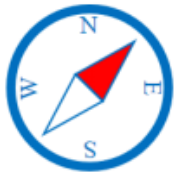
MOLE: Breaking GPU TEEs with GPU Embedded MCU

Hongyi Lu*, Yunjie Deng*, Sukarno Mertoguno
Shuai Wang, Fengwei Zhang



COMPASS Lab
COMPuter And System Security Lab

SUSTech Southern University of Science and Technology

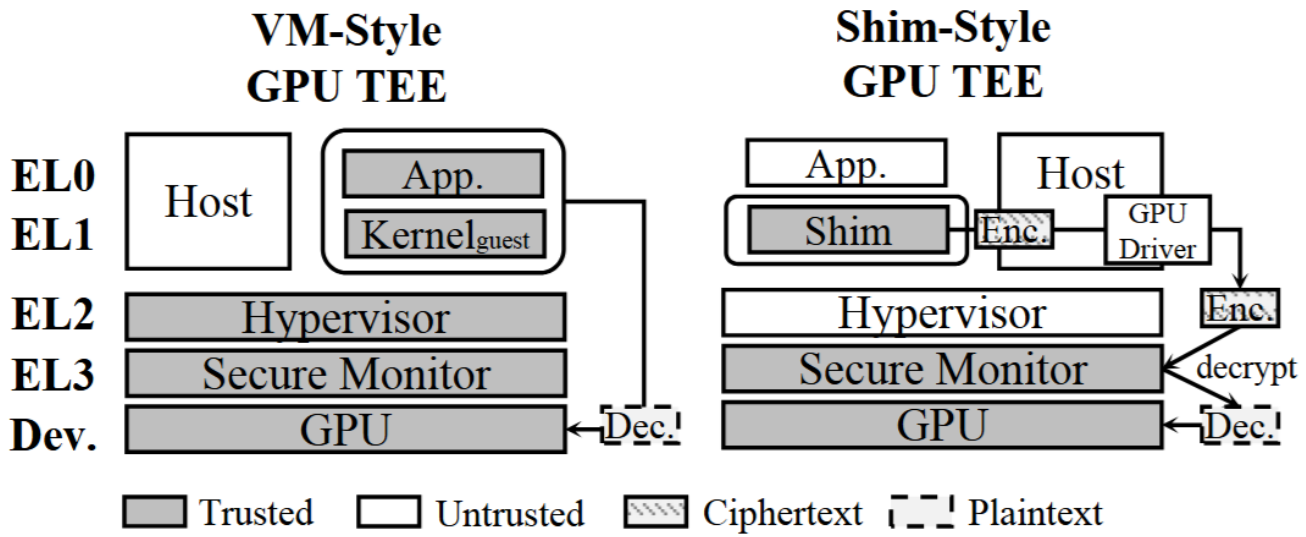
Georgia Tech

GPU Trusted Execution Environment

- Protect GPU data/program from **high-privileged adversaries** (e.g., OS, VMM). Before H100, these were implemented with **CPU-side primitives** like SGX, TrustZone…

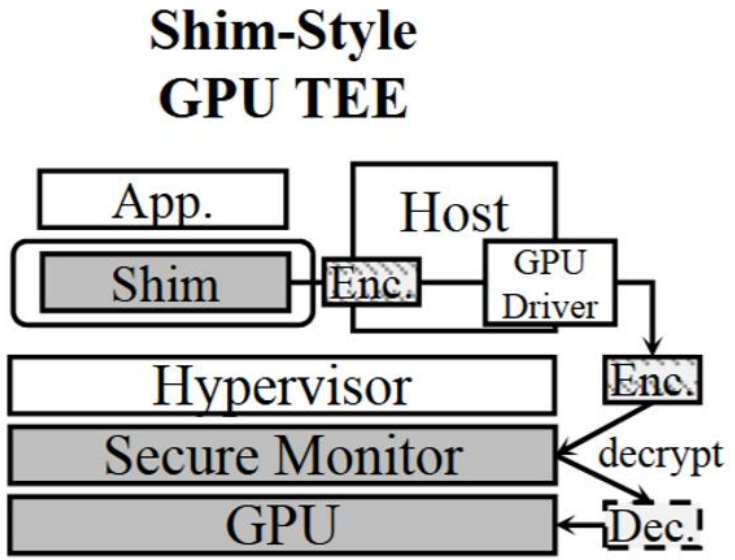
- **VM-style GPU TEE:**
  - All-in-one VM
  - Large TCB
  - NVIDIA H100, ACAI (Sec 24)

- **Shim-style GPU TEE:**
  - Protects essential components
  - Small TCB
  - StrongBox (CCS 22), CAGE (NDSS 24), MyTEE (NDSS 23)



Shim-style GPU TEE

- **Secure:**
  - Data is either encrypted or isolated
- **Low TCB:**
  - Only security-critical ∈ TCB



Q: Do the **security-unrelated** parts really have **NO security implications?**

A: Wrong, GPU has a hidden MCU!

