27th USENIX Security Symposium, Baltimore, MD, USA, 15-17 August 2018

The Guard's Dilemma

Efficient Code-Reuse Attacks Against Intel SGX

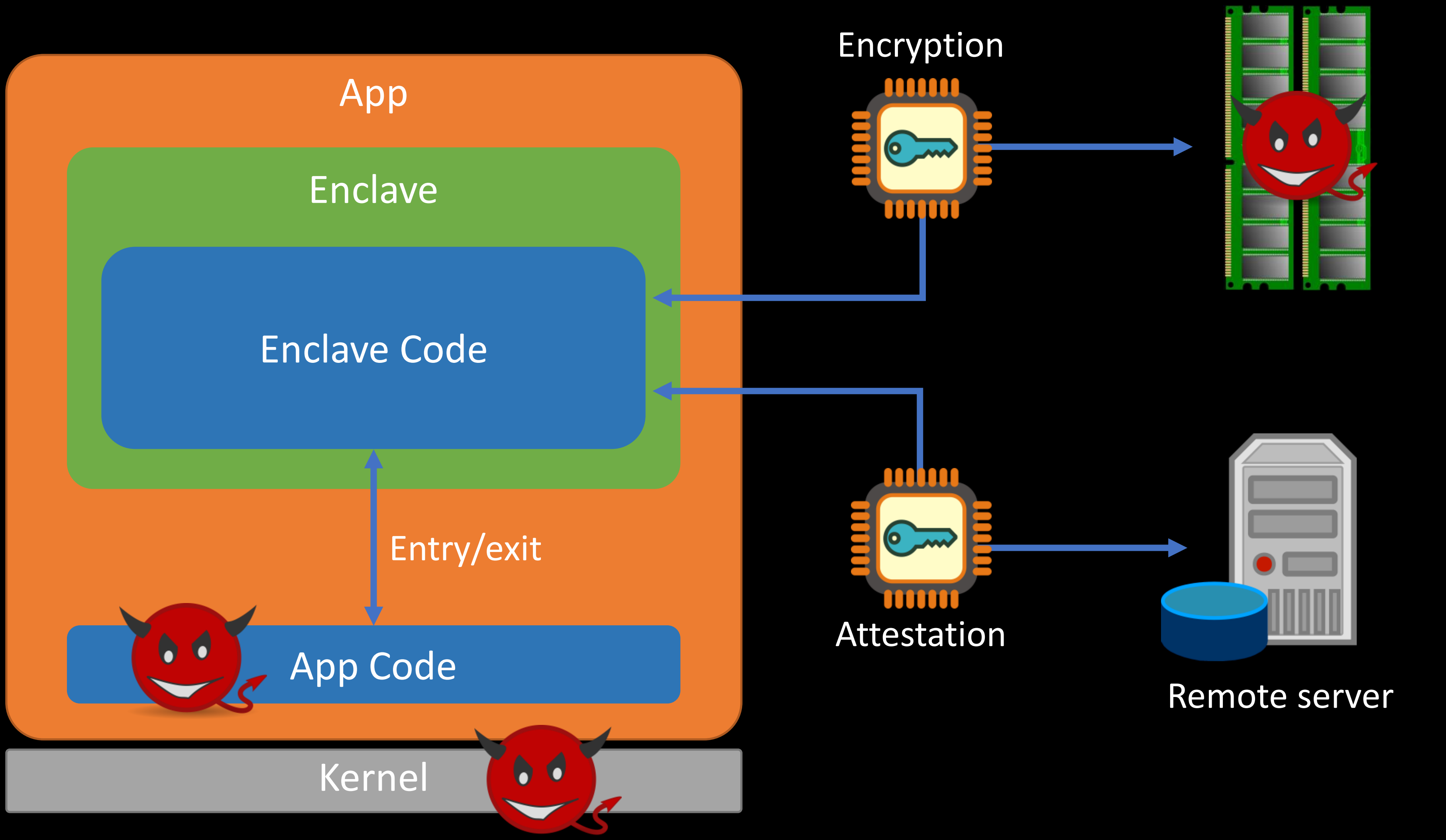
Andrea Biondo[1], Mauro Conti[1], Lucas Davi[2], Tommaso Frassetto[3], Ahmad-Reza Sadeghi[3]

