TRACES: TEE-based Runtime Auditing for Commodity Embedded Systems

Adam I. Caulfield[+], Antonio Joia Neto[+], Norrathep Rattanavipanon[*], Ivan De Oliveira Nunes[+]

[+]Rochester Institute of Technology; [*]Prince of Songkla University, Phuket Campus

ACSAC 2024
December 9-13, 2024 · Honolulu, Hawaii, USA

1. Introduction
2. Background
   - Remote Attestation
   - Control Flow Attestation
   - Runtime Auditing
3. TRACES
4. Evaluation
5. Conclusion

Embedded devices

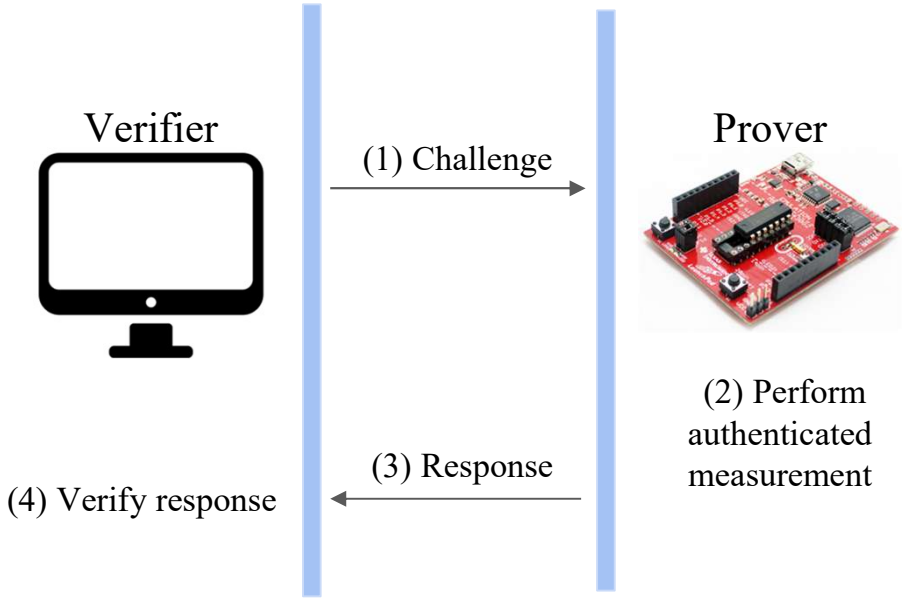
- Low-cost, energy efficient MCUs
  - Limited processing power, memory size , memory protection, etc ..
- Resource constrained impact security features
- Execute safety-critical tasks in modern systems

Remote Attestation (RA)

- A **security mechanism** designed to verify the integrity of embedded devices.
- Ensures that the device's software state is untampered and operating as expected.

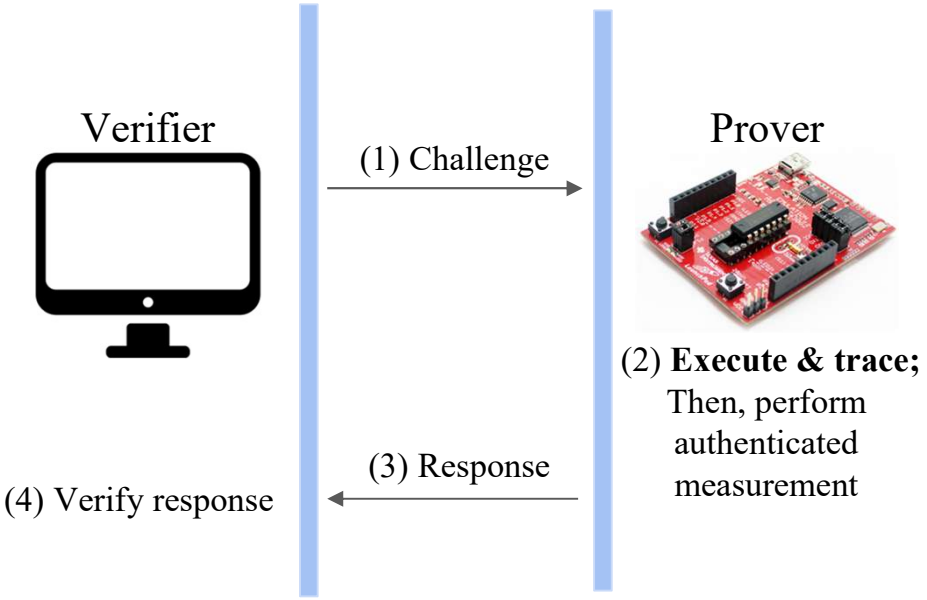
How it works :

- A trusted verifier (e.g., server or monitoring system) requests evidence of the device's integrity.
- The device (prover) generates a cryptographic response based on its current software state.
- The verifier checks the response to confirm the device is secure.



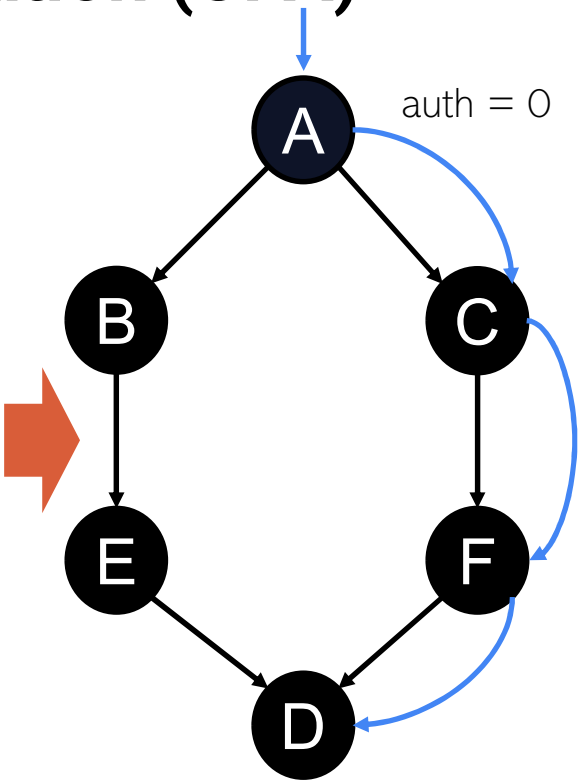
Control Flow Attestation (CFA)

- CFA extends RA to provide proof of runtime behavior of the application
- Prover records a trace of the control flow transfers (branches) during execution
- The trace and memory are measured to obtain proof about both static and runtime states