①: *GPU Ctrl* ②: *Memory* ③: *Interrupt* ④: *Scheduler* ×: *None.*

GPU TEE	GPU Mgmt.	S-Task	NS-Task
StrongBox	×	①②③④	④
CAGE	×	①②③④	④
MyTEE	×	①②③	×

Mali GPU MCU

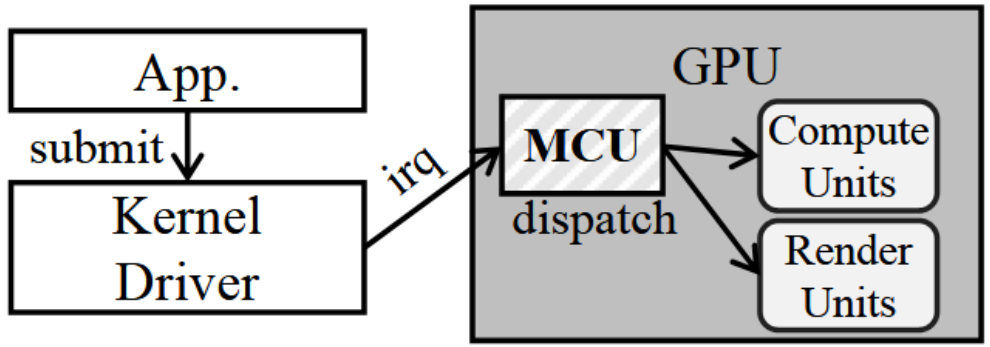
MCU Info:

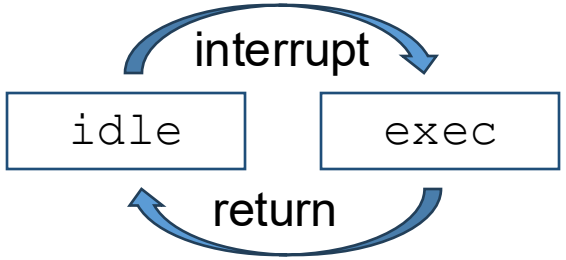
- Scheduling, resource allocation
- **Viewed as part of the GPU**

Basic Info:

- Cortex-M ISA
- Event-driven execution

Serve as the **bridge** between the GPU / Kernel





Mali GPU MCU

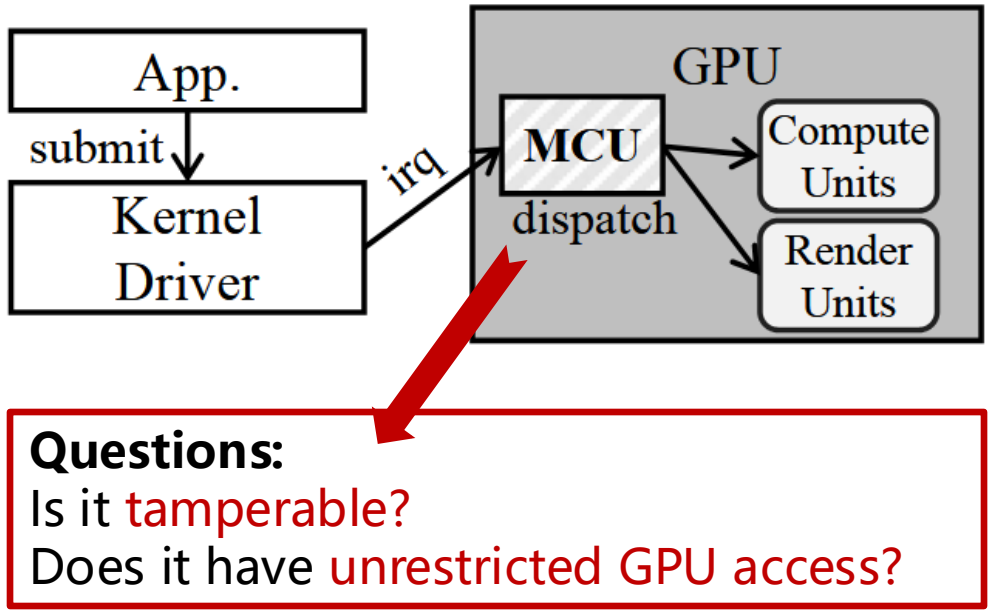
MCU Info:

- Scheduling, resource allocation
- Viewed as part of the GPU

Basic Info:

- Cortex-M ISA
- Event-driven execution

Serve as the **bridge** between the GPU / Kernel



Weaponize MCU

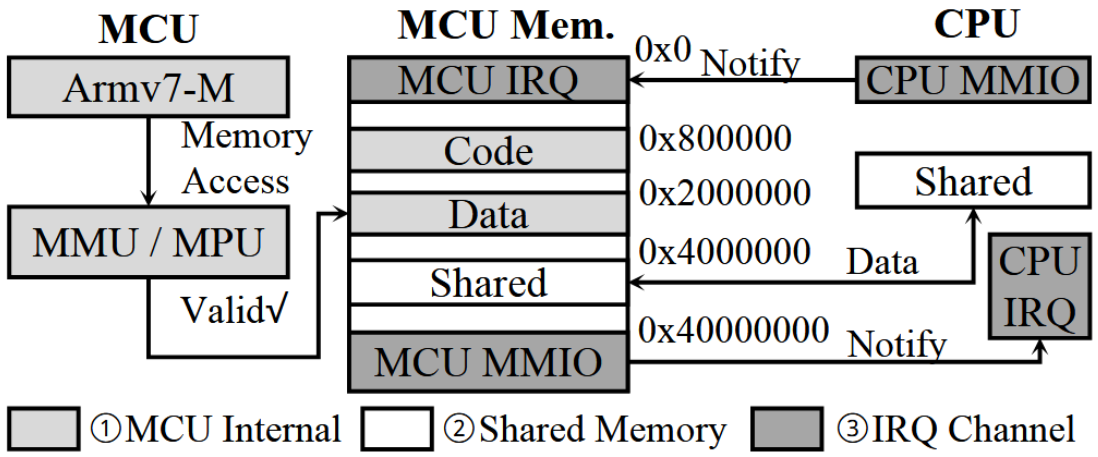
Reverse-engineering shows the MCU:

- uses multiple interrupts:

  `irq#0: t_submit irq#2: t_complete`

- has an MPU/MMU

  (can be disabled from kernel/firmware)



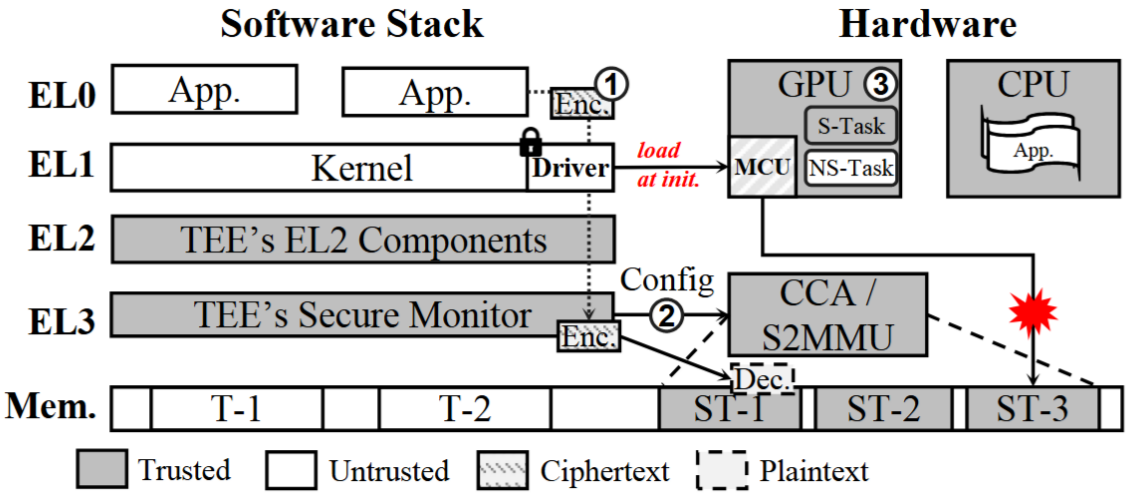
Experiments:

- Write `0xdeadbeef` to normal memory at `reset` **NO VERIFICATION!**
- Write `0xdeadbeef` to TEE-protected memory at `reset` **NO PROTECTION!**
- Trigger malicious code at `task_submit` **EXPLOITATION!**

Revisit GPU TEE Workflow

1. Encrypt the data to pass through untrusted driver components
2. Protect the data via S2MMU/TrustZone upon decryption
3. Protected plain-text data is processed by GPU

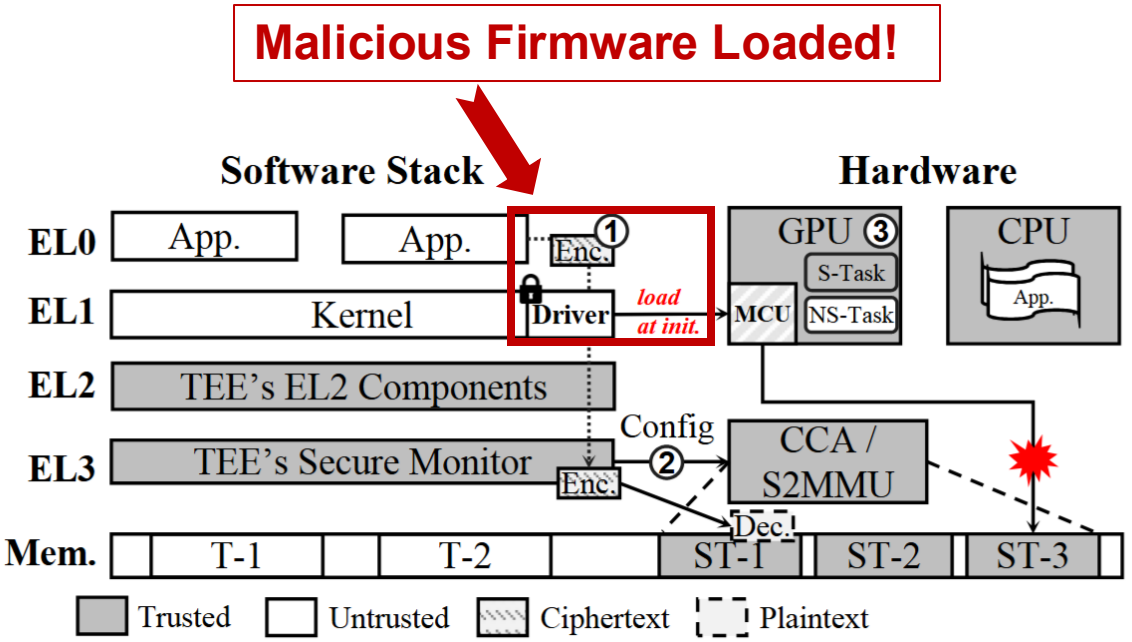
So, is the data either protected or encrypted?



Revisit GPU TEE Workflow

1. Encrypt the data to pass through untrusted driver components
2. Protect the data via S2MMU/TrustZone upon decryption
3. Protected plain-text data is processed by GPU