[1] University of Padua, Italy

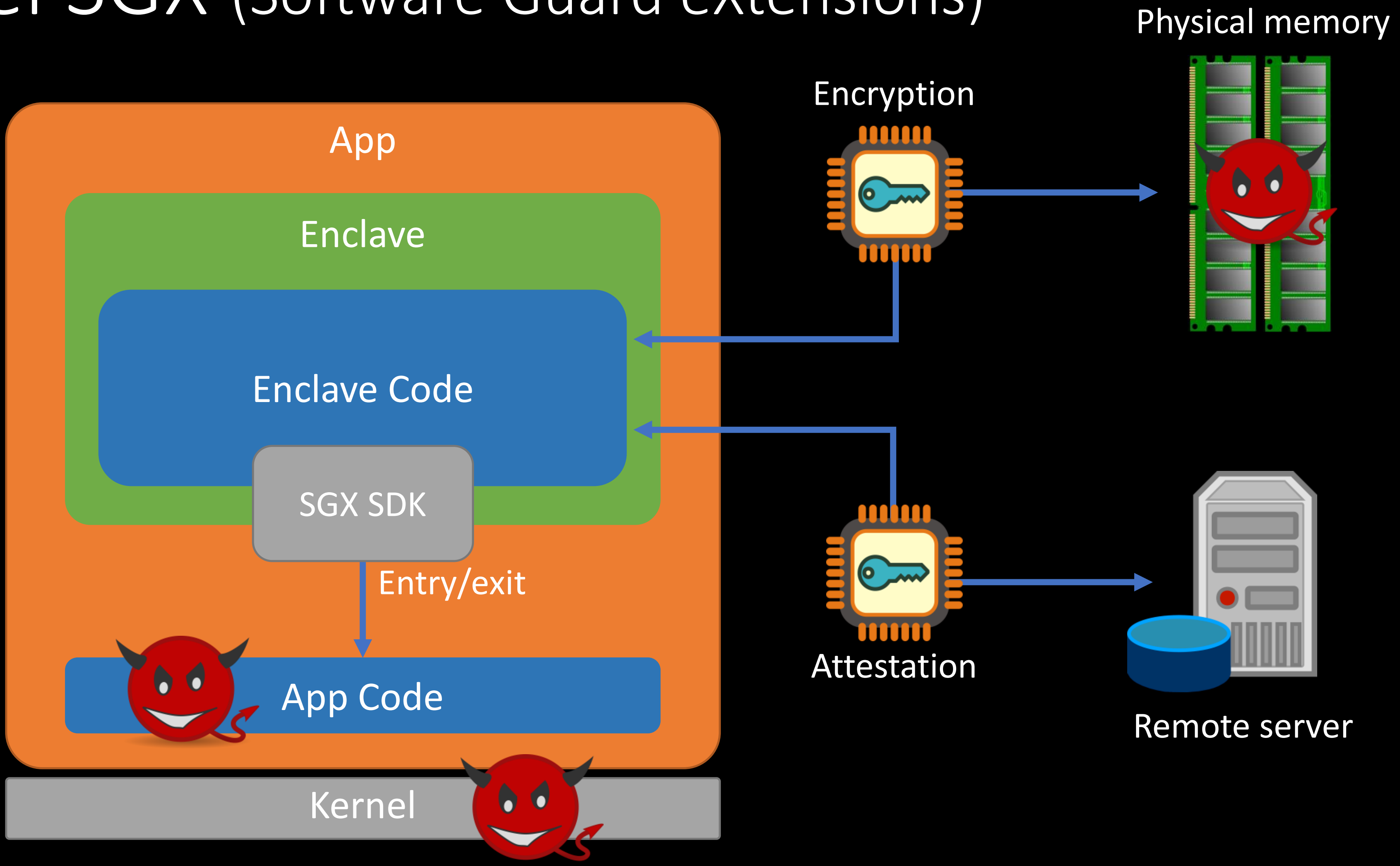
[2] University of Duisburg-Essen, Germany

[3] TU Darmstadt, Germany

Intel SGX (Software Guard eXtensions)



Intel SGX (Software Guard eXtensions)



SGX provides strong isolation.