Control Flow Attestation (CFA)

```
A  if (auth)
B      do   function1();
C      else function2()
D  end
E  def function1(){…}
F  def function2(){…}
```



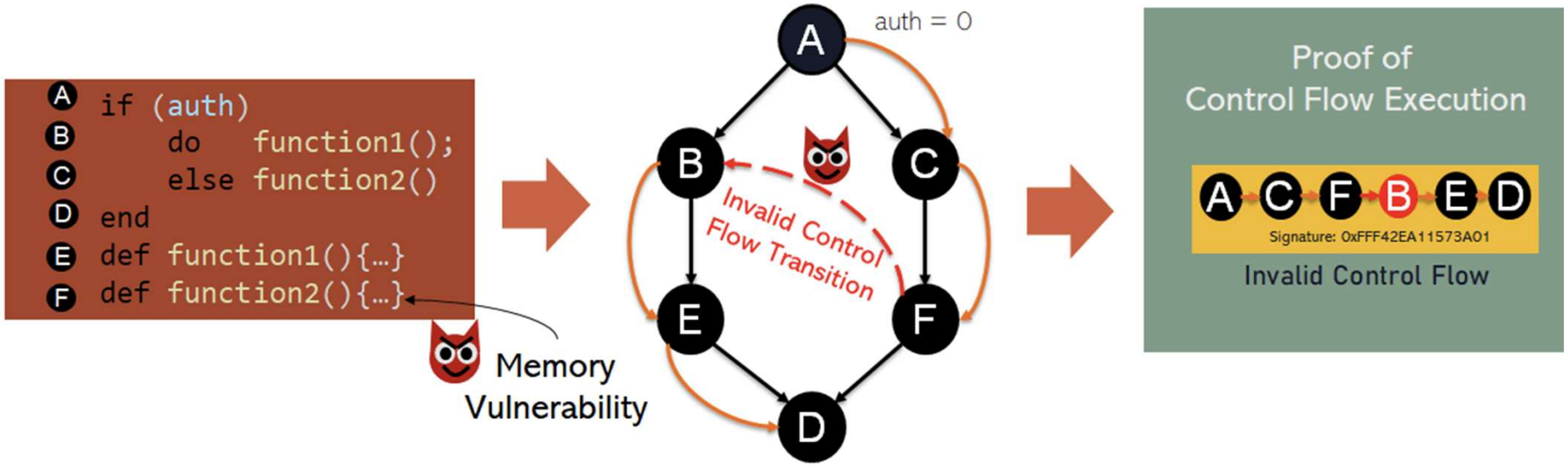
CFA

Secure Memory Region

Proof of Control Flow Execution (Evidence)

A → C → F → D

Control Flow Attestation (CFA)

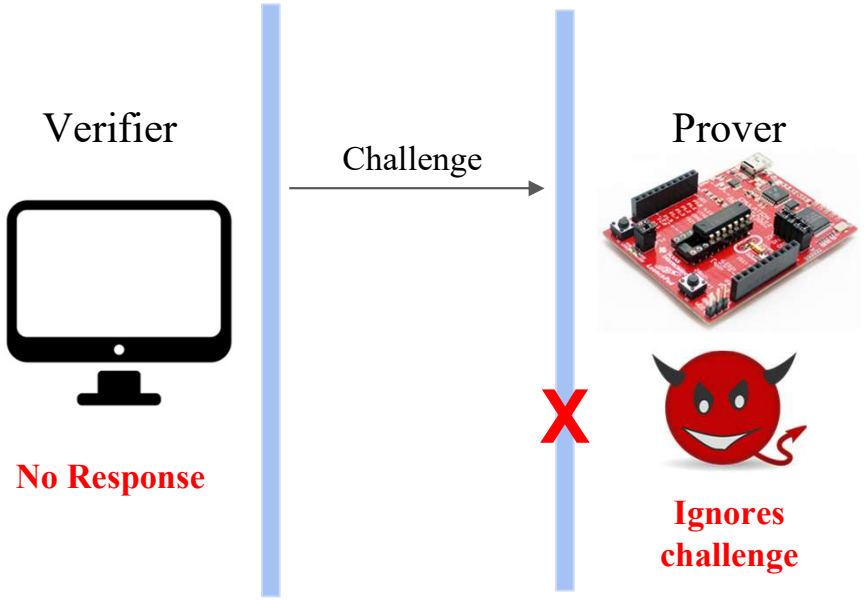


A challenge with *attestation*

- Attestation is a ***passive*** technique
- Compromised Prover may not participate in the protool
- No response → *detect* compromise

but…

- Cannot *obtain proof* of the exact malicious behavior

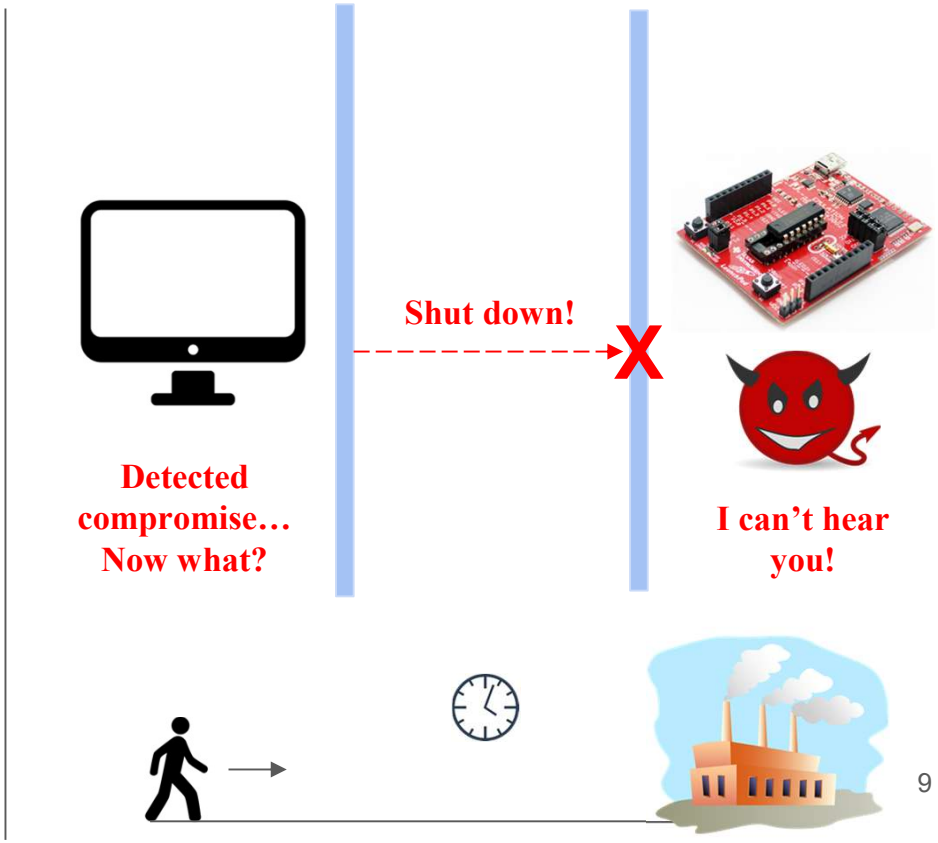


After detection…

- How to resolve compromises?