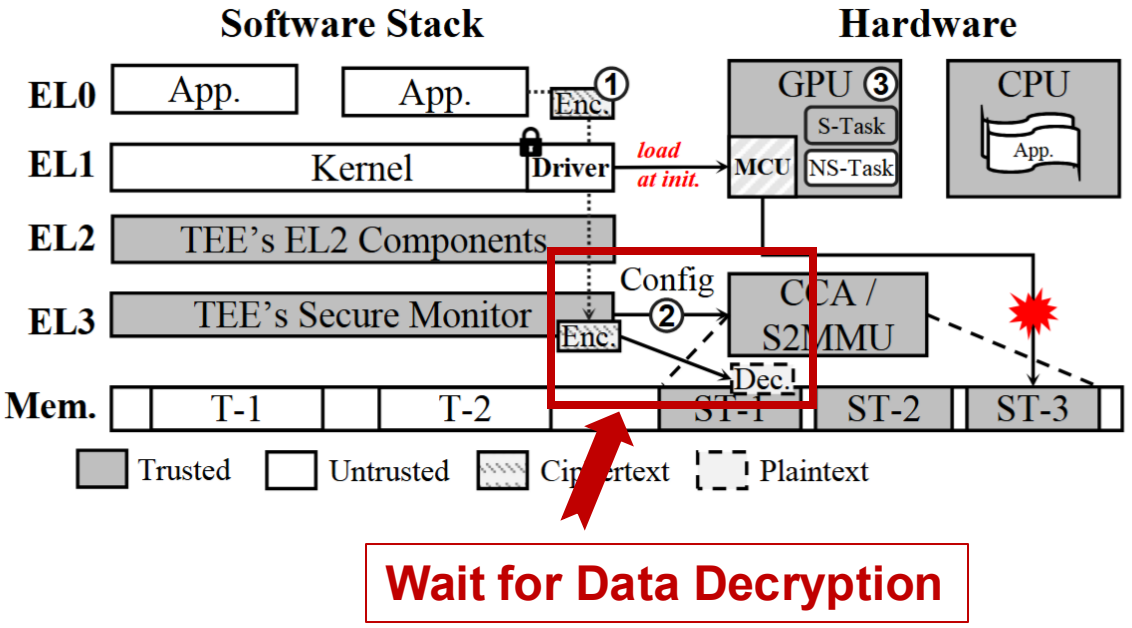
Data is either protected or encrypted, right?



Revisit GPU TEE Workflow

1. Encrypt the data to pass through untrusted driver components
2. Protect the data via S2MMU/TrustZone upon decryption
3. Protected plain-text data is processed by GPU

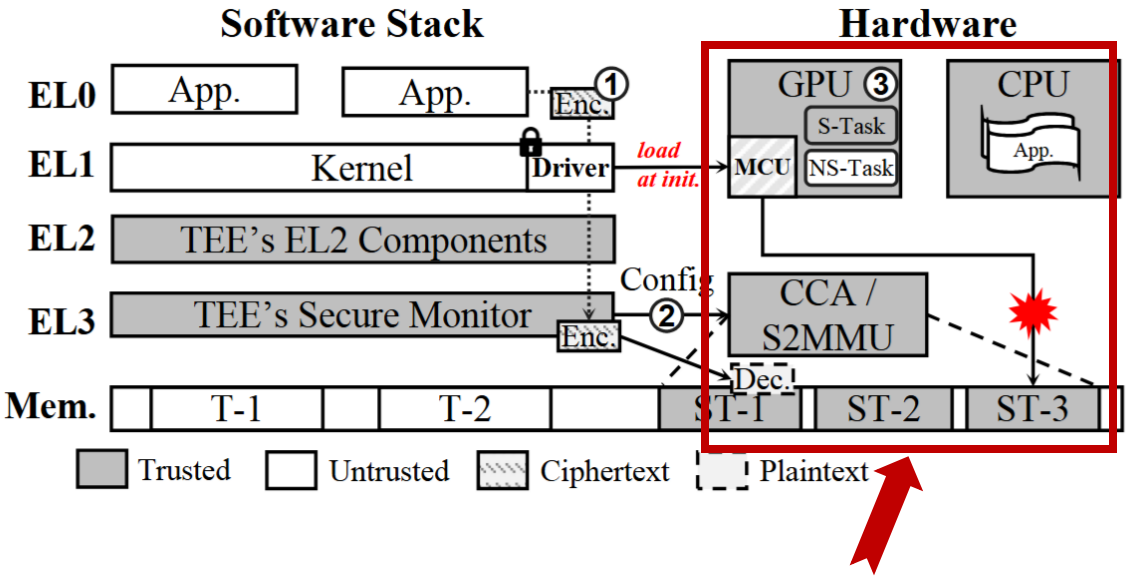
Data is either protected or encrypted, right?