(that's what it says on the box!)

Controlled-Channel Attacks: Deterministic Side Channels for Untrusted Operating Systems

Yuanzhong Xu, *The University of Texas at Austin*, *yxu@cs.utexas.edu*
Weidong Cui, *Microsoft Research*, *wdcui@microsoft.com*
Marcus Peinado, *Microsoft Research*, *marcuspe@microsoft.com*

Software Grand Exposure: SGX Cache Attacks Are Practical

Ferdinand Brasser[1], Urs Müller[2], Alexandra Dmitrienko[2], Kari Kostiainen[2], Srdjan Capkun[2], and Ahmad-Reza Sadeghi[1]

Boffins show Intel's SGX can leak crypto keys

Software Guard Extensions are supposed to hide data. But the 'Prime+Probe attack' fixes that

CacheZoom: How SGX Amplifies The Power of Cache Attacks

Ahmad Moghimi, Gorka Irazoqui, and Thomas Eisenbarth

Inferring Fine-grained Control Flow Inside SGX Enclaves with Branch Shadowing

Sangho Lee, Ming-Wei Shih, Prasun Gera, Taesoo Kim, and Hyesoon Kim, *Georgia Institute of Technology;* Marcus Peinado, *Microsoft Research*

Telling Your Secrets Without Page Faults: Stealthy Page Table-Based Attacks on Enclaved Execution

Jo Van Bulck, *imec-DistriNet, KU Leuven;* Nico Weichbrodt and Rüdiger Kapitza, *IBR DS, TU Braunschweig;* Frank Piessens and Raoul Strackx, *imec-DistriNet, KU Leuven*

Three more data-leaking security holes found in Intel chips as designers swap security for speed

Apps, kernels, virtual machines, SGX, SMM at risk from attack

SgxPectre Attacks: Stealing Intel Secrets from SGX Enclaves via Speculative Execution

Guoxing Chen, Sanchuan Chen, Yuan Xiao, Yinqian Zhang, Zhiqiang Lin, Ten H. Lai
Department of Computer Science and Engineering

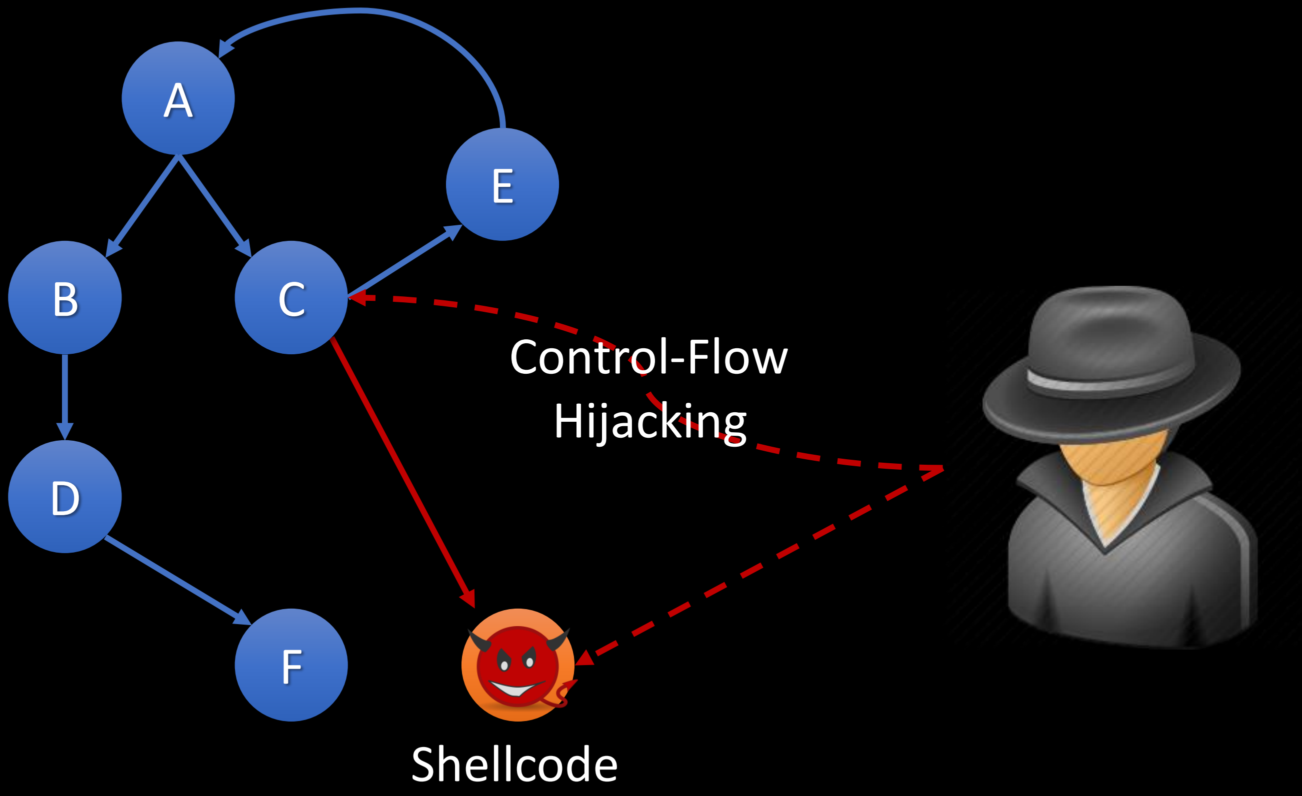
Foreshadow: Extracting the Keys to the Intel SGX Kingdom with Transient Out-of-Order Execution

Jo Van Bulck, *imec-DistriNet, KU Leuven;* Marina Minkin, *Technion;* Ofir Weisse, Daniel Genkin, and Baris Kasikci, *University of Michigan;* Frank Piessens, *imec-DistriNet, KU Leuven;* Mark Silberstein, *Technion;* Thomas F. Wenisch, *University of Michigan;* Yuval Yarom, *University of Adelaide and Data61;* Raoul Strackx, *imec-DistriNet, KU Leuven*

Just like normal programs,
SGX code can have bugs.

Control-Flow Attacks

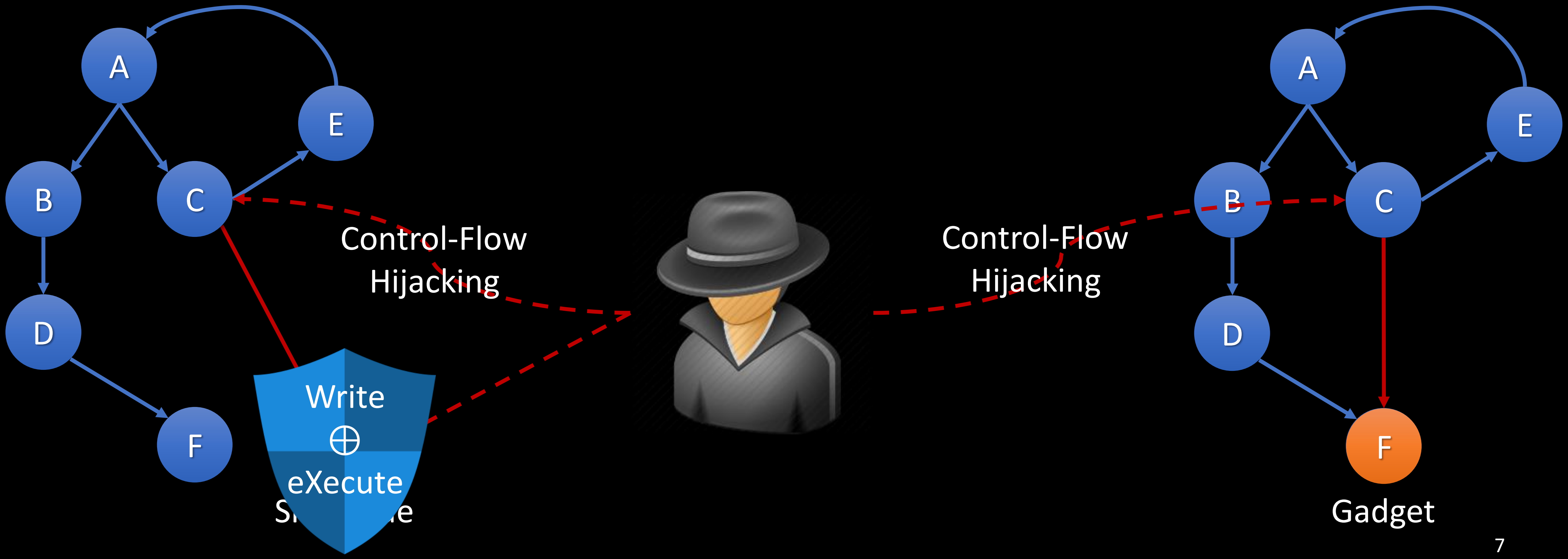
Code Injection



Control-Flow Attacks

Code Injection

Code Reuse
(e.g., Return-Oriented Programming)

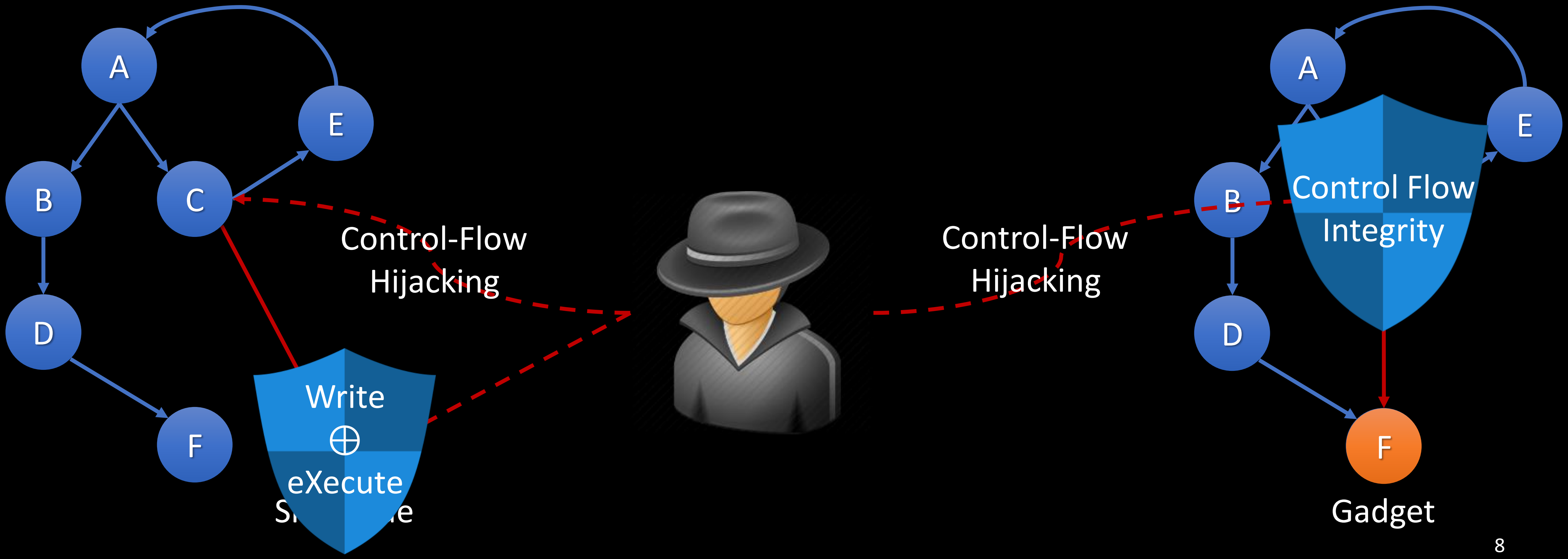


Control-Flow Attacks

Code Injection

Code Reuse

(e.g., Return-Oriented Programming)



Related work

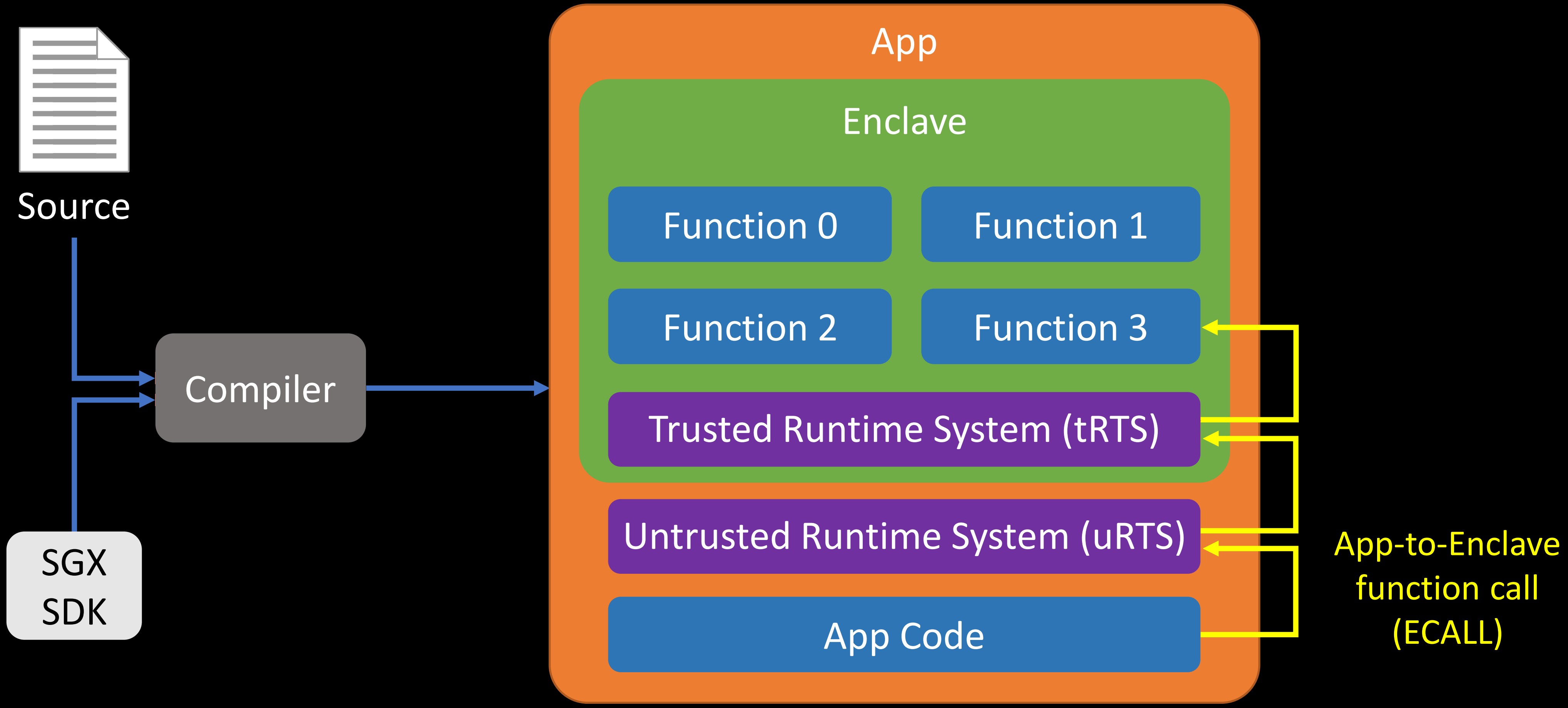
Dark-ROP
[Lee et al., USENIX Security 2017]

- Remote attestation + loader = no access to enclave code
- ROP still feasible by finding gadgets through oracles

SGX-Shield
[Seo et al., NDSS 2017]

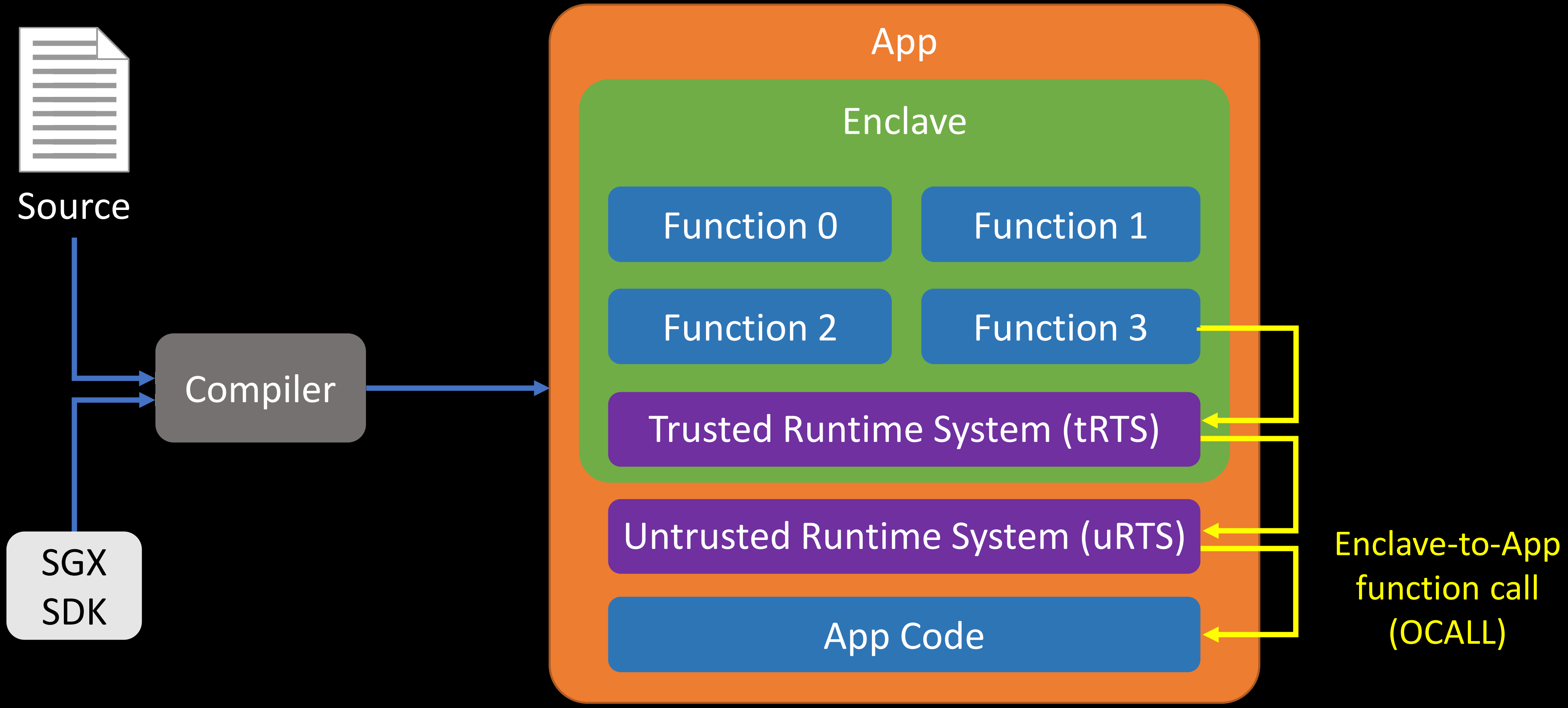
- Fine-grained enclave randomization, W⊕X, Software Fault Isolation, Control Flow Integrity
- State-of-the-art hardening scheme

The SGX SDK

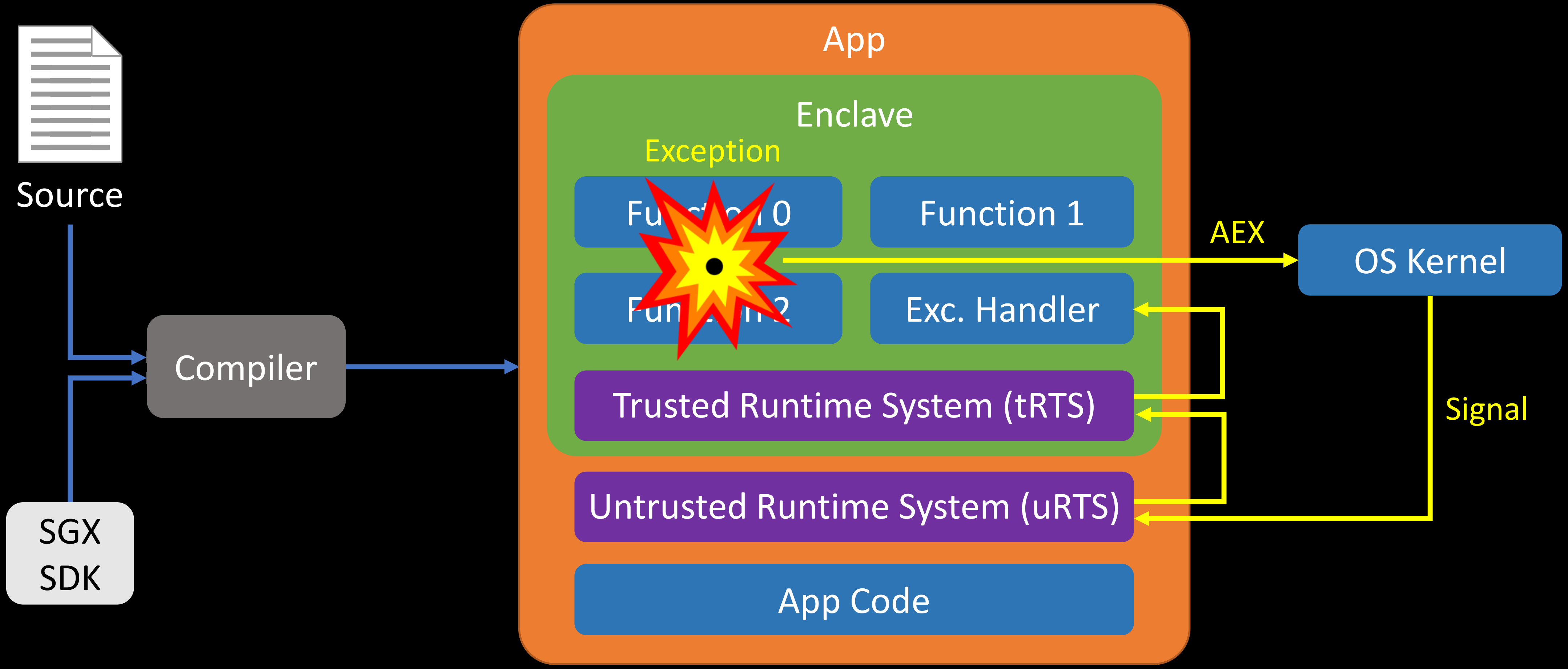


Source
Compiler
SGX
SDK
App
Enclave
Function 0
Function 1
Function 2
Function 3
Trusted Runtime System (tRTS)
Untrusted Runtime System (uRTS)
App Code
App-to-Enclave
function call
(ECALL)

The SGX SDK


Source
Compiler
SGX SDK
App
Enclave
Function 0
Function 1
Function 2
Function 3
Trusted Runtime System (tRTS)
Untrusted Runtime System (uRTS)
App Code
Enclave-to-App function call (OCALL)

The SGX SDK


Source
Compiler
SGX SDK
App
Enclave
Exception
Function 0
Function 1
Function 2
Exc. Handler
Trusted Runtime System (tRTS)
Untrusted Runtime System (uRTS)
App Code
AEX
OS Kernel
Signal

The Guard’s Dilemma

Novel SGX code-reuse attack

Dispatches ROP gadgets

Uses only existing tRTS functionality

Why?