Usually:

- Reboot
  - Not guaranteed since adversary
- Physical intervention



Runtime auditing

- Guarantees runtime evidence is accurate/authentic
- Guarantee eventual delivery of runtime evidence to Vrf
  - Assuming eventual communication
- Remotely intervene after compromise detection

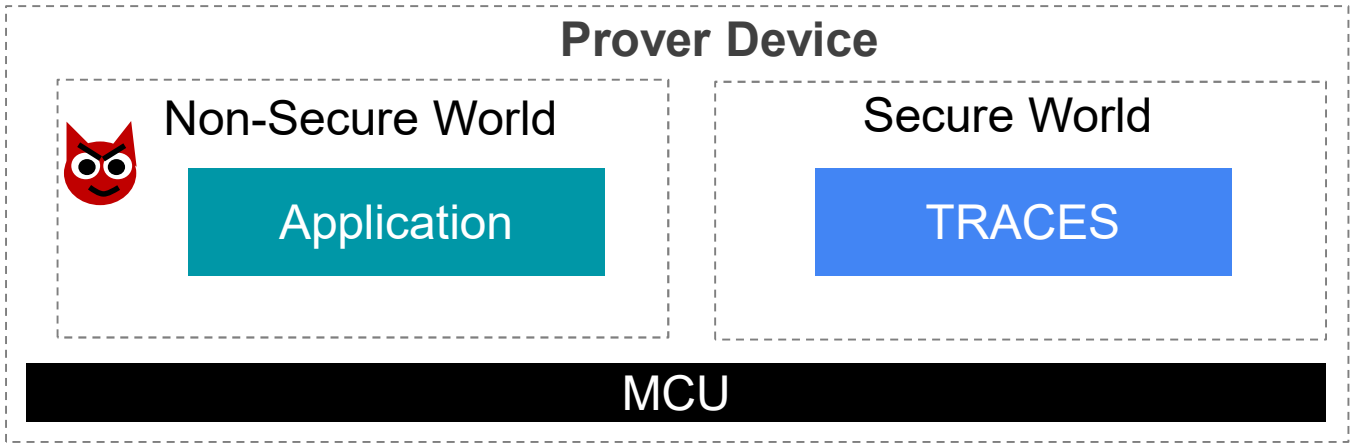
Current Models:

- Achieve Runtime Audition with hardware modifications

Our Contribution:

- First design realizing secure runtime auditing on off-the-shelf MCUs
  - Can be deployed in devices that are currently In the market

System Model

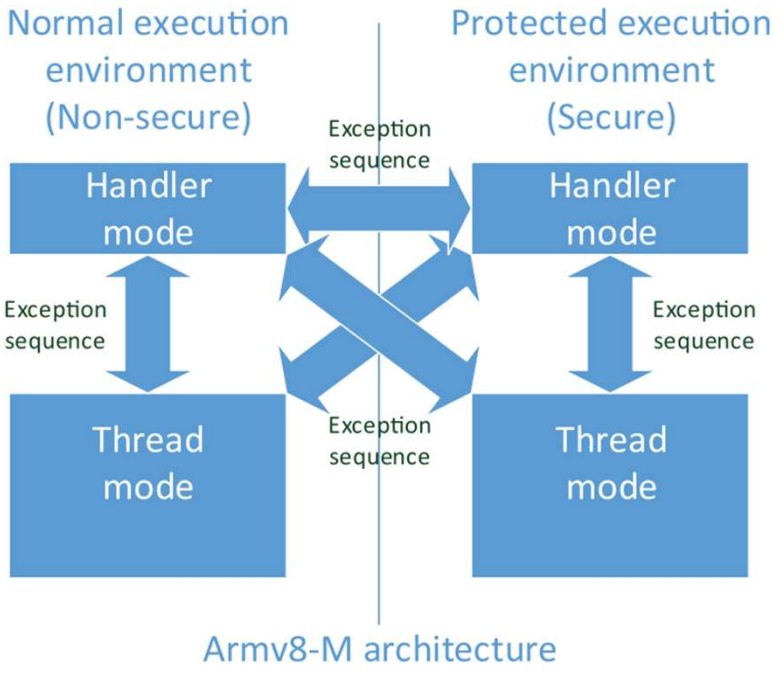


- Adversary has total control over the Non-Secure World

(including access to privilege mode, interrupt triggers, peripheral configurations)

TrustZone in ARM Cortex-M

- Security extension for ARM Cortex-M and -A profiles
- Splits memory into two security states
  - Secure and Non-Secure
- Peripherals and Memory can be assigned as Secure or Non-Secure by code executing in Secure state
- Non-Secure state cannot access or execute resources assigned to Secure state.
- Secure World always initialize before Non Secure World

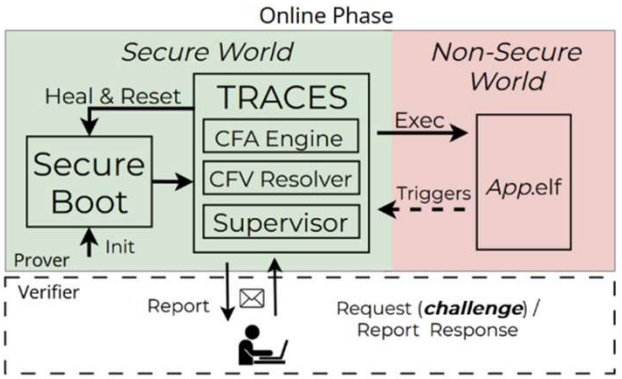


TRACES: Key Idea

1. Have a Secure World framework to achieve runtime auditing of the Non-Secure World application.
   a. Manage the execution and attestation of applications in the Non-Secure world

   b. Enforce Trigger generation/transmission of evidence from the Secure World.

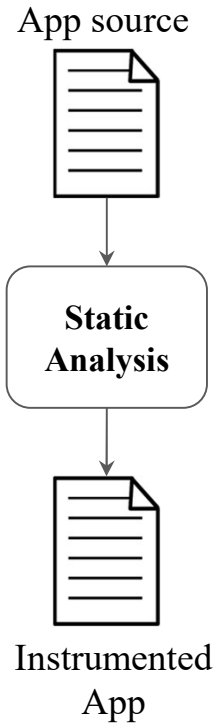
   c. Implement heal function in Secure World, and execute after Vrf responds.



TRACES Workflow: Verifier Offline phase

Control Flow Attestation Instrumentation :

- Binary analysis on the Application during compilation time
- Instrument Application with additional instructions to track its execution control flow
- Instrumented Application is deployed in the Non-Secure World



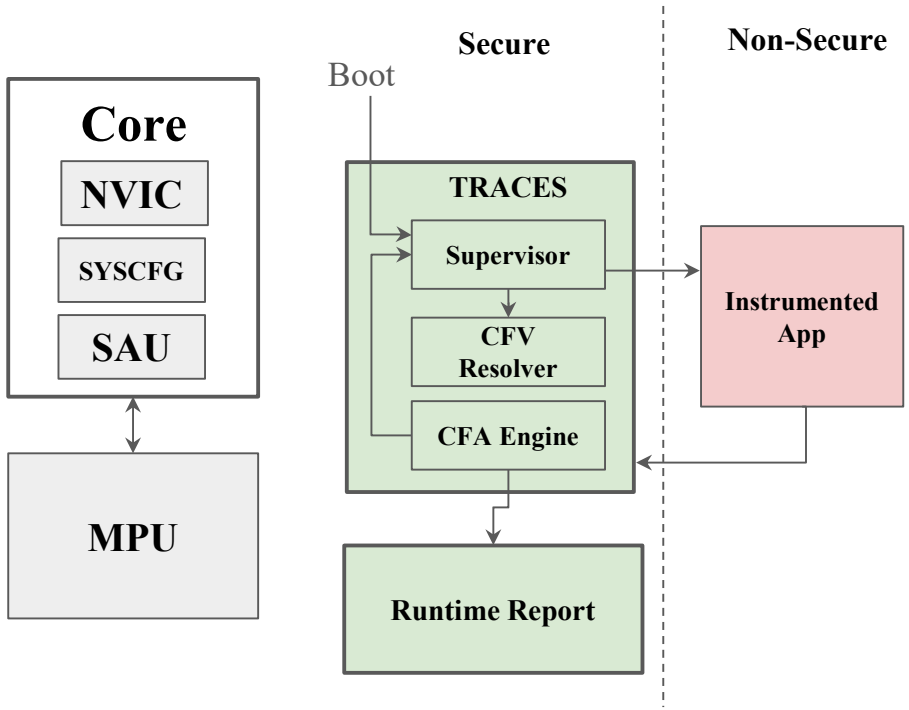
TRACES Workflow: Online phase

Three modules:

1. Supervisor:
   - Configurations & communication

2. CFA Engine:
   - Logging and report generation

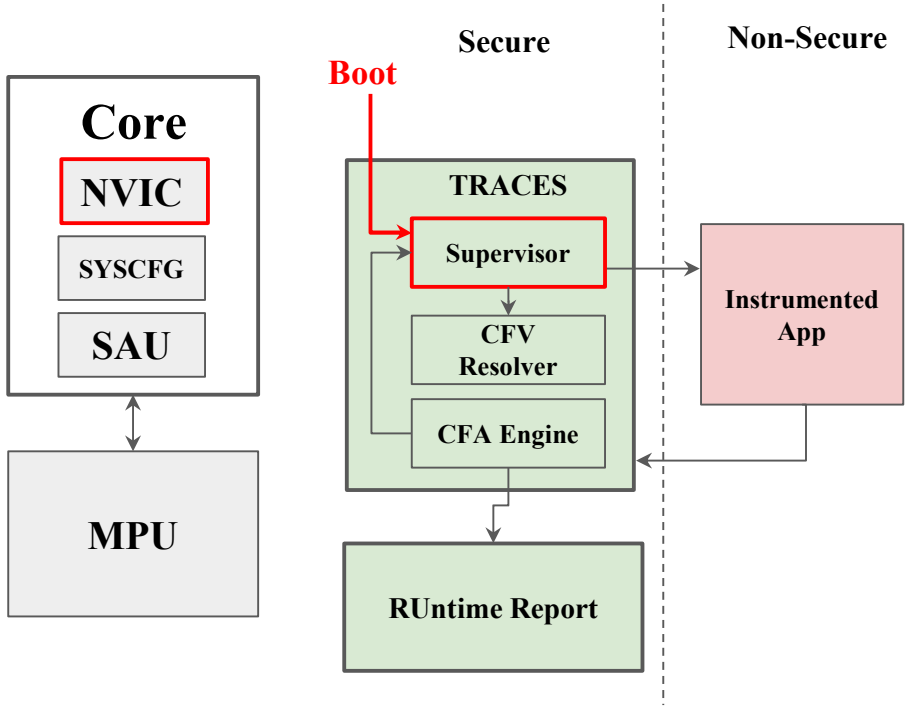
3. CFV Resolver:
   - Configurable healing action



TRACES Workflow: Online phase

Upon startup, Supervisor configures the system

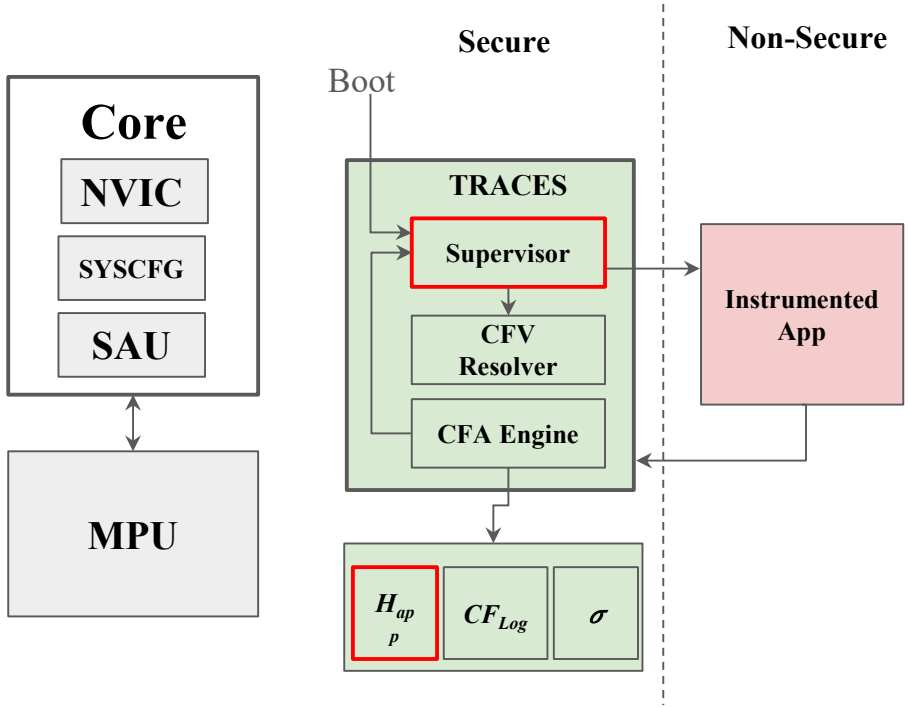
- Configures secure timer through NVIC as watchdog timer



TRACES Workflow: Online phase

Upon startup, Supervisor configures the system

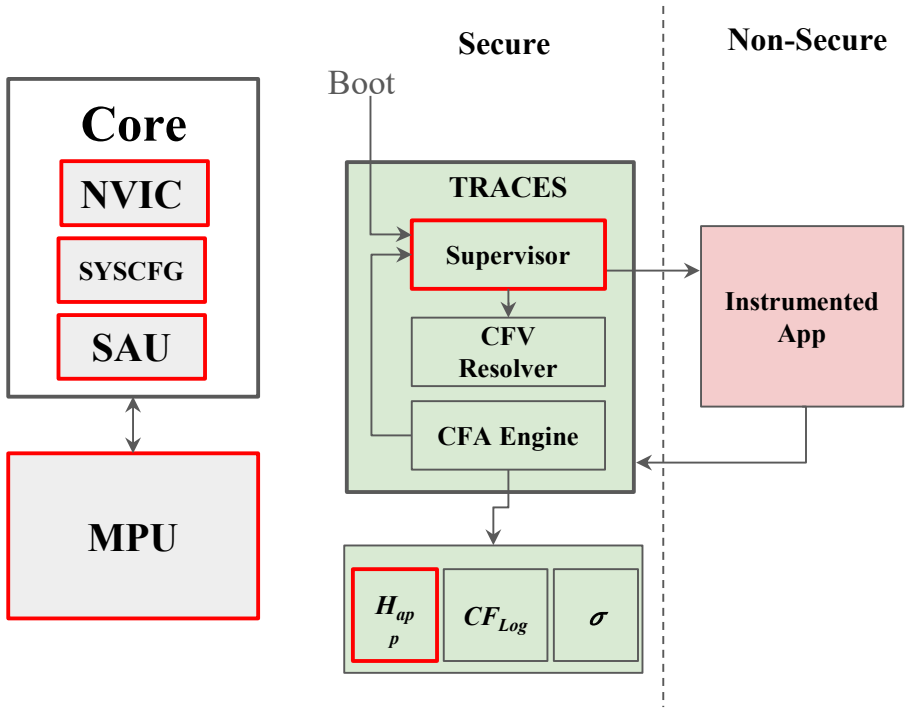
- Configures MPU so that:
  - App data is non-executable
  - App code is immutable
- Revokes access to MPU through SYSCFG and SAU
- Deactivate interrrupts in the Non Secure World
- Measures state of App (hash)
- Waits for an attestation request



TRACES Workflow: Online phase

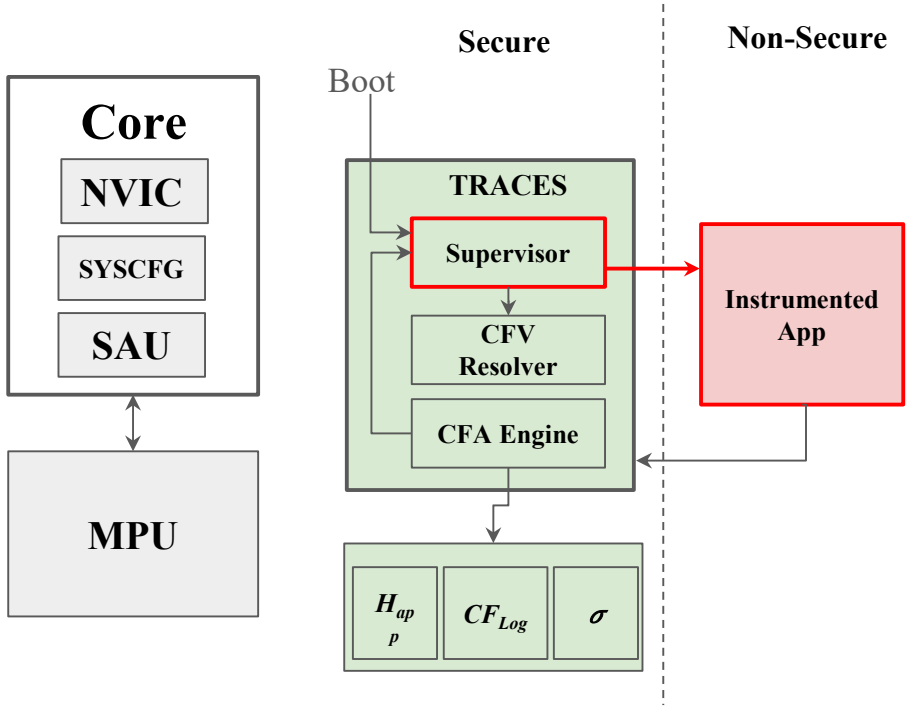
With these configurations, compromised Non-Secure World cannot tamper with:

1. The secure watchdog
2. Binary of the Instrumented App
3. Non Secure Interrupts



TRACES Workflow: Online phase

Upon receiving an authentic challenge, the Supervisor starts running App

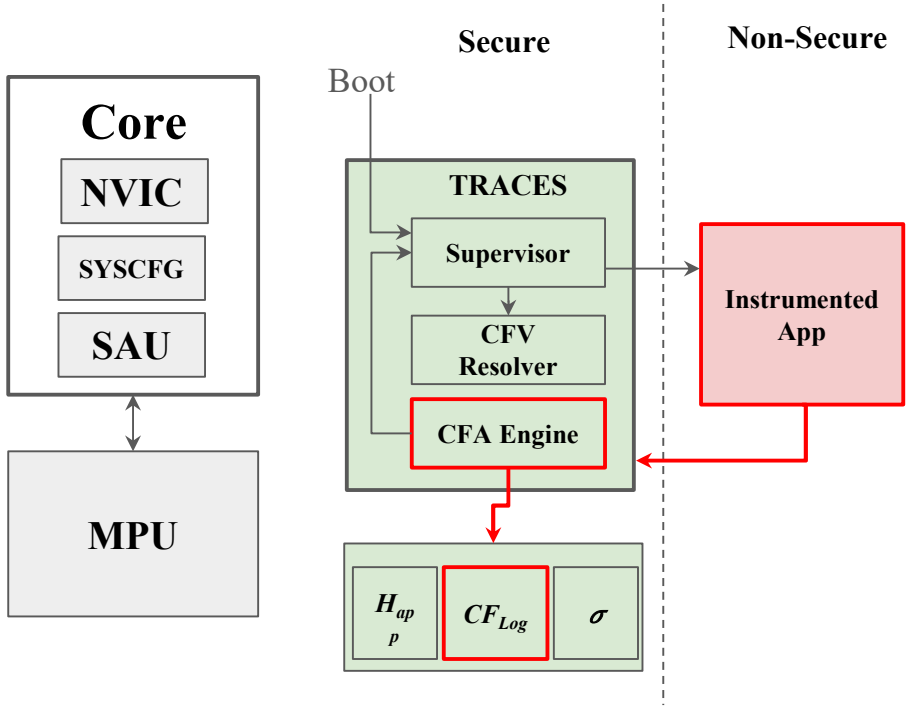


TRACES Workflow: Online phase

Upon receiving an authentic challenge, the Supervisor starts running App

Instrumented instructions in App call CFA Engine at branches

CFA Engine writes the destination address of the branch to the CF_{Log}

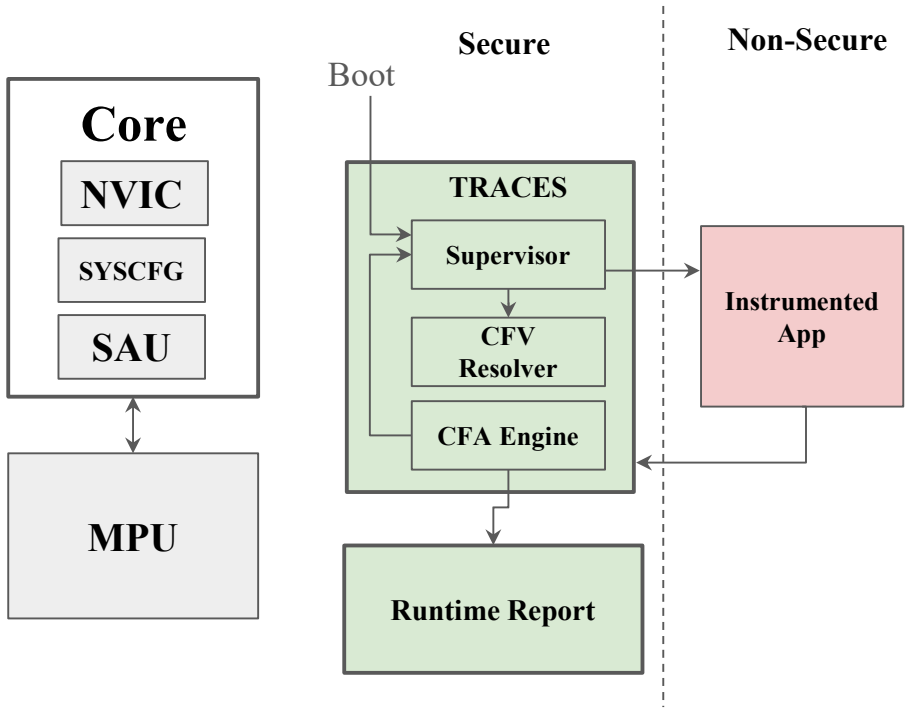


TRACES Workflow: Online phase

During App execution, a new report is generated when

THIS ARE THE THINGS TO ACHIEVE AUDITING – active root of trust

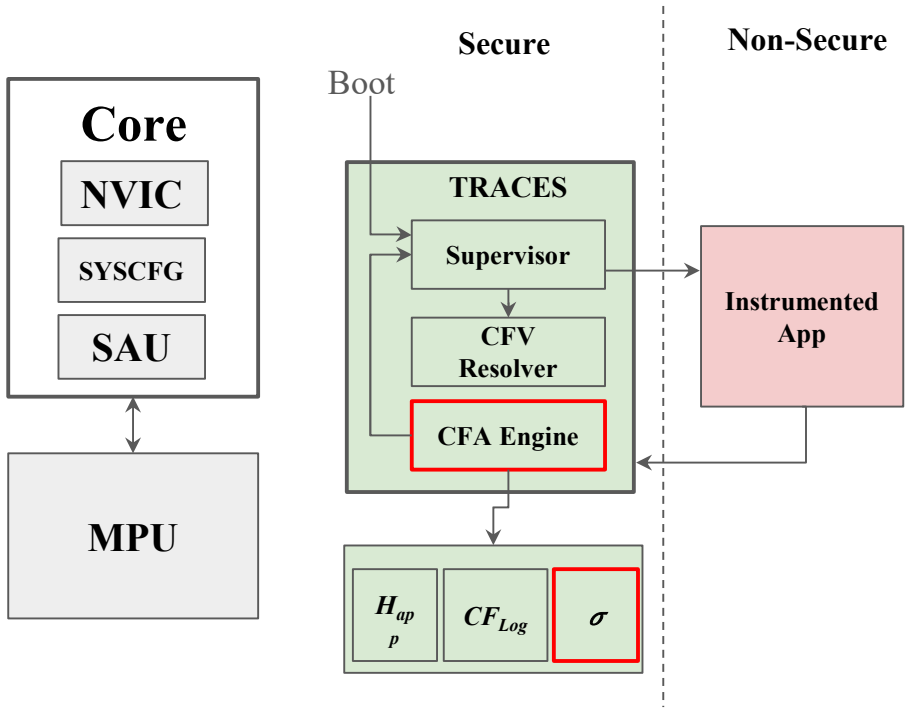
- Timeout has been reached
- CF_{Log} reaches maximum size
- App has concluded



TRACES Workflow: Online phase

CFA Engine generates report by computing a MAC (σ) over:

- CF_{Log}, H_{app}, and challenge

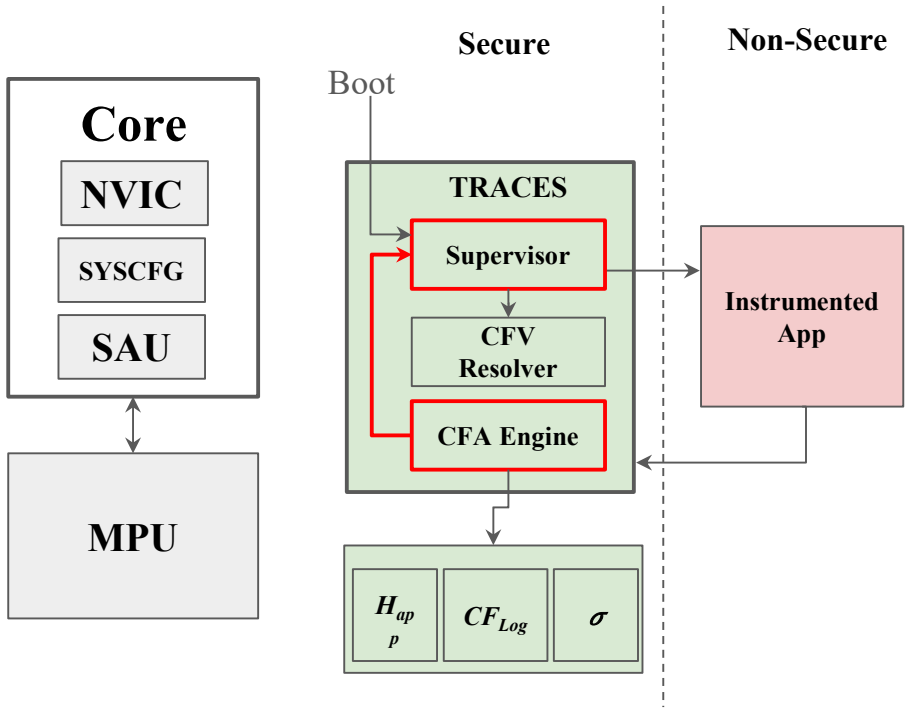


TRACES Workflow: Online phase

CFA Engine generates report by computing a MAC (σ) over:

- CF_{Log}, H_{app}, and challenge

Once σ is generated, CFA Engine invokes Supervisor to transmit report



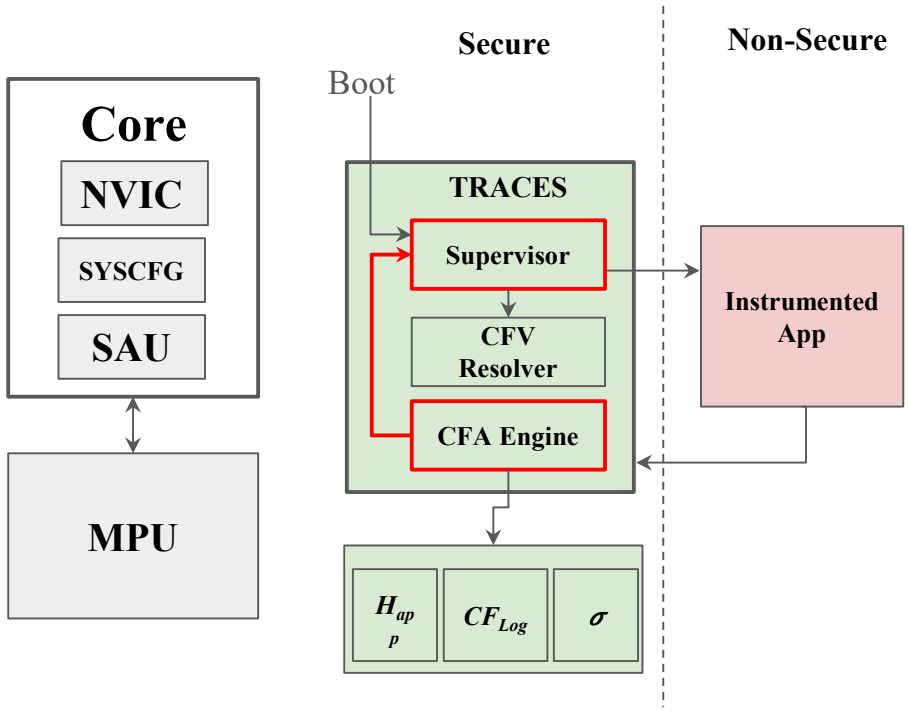
TRACES Workflow: Online phase

CFA Engine generates report by computing a MAC (σ) over:

- CF_{Log}, H_{app}, and challenge

Once σ is generated, CFA Engine invokes Supervisor to transmit report

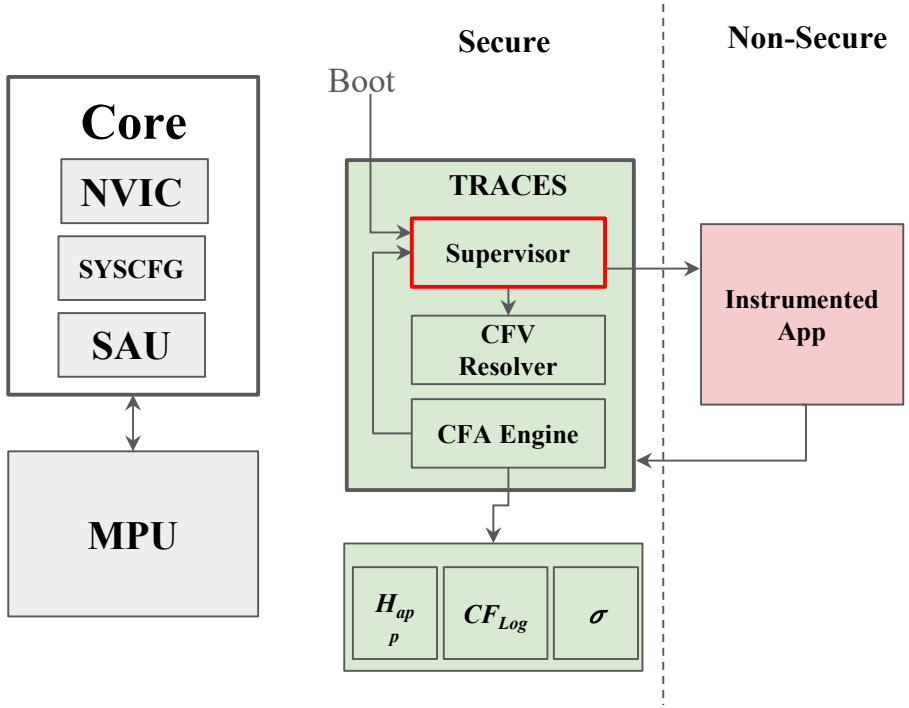
Supervisor waits in an idle state until receiving an authentic message back from Vrf



TRACES Workflow: Online phase

After receiving an authentic message back from Vrf, the Supervisor:

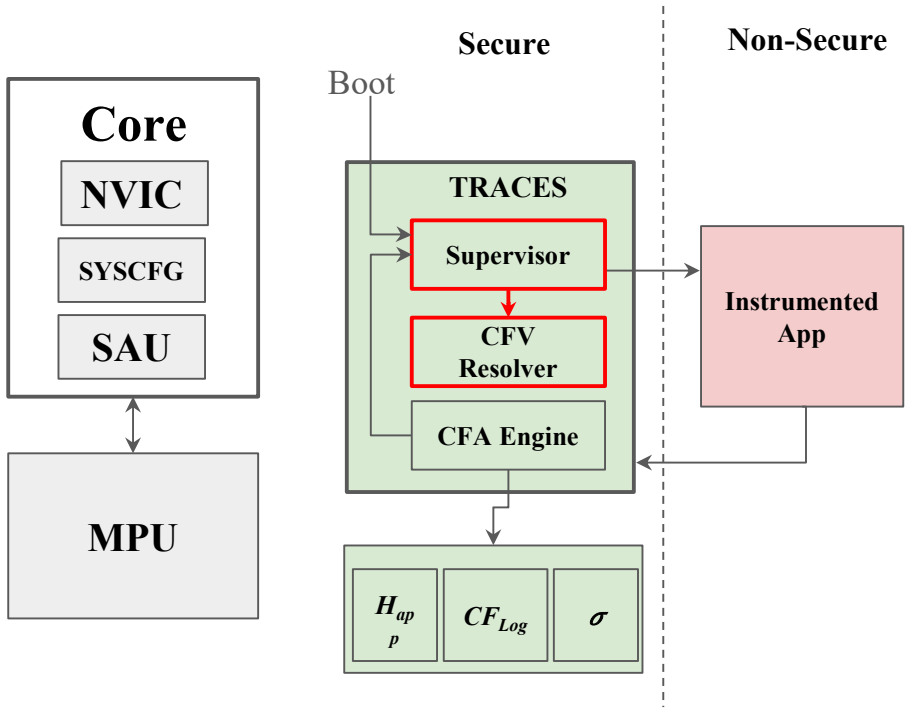
- Invokes CFV Resolver to execute the configured remediation action
- Resumes App
- Ends App and waits for next request



TRACES Workflow: Online phase

If Adversary causes a reboot during the attestation:

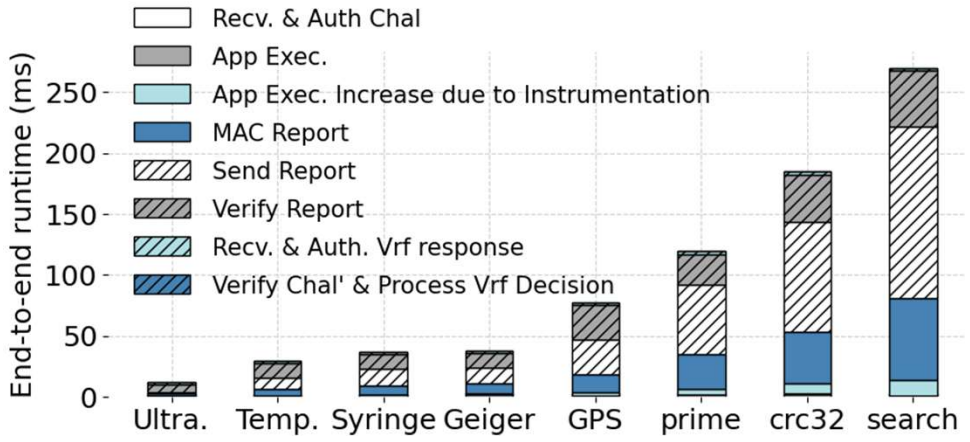
- Supervisor generates a new report And send to verifier before executing anything else



TRACES: End-to-end timing results

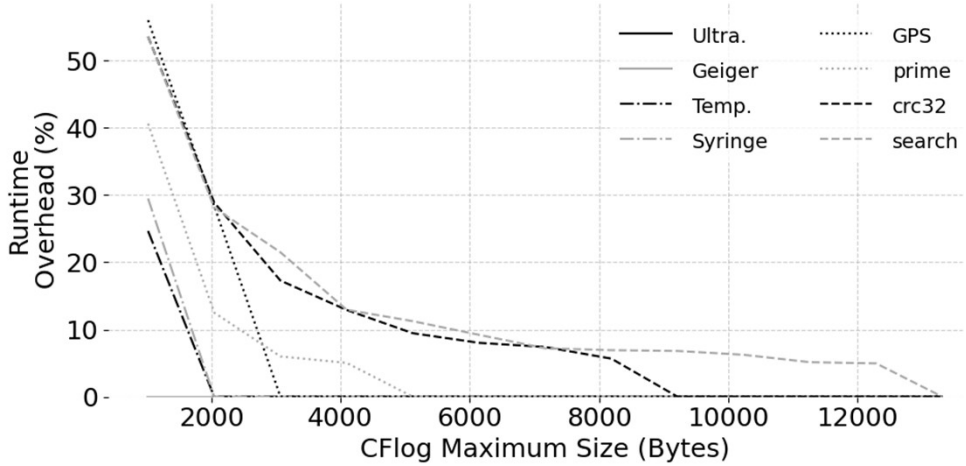
End-to-end runtime of TRACES compared to (best-effort) CFA for one report

- ~2.2 ms additional runtime



TRACES: End-to-end timing results

End-to-end runtime as maximum storage for CF_{Log} decreases



Conclusion

Contribution: TRACES achieves runtime auditing for “off-the-shelf MCUs”

Limitations: Trade-offs in overheads

- Requires App instrumentation
- Small CF_{Log} storage → less memory, but more latency
- Large CF_{Log} storage → more memory, but less latency

Thank you

https://github.com/RIT-CHAOS-SEC/TRACES