Revisit GPU TEE Workflow

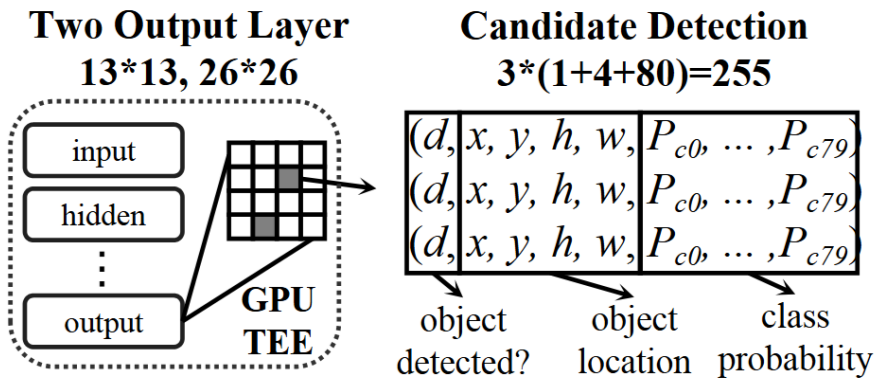
1. Encrypt the data to pass through untrusted driver components
2. Protect the data via S2MMU/TrustZone upon decryption
3. Protected plain-text data is processed by GPU

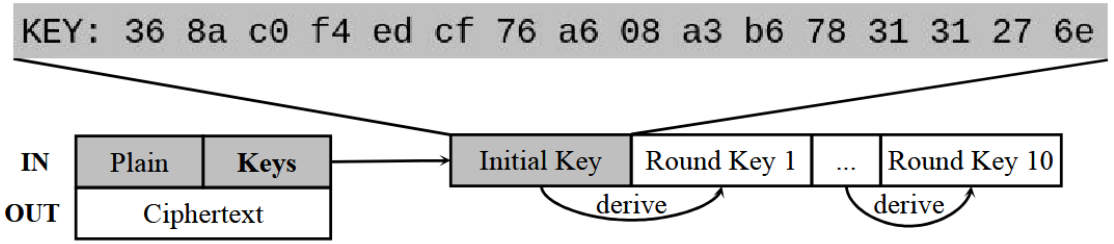
Data is either protected or encrypted, right?

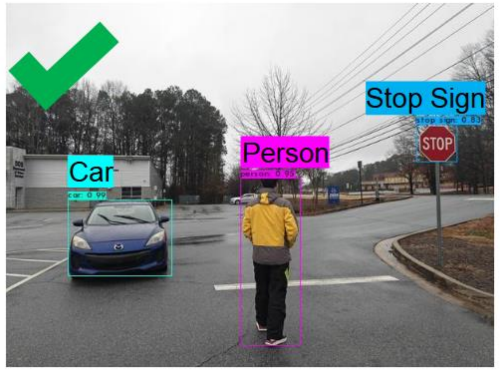


Direct Access to Protected Memory

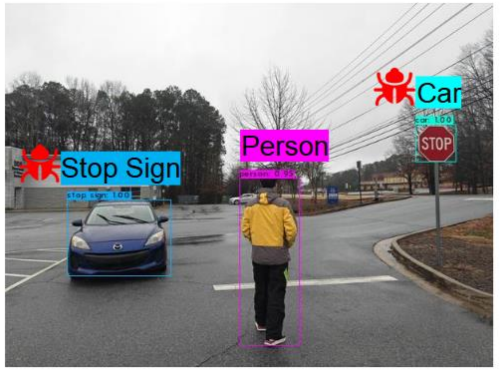
Demonstration





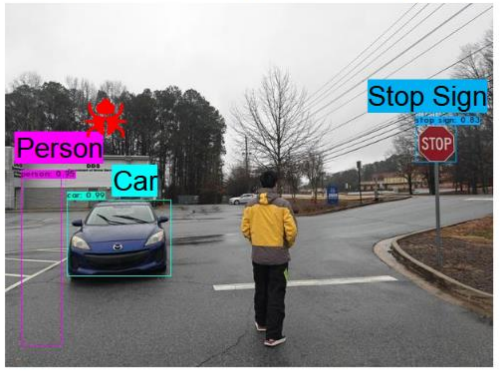


(a) Original result.



(b) Car and sign exchanged.

(c) Person and sign removed.



(d) Person at wrong location.

Acknowledgements

Dear [redacted]

We appreciate the acknowledgment.

The findings seem interesting and may require MyTEE to adopt additional measures to monitor (lock) MCU firmware updates during device runtime.

We look forward to seeing the details in the published paper.

Thanks,

Jinsoo Jang.

Hi [redacted]

Thanks for your responsible disclosure file. I am interested in the MCU-based vulnerability and the related attacks to our GPU TEE works, StrongBox, and CAGE.

Currently, in StrongBox/CAGE, we assume the GPU and its firmware are trusted. Such assumption is guaranteed by secure boot and attestation works. However, the MCU-based attacks find a new way to bypass it and spy the confidential GPU applications running on the GPU. Thus, StrongBox/CAGE (and possibly other GPU TEEs with an untrusted driver) should introduce additional protection for GPU devices/firmware to mitigate MCU-based attacks.

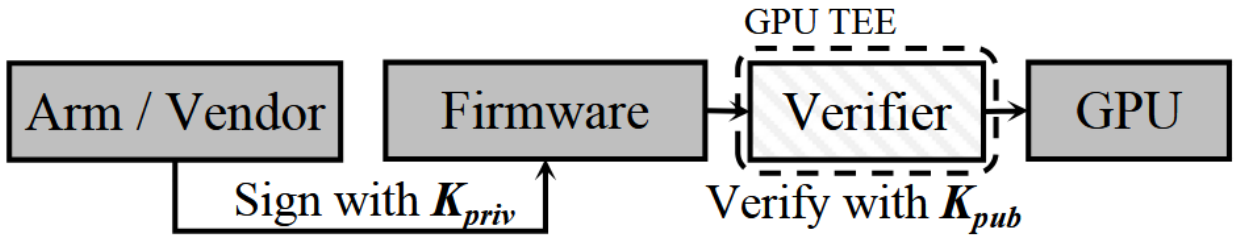
Please feel free to contact me if you have any questions. I am glad to discuss the MCU-based vulnerabilities/attacks with you.

Sincerely,
WANG Chenxu

Dilemma in Firmware Verification

Goals:

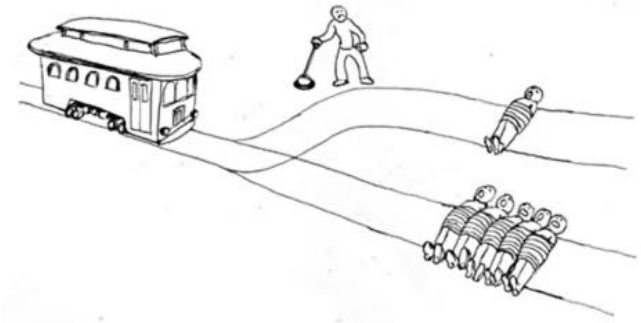
- Verify the firmware
- Ensure firmware integrity



Verification must be in GPU TEEs

TCB Dilemma:

- Firmware-related: 60 KLoC
- Shim-style GPU TEE: 5 KLoC



Beyond TEEs, Firmware Supply Chain of Arm

Problem:

- No verification
- No way to audit (closed source)

Unreliable Supply Chain

1. Arm
2. Vendors with NDA
3. Developer / User

Vendor	Usage	Distribute via	Verified	#Ref.
Google	Mobile Phone Tablet	Bundled	✘	0
MediaTek	Mobile Phone Tablet	Bundled GitHub	✘	2
HiSilicon (Huawei)	Mobile Phone	Bundled	✘	0
Rockchip	IoT Device	GitHub GitLab	✘	**41**

Some vendors simply put their firmware on GitHub (vulnerable to phishing attack)

Beyond TEEs, Firmware Supply Chain of Arm

Dear [redacted],

Apologies for the delay in getting back to you on this, we' ve had several meetings internally to investigate how we (and the industry as a whole) might tackle the issue of signing.

Arm currently distributes firmware to partners via private secure channels accessible only by approved licencees, as such the attestation is provided by the delivery mechanism itself. If end users wish to download arbitrary firmware for use on their devices from third parties then those third parties should provide a mechanism for attestation.

The Arm maintained public repositories containing Mali firmware (and the linux-firmware tree, currently) are limited firmwares which are functional only with very specific hardware and driver releases provided for partners to conduct testing; however, we plan to begin providing checksums (e.g., md5 or SHA256) for any binary image made available outside of our secure partner delivery hub soon, hopefully with the next releases.

Let me know if you have any questions or concerns.

Thanks,
Aaron
Arm PSIRT

Other Platforms

- **NVIDIA: NV-RISCV**
  - GPU System Processor (GSP):
    - Scheduling, resource management
  - Secure Processor:
    - Confidential computing, firmware verification
  - NVIDIA did not release many details on these two MCUs

- **AMD: MES**
  - Micro Engine Scheduler (MES):
    - Scheduling, …
  - AMD releases MES documentation, but does not allow modified firmware

My Wife & Me

My Homepage

Email Me

Thank You!

I am on the market!

Contact me!