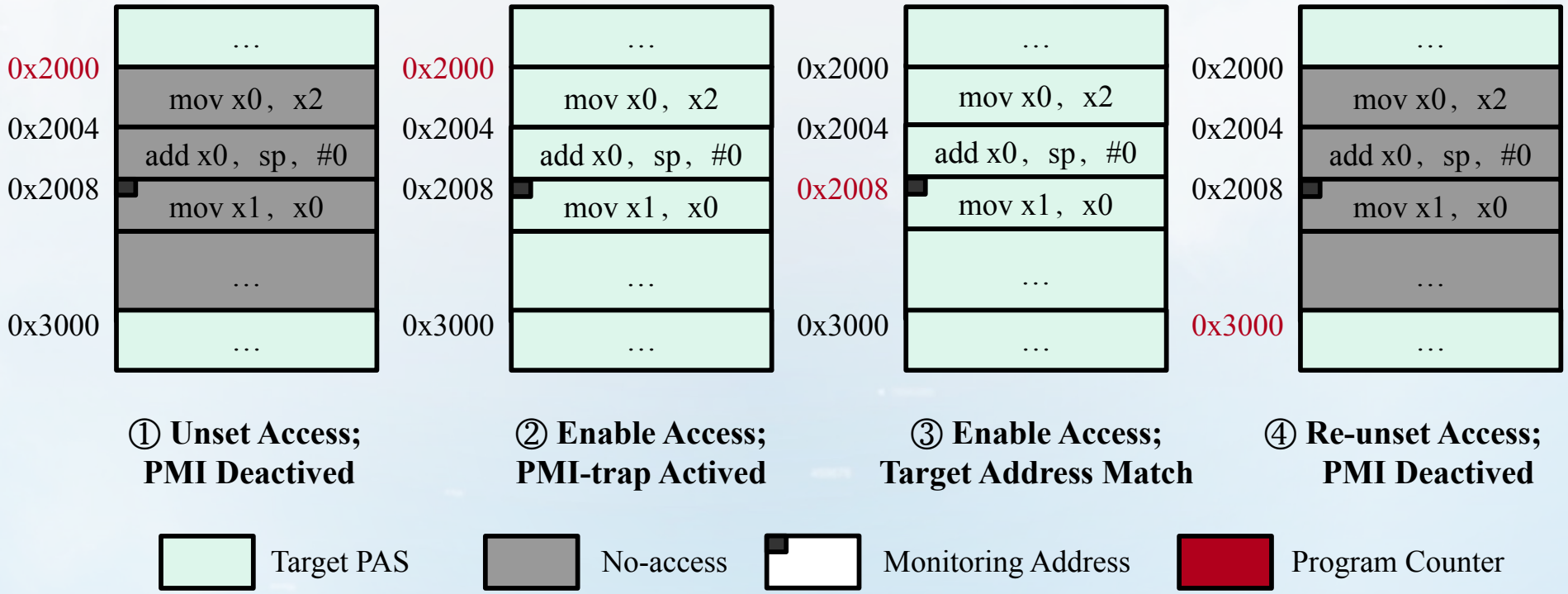
① Memory Acquisition; ② **Memory Access Traps**; ③ Instruction Tracing

GPC-based Memory Access Traps



- When RME-enforced GPC verification fails, a Granule Protection Fault (GPF) is generated to prevent unauthorized access. This fault can be rerouted to the EL3 Root world.
- *How to provide fine-grained memory access traps?*
  - The granule protection information (GPI) of GPT corresponds to a page (typically 4KB) is coarse.

Fine-grained Memory Access Traps



- Fine-grained improvement: Leverage PMU and GIC hardware features to enhance GPC-based traps to instruction-level granularity
- Isolation control for PMU and GIC: MMIO isolation and system register restriction

Evaluation

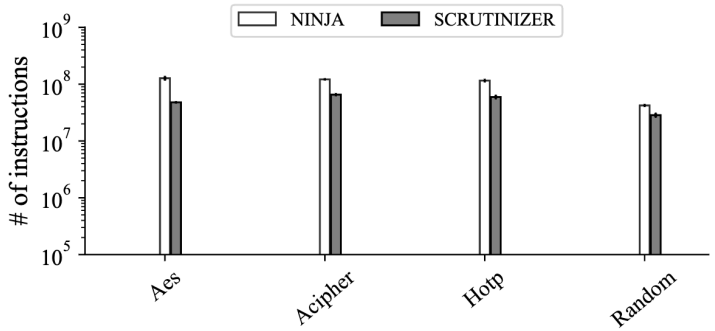
- Functional Prototype: Arm FVP Base RevC-2xAEMvA with RME enabled
- Performance Evaluation based on the Armv8 Juno R2 Board with GPT-analogue + FVP instruction counts

Compared to EL3-based memory acquisition (NINJA*), SCRUTINIZER is improved by 20x

SCRUTINIZER's memory trap overhead is reduced by 49.5%

TABLE IV: Acquisition performance comparison in time cost with different memory size.

# of Size	Kernel			Hypervisor		
	Vanilla	SCRUTINIZER	NINJA	Vanilla	SCRUTINIZER	NINJA
4KB	1.63 µs	4.90 µs	27.90 µs	1.60 µs	3.65 µs	27.88 µs
256KB	77.20 µs	82.60 µs	1.76 ms	77.41 µs	81.72 µs	1.76 ms
512KB	142.80 µs	146.20 µs	3.52 ms	135.12 µs	144.80 µs	3.52 ms
1MB	326.60 µs	334.20 µs	7.05 ms	323.61 µs	333.11 µs	7.05 ms
4MB	1.20 ms	1.25 ms	28.21 ms	1.21 ms	1.24 ms	28.20 ms
16MB	5.46 ms	5.51 ms	112.81 ms	5.38 ms	5.46 ms	112.80 ms



*NINJA: Towards Transparent Tracing and Debugging on ARM, USENIX Security 2017

Conclusions

- SCRUTINIZER
  - Provides a secure forensics framework for compromised TrustZone
  - Leverage the hardware features of Arm CCA to create an isolated forensic environment
  - Optimizes the TCB and performance
  - Ensures platform compatibility

Source Code Available at https://github.com/Compass-All/SCRUTINIZER