Building Intelligent Trustworthy Computing Systems: Challenges and Opportunities
4 November, 2021

How Secure are Trusted Execution Environments? Finding and Exploiting Memory Corruption Errors in Enclave Code

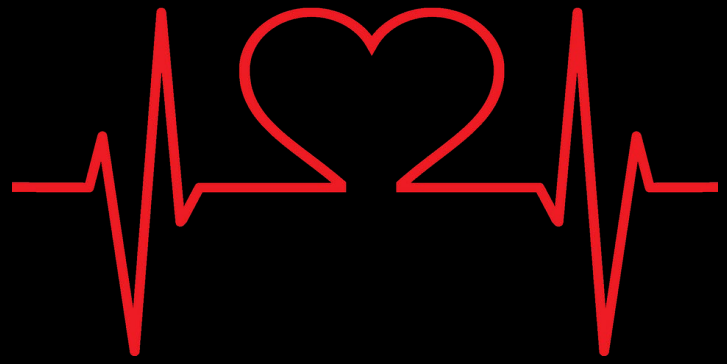
Lucas Davi
Secure Software Systems
University of Duisburg-Essen, Germany

Motivation

- How to reliably protect sensitive data and code from disclosure and modification?

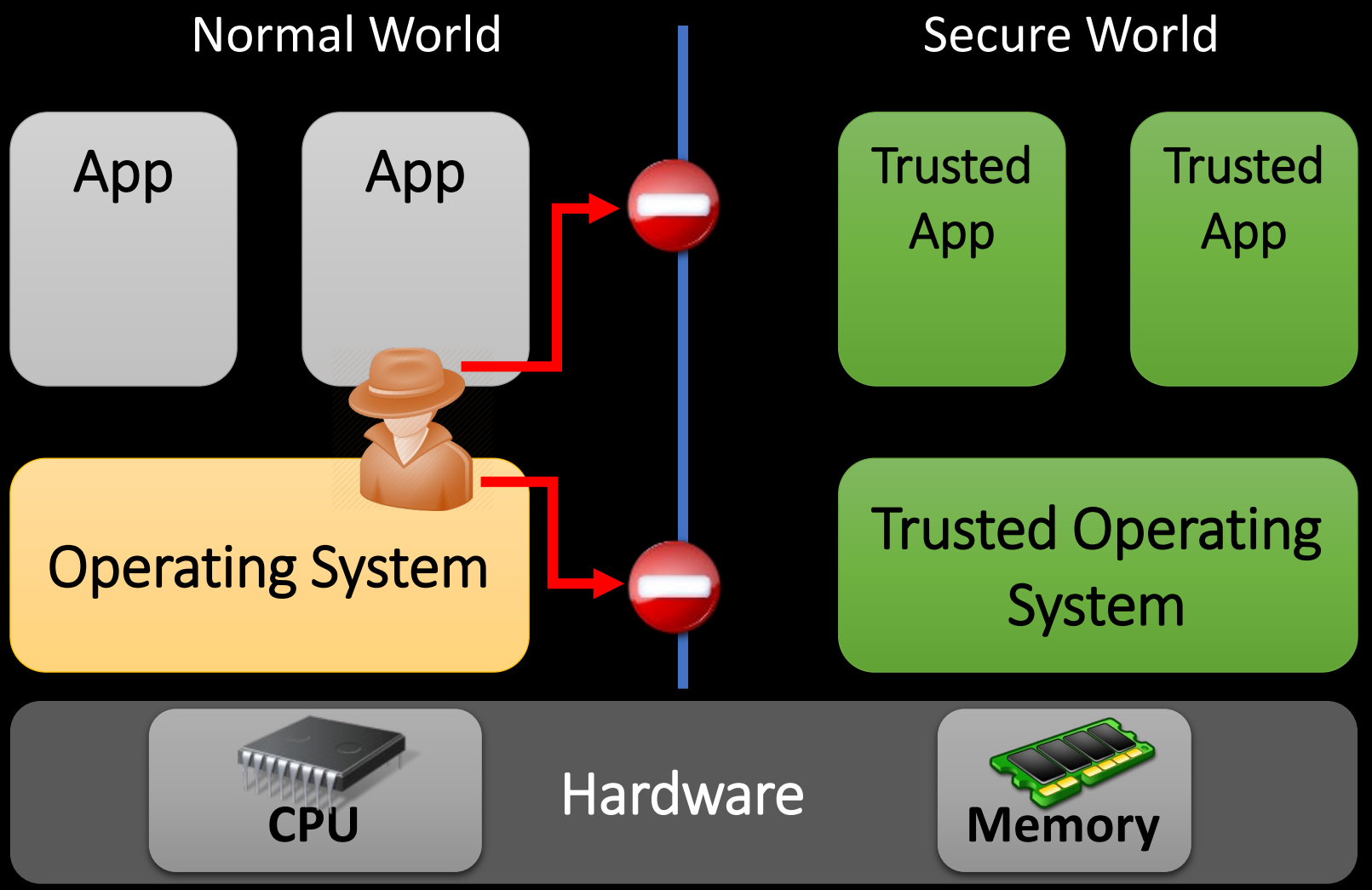
Passwords

Intellectual Property



Medical records

Trusted Execution Environments (TEEs)

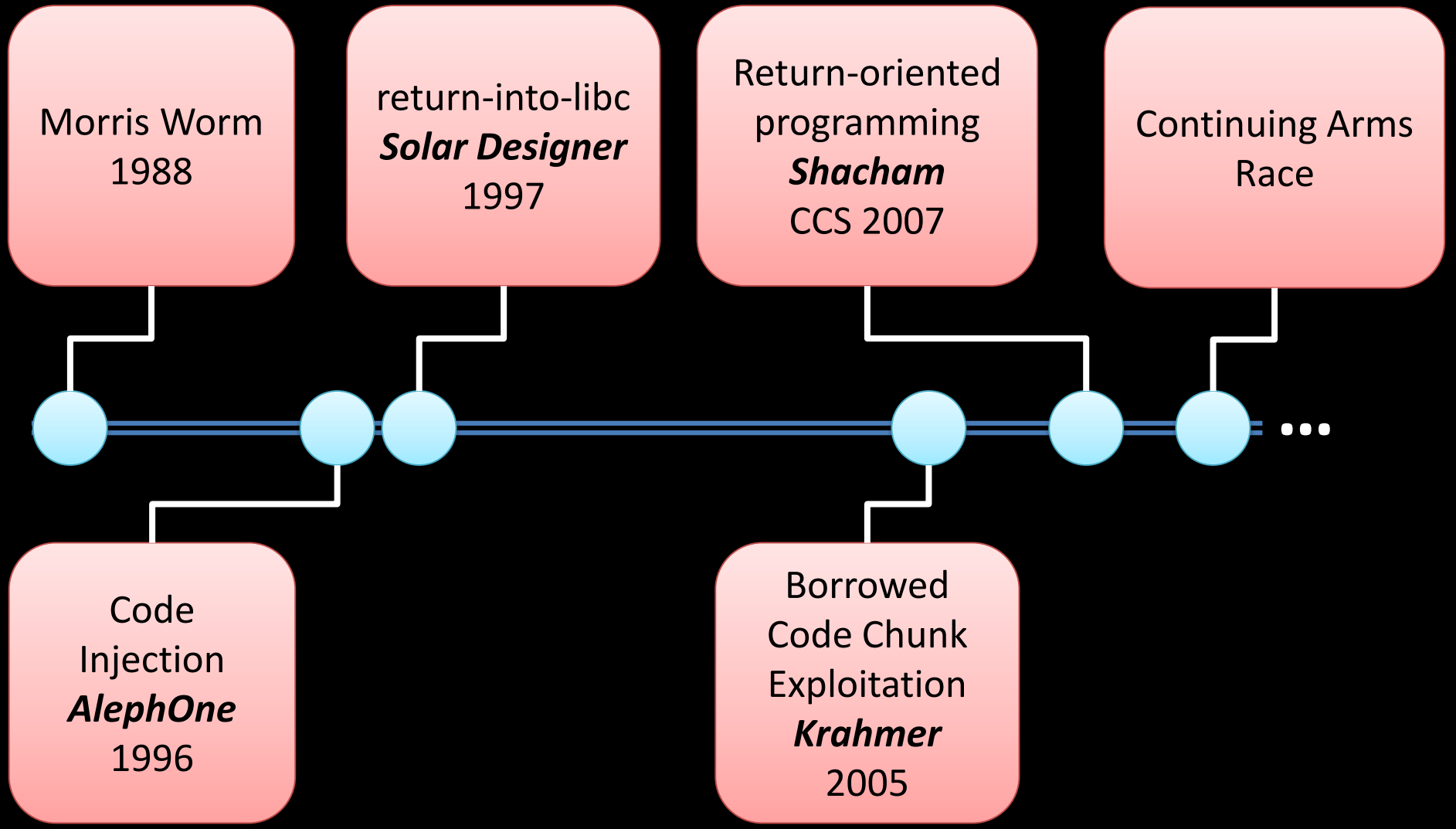


Popular TEE Implementations:

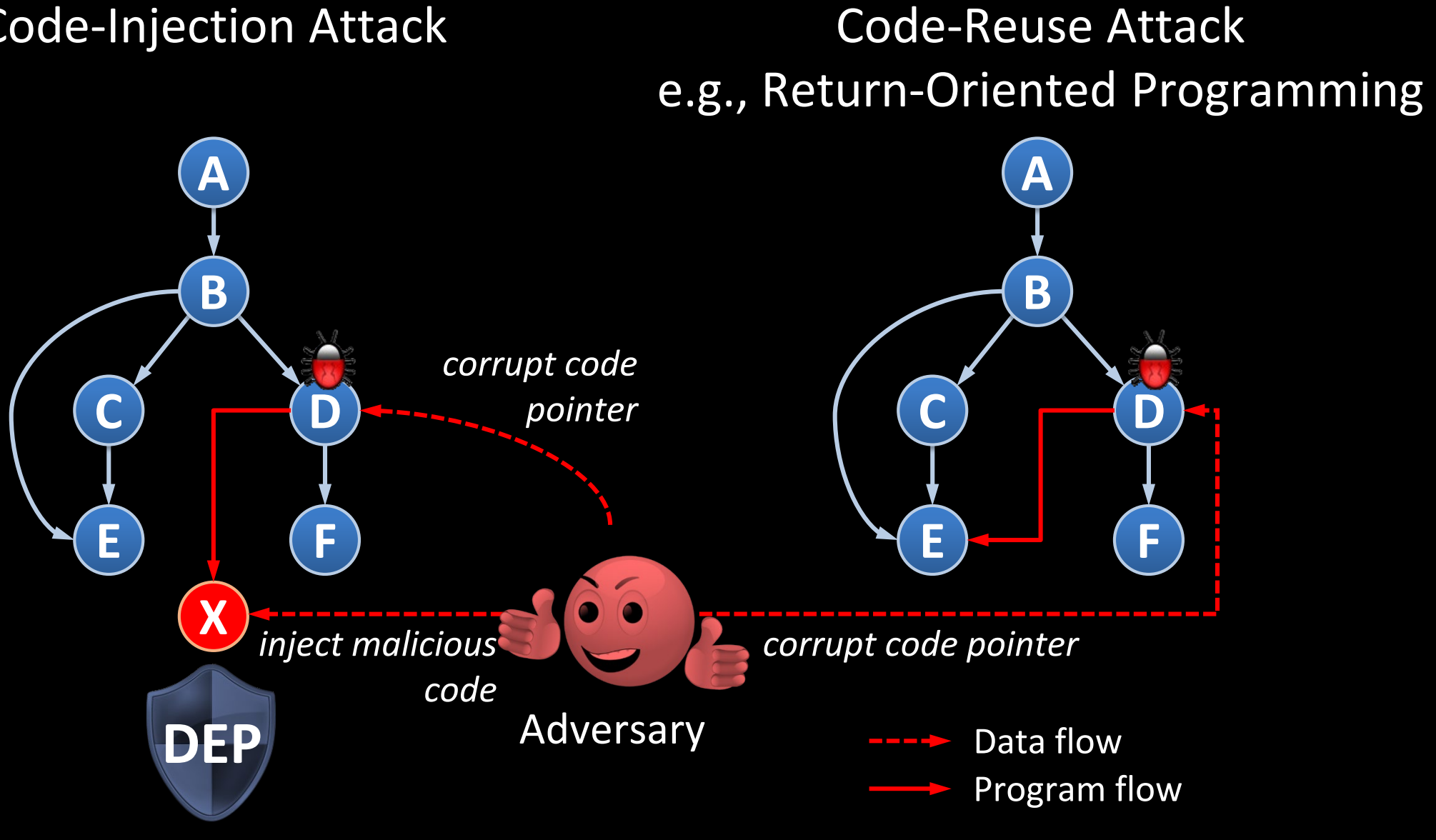
- ARM TrustZone
- Intel Software Guard Extensions (SGX)

Focus of This Talk:
Memory corruption attacks against TEE software

Three Decades of Software Exploits



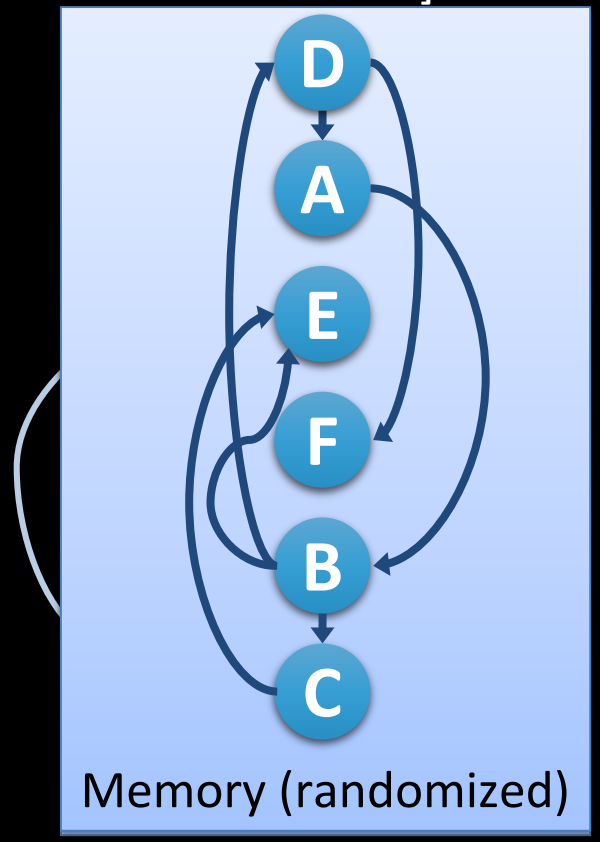
Memory Corruption Attack Classification



Probabilistic vs Enforcement-Based Defense Approach

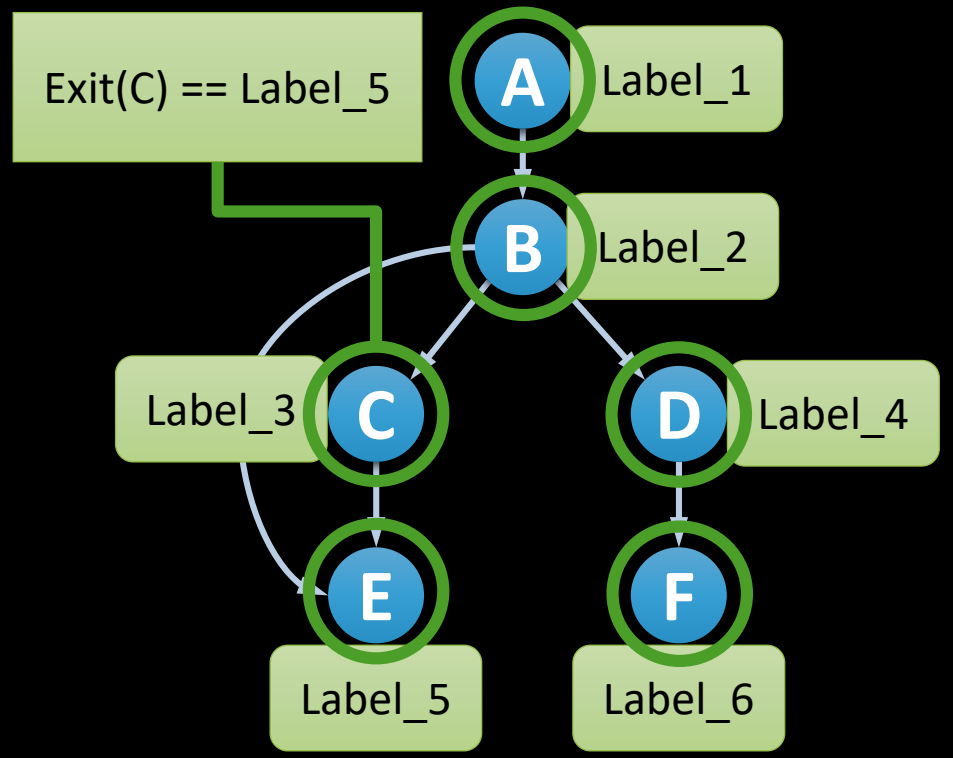
(Fine-grained) Code Randomization

[Cohen 1993 & Larsen et al., SoK IEEE S&P 2014]



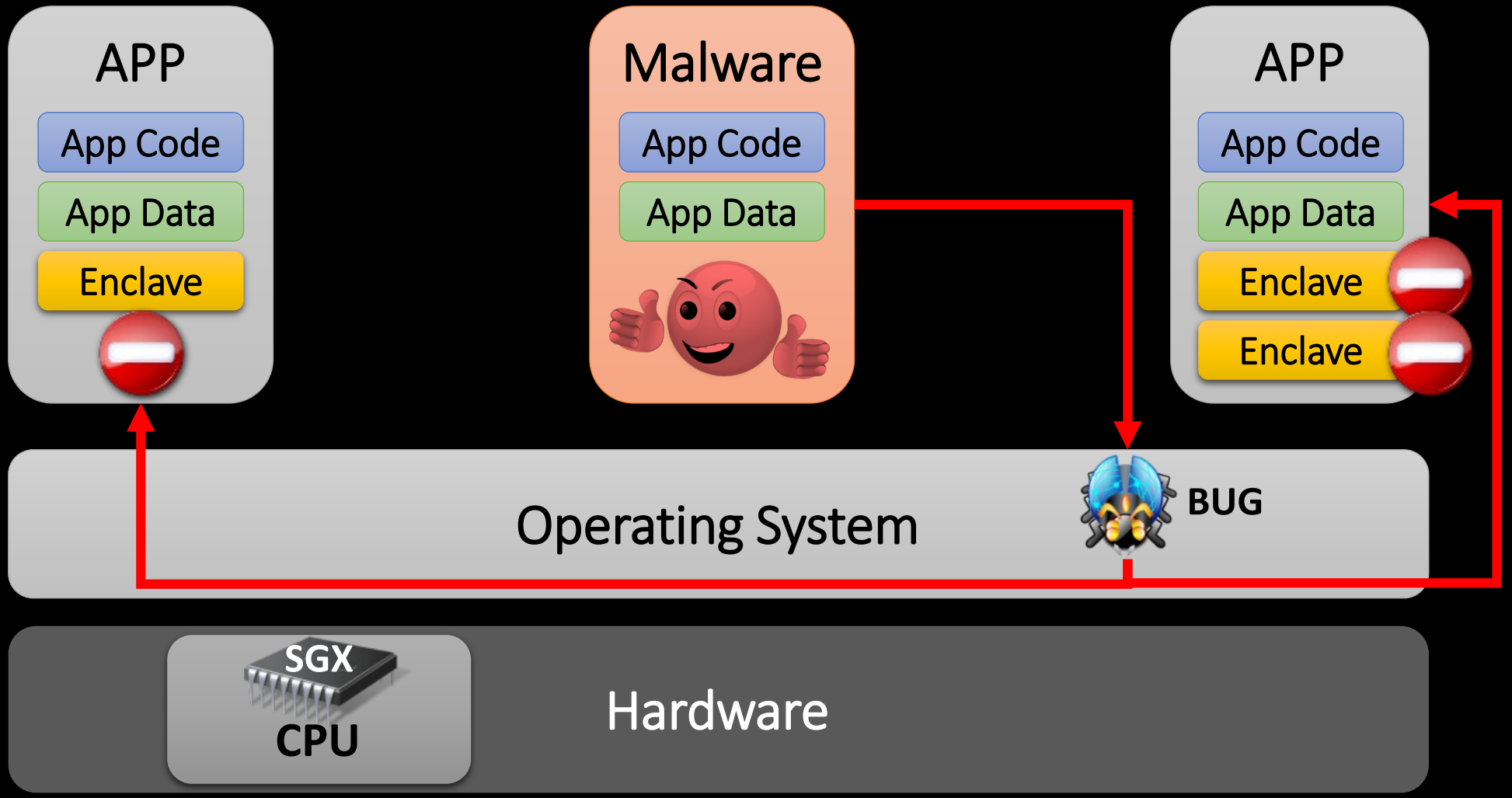
Control-Flow Integrity (CFI)

[Abadi et al., CCS 2005 & TISSEC 2009]

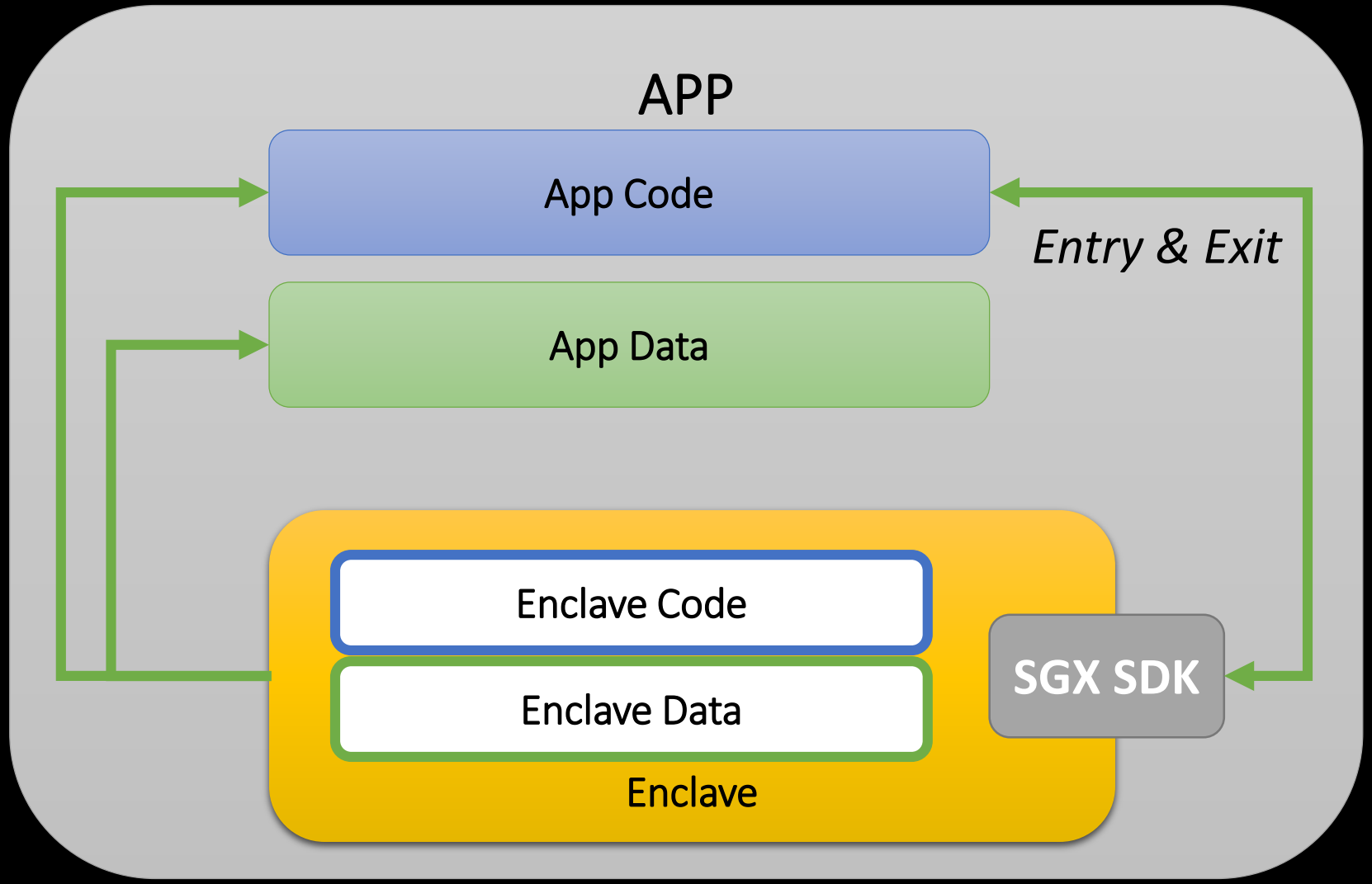


Intel Software Guard Extensions (SGX)
[McKeen et al., Hoekstra et al., Anati et al., HASP’13]

Overview on Intel SGX



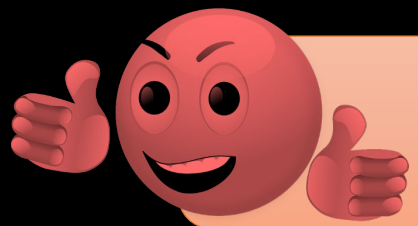
App-Enclave Communication



Entry to Enclave code is only allowed at **pre-defined entry points**

First Run-Time Attacks and Defenses Targeting Intel SGX

Existing Attacks and Defenses



Dark ROP
[USENIX Sec. 2017]

- ROP attack against (unknown) encrypted enclave binaries
- Based on probing attacks
- Requires kernel privileges and ability to repeatedly crash the enclave

SGX-Shield
[NDSS 2017]

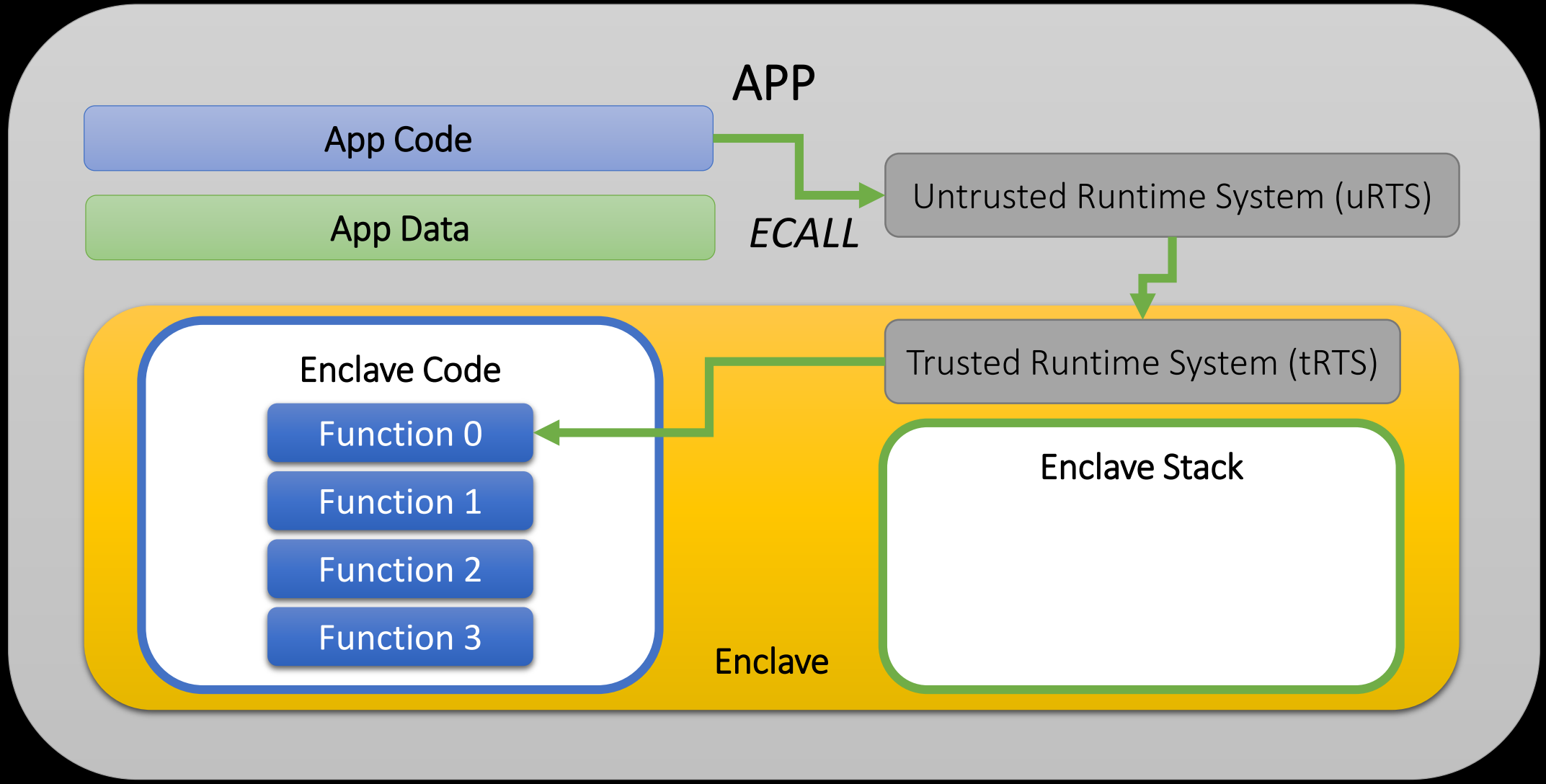
- Enforces fine-grained memory randomization of SGX enclave
- Software-based data execution prevention (DEP)
- Proposes control-flow integrity for return instructions

Can we bypass memory randomization in SGX?

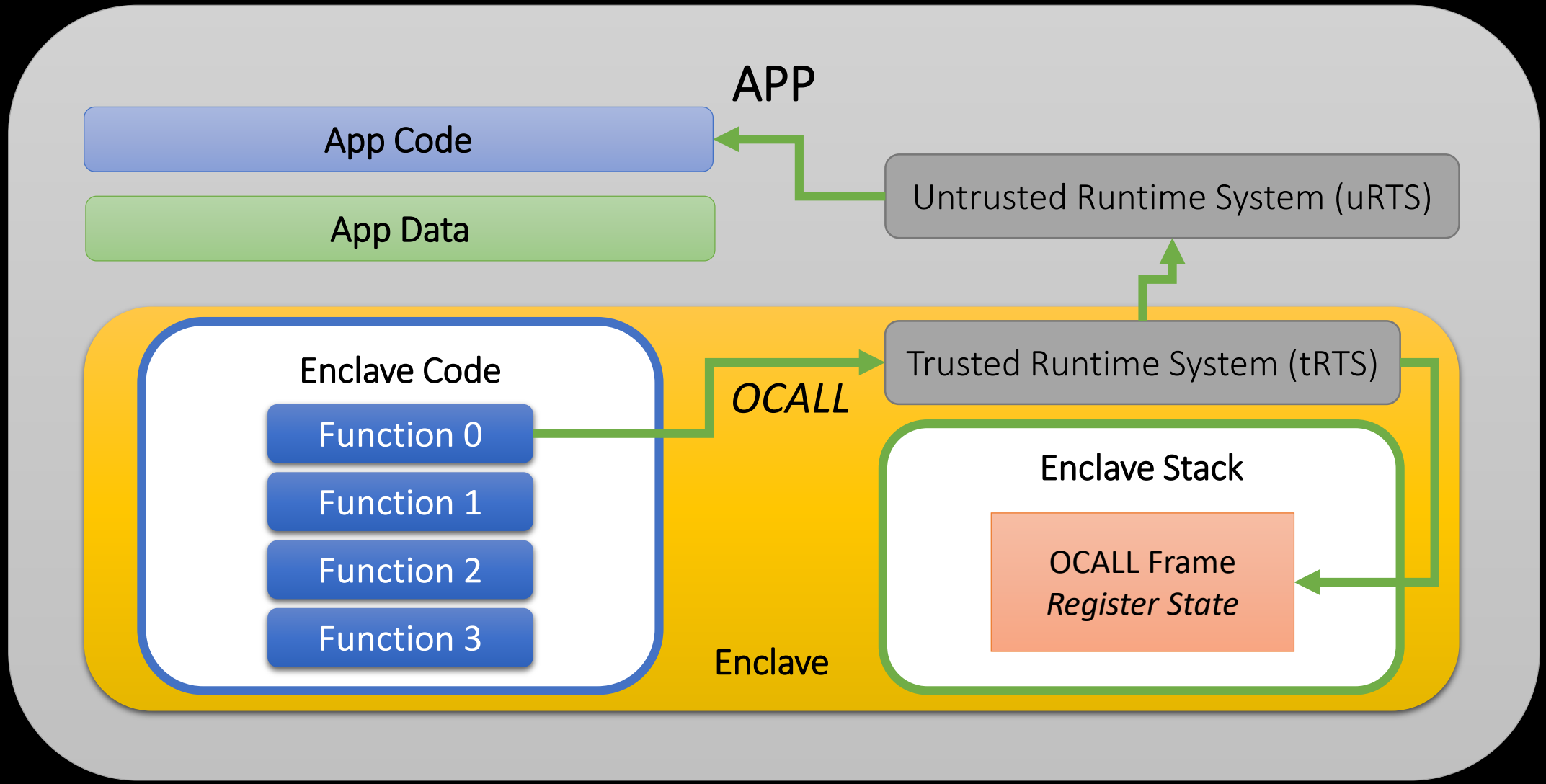
Our main observation is that the Intel SGX SDK includes dangerous return-oriented programming gadgets which are essential for app-enclave communication

[with Biondo et al., USENIX Security 2018]

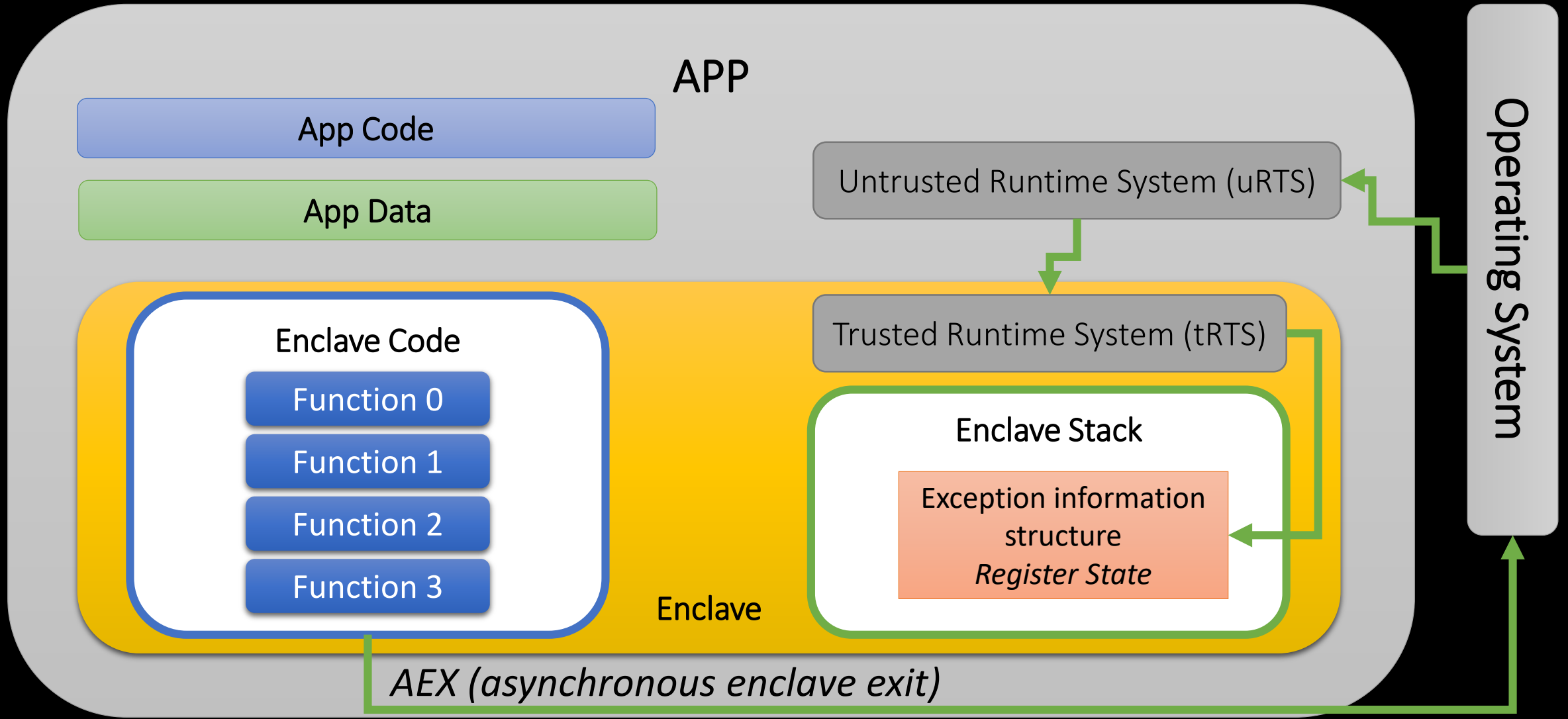
ECALL: Call into an enclave



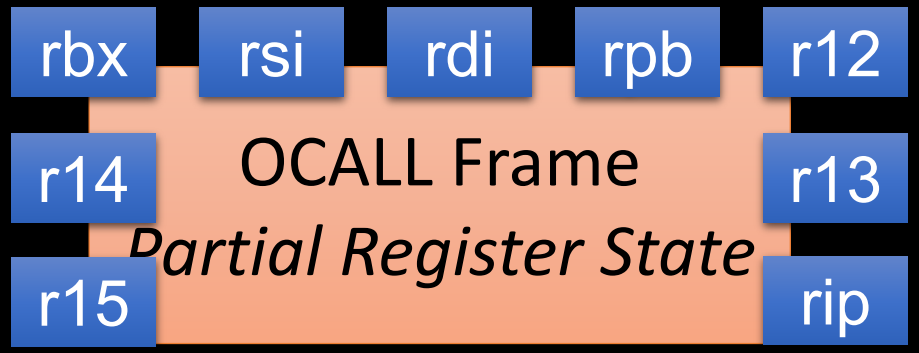
OCALL: Enclave Call to the Host Application



AEX: Asynchronous Enclave Exit (Exception)



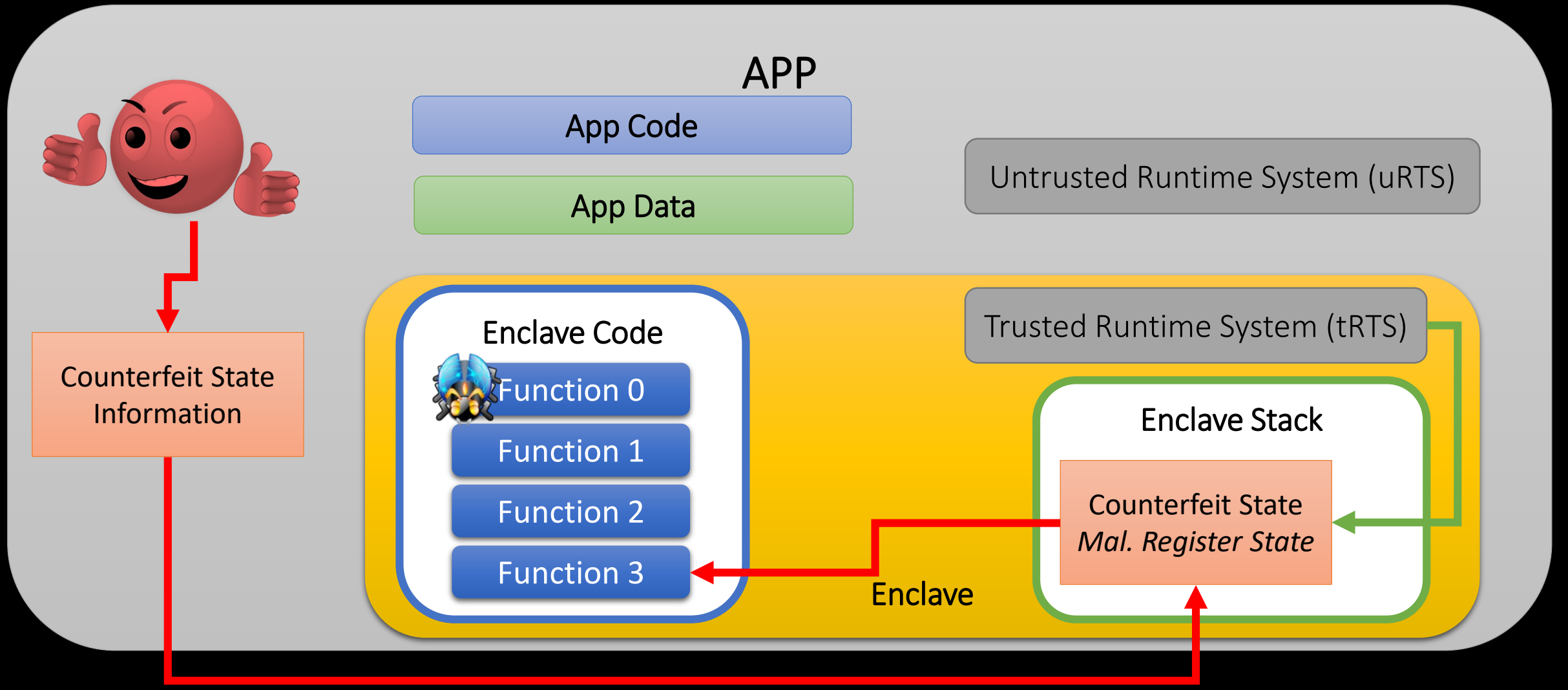
Restoring State is Critical



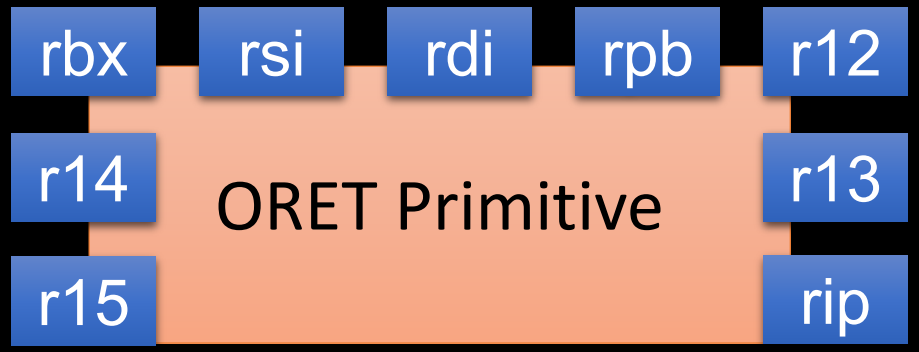
- When OCALL returns, the register state is restored by the tRTS function *asm_oret()*
- After handling the exception, the register state is restored by the tRTS function *continue_execution()*

If an attacker manages to inject a **fake exception structure** or **fake ocall frame**, the attacker controls the subsequent state

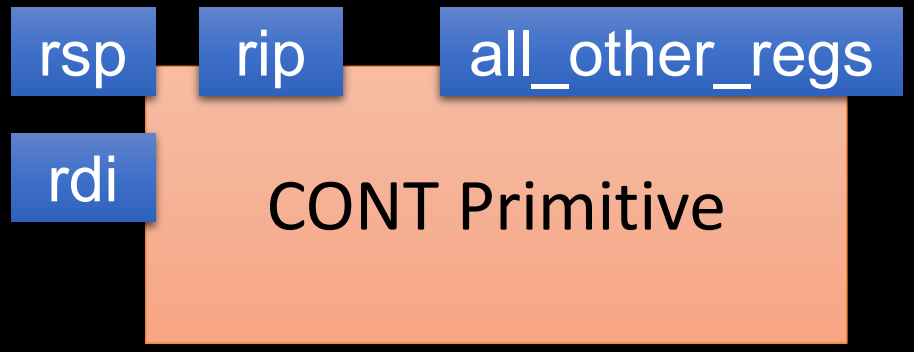
Basic Attack Idea



Two Attack Primitives

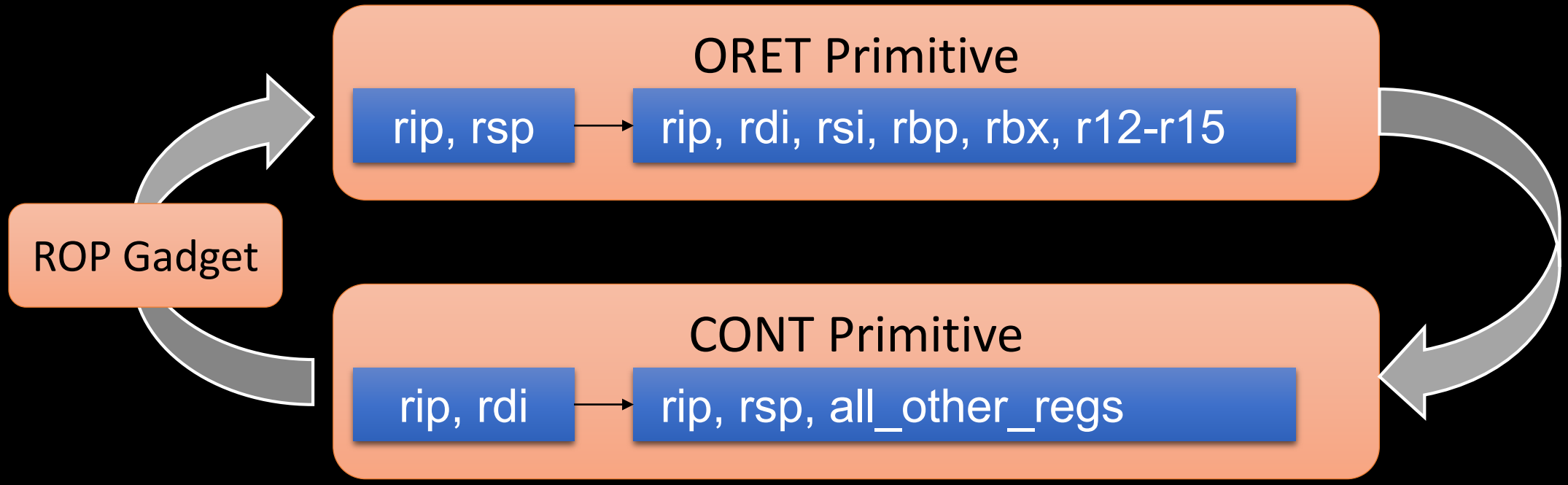


- Prerequisite: stack control

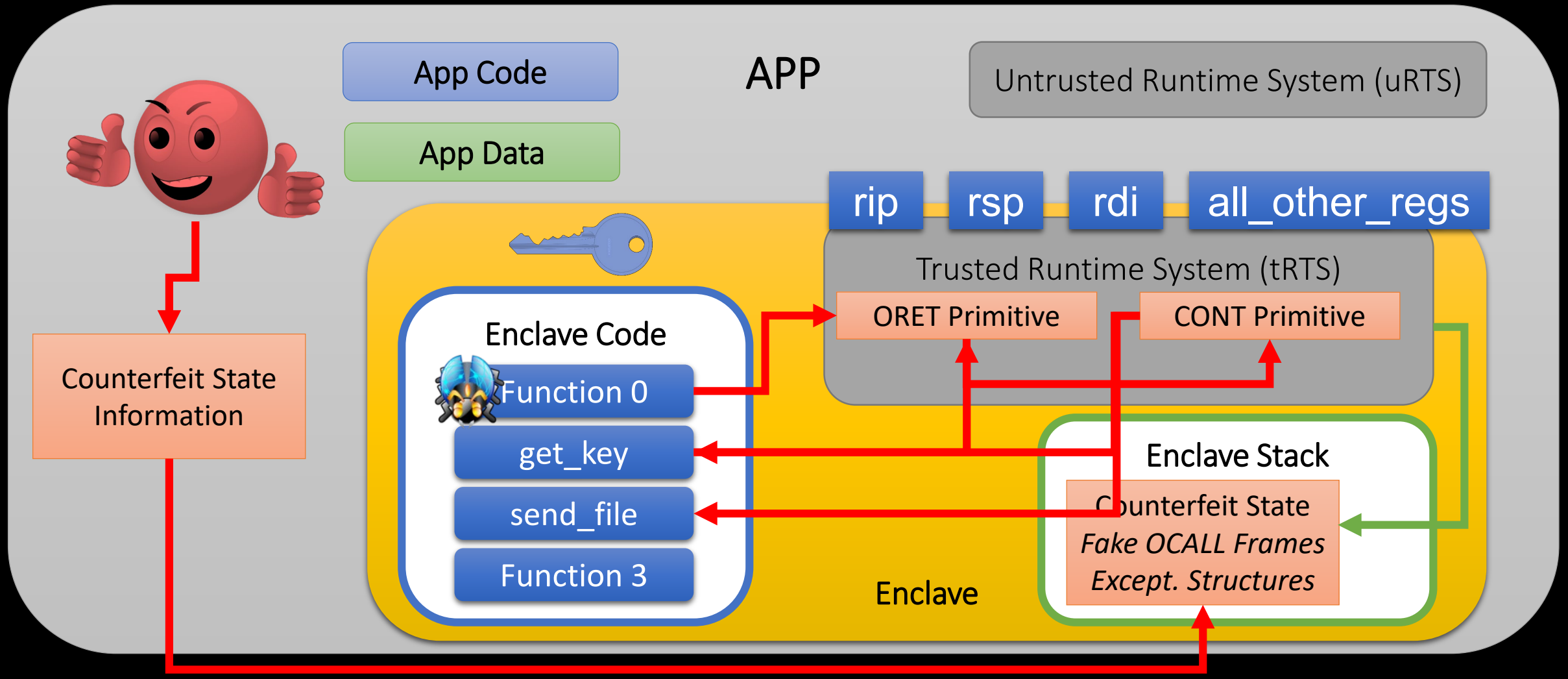


- Prerequisites: function pointer overwrite and control of rdi register

Chaining the Two Primitives

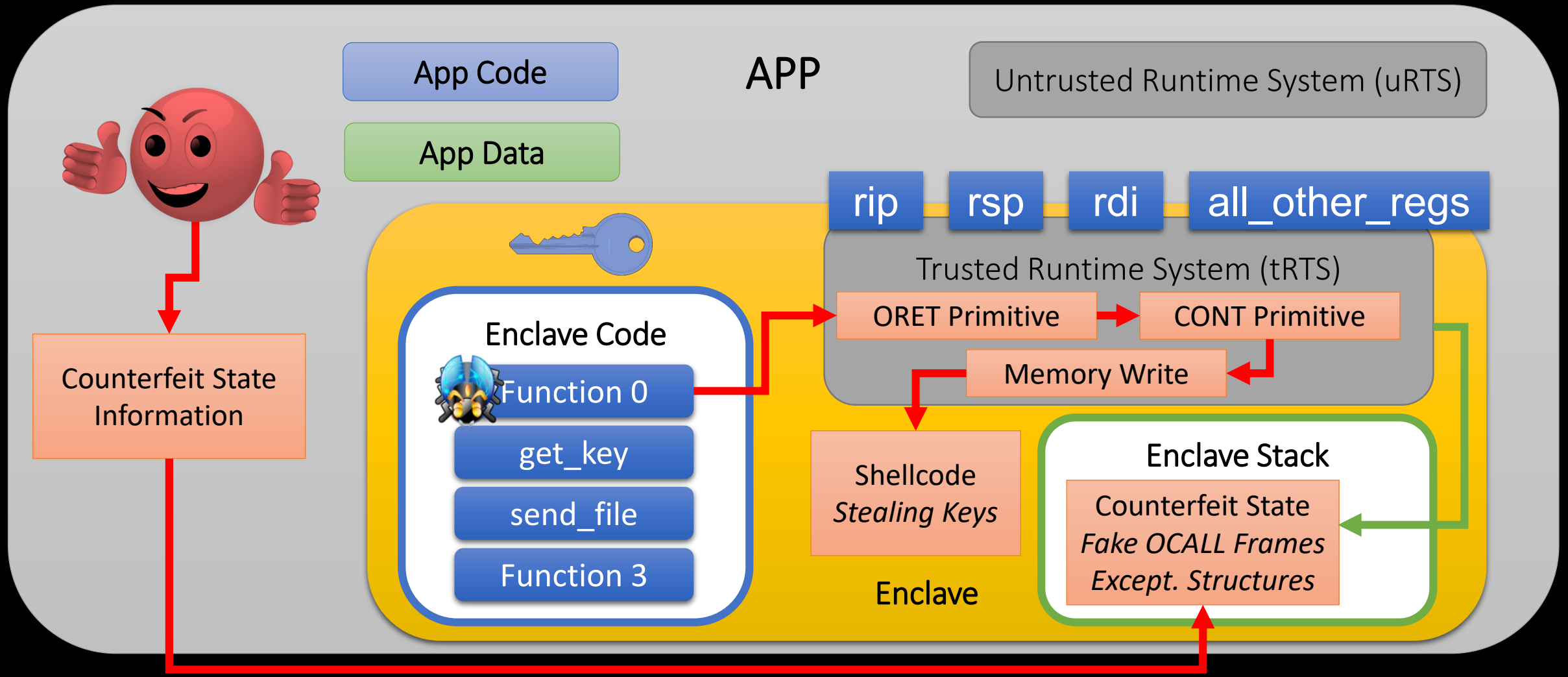


Attack Workflow for Stealing SGX-Protected Keys



However, this attack doesn‘t work if SGX-Shield randomizes the SGX address space

Revisited Attack to Bypass SGX-Shield



Possible Defenses

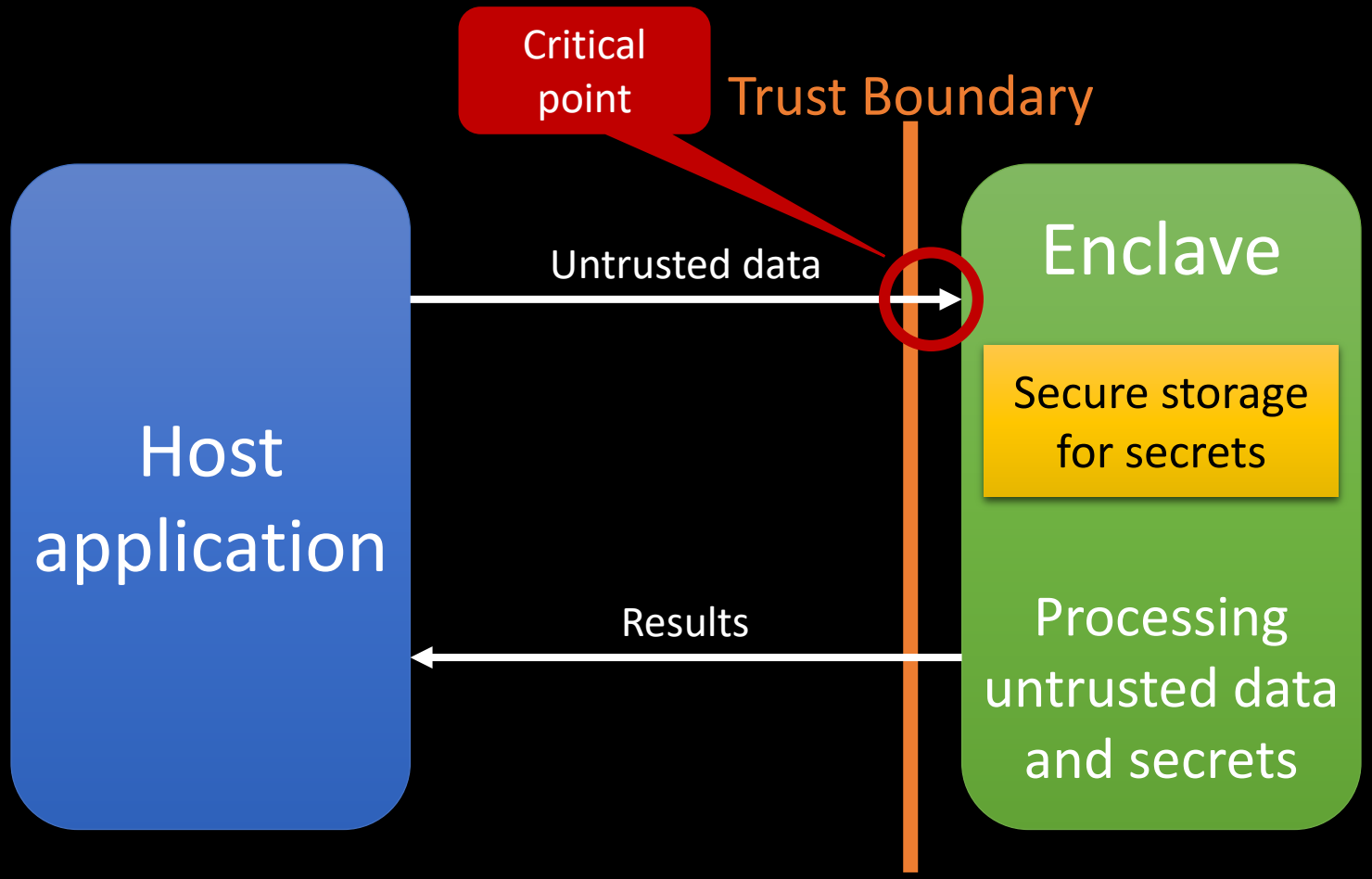
- Removing SDK from enclave memory?
  - Not feasible as OCALL, ECALL, AEX require the tRTS
- Randomizing SDK code?
  - Challenging, the tRTS is accessed through fixed entry points
- Discovering vulnerabilities beforehand?
  - Last part of this talk

There are several open questions:

1. How likely are memory corruption vulnerabilities in SGX enclaves?

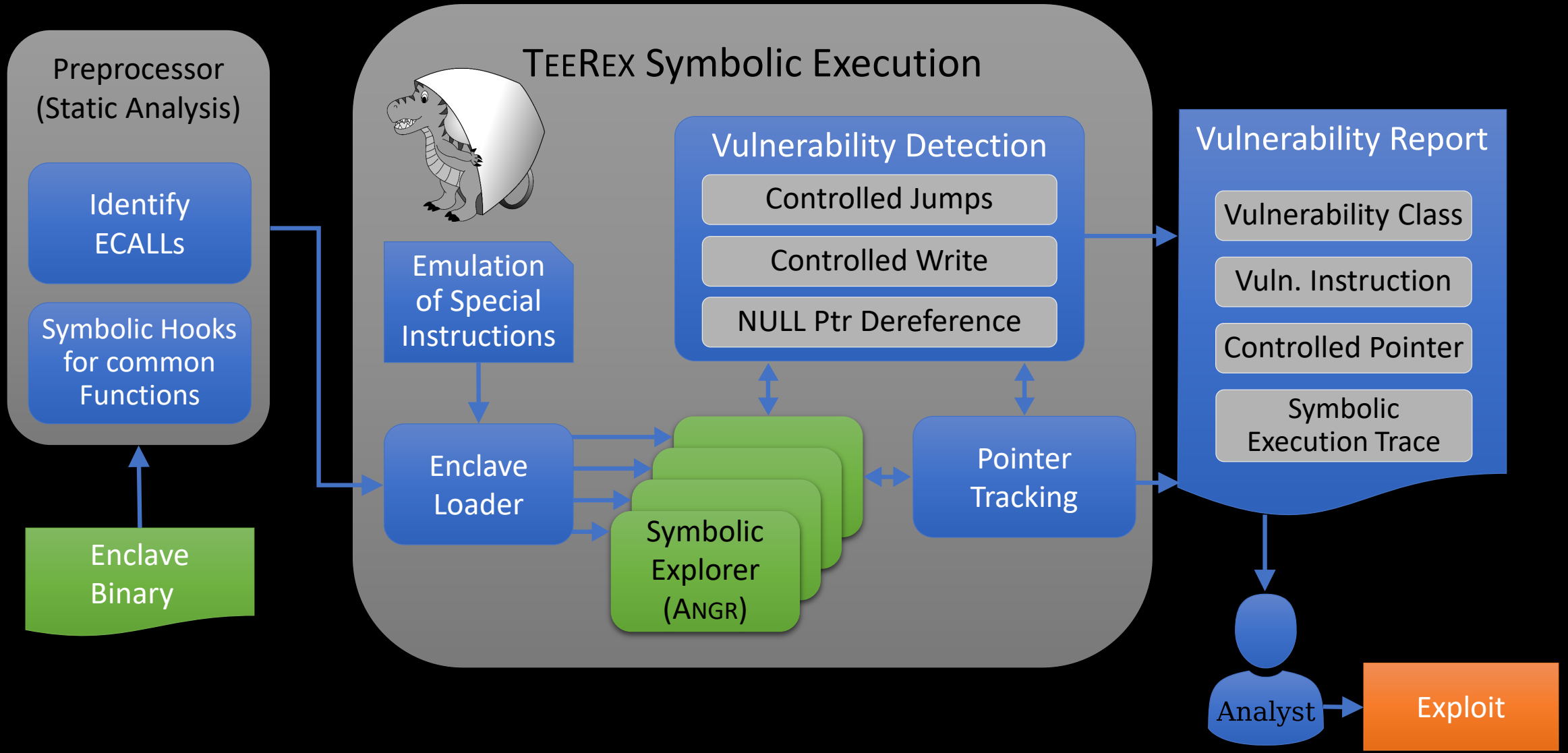
2. Can we develop an automated analysis system that discovers memory corruption vulnerabilities?

System Model of SGX



TeeRex Architecture

[with Cloosters et al., USENIX Sec. 2020]



Exploits in Public Enclaves found with TeeRex

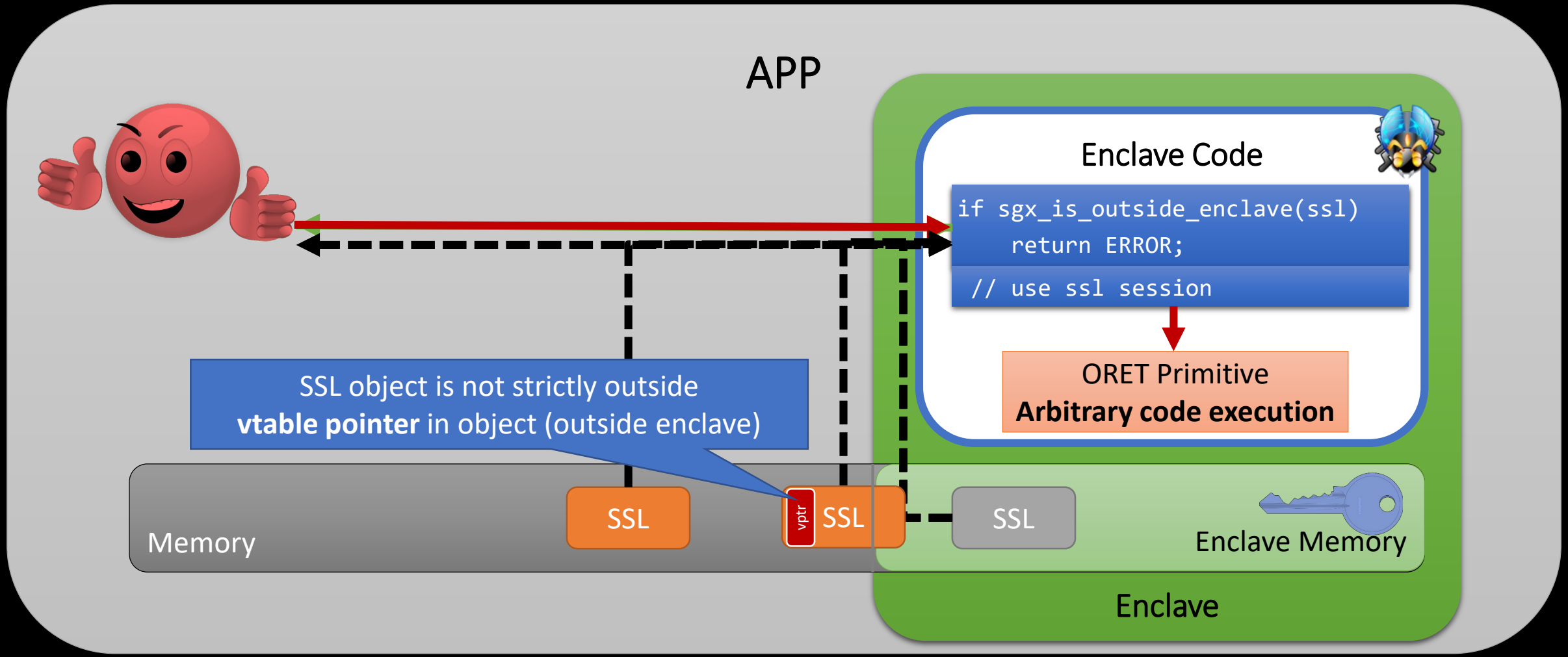
	Project	Exploit	Fixed	Source Code	Target
	**Intel** SGX GMP Example	✓	✓	✓	Linux amd64
	**Baidu** Rust SGX SDK “tlsclient”	✓	✓	✓	Linux amd64
	TaLoS	✓	Not planned	✓	Linux amd64
	**WolfSSL** Example Enclave	✓	✓	✓	Linux amd64
	Synaptics Fingerprint Driver	✓	✓	×	Windows amd64
	Goodix Fingerprint Driver	✓	✓	×	Windows amd64
	SignalApp Contact Discovery	×	-	✓	Linux amd64

CVE-2019-18619

CVE-2020-11667

 Exploit Source Code: https://github.com/uni-due-syssec/teerex-exploits

Baidu/Apache Rust SDK: tlsclient
Pointers to overlapping memory



Discussion

- Symbolic execution vs fuzzing
- Mitigation technologies for TEEs
- What about other TEE architectures? ARM TZ, KeyStone, CURE

Conclusion

Harware-assisted application security is vital to implement trustworthy systems and enhanced security services

However, we need to make sure that an attacker cannot exploit bugs inside the TEE

Q&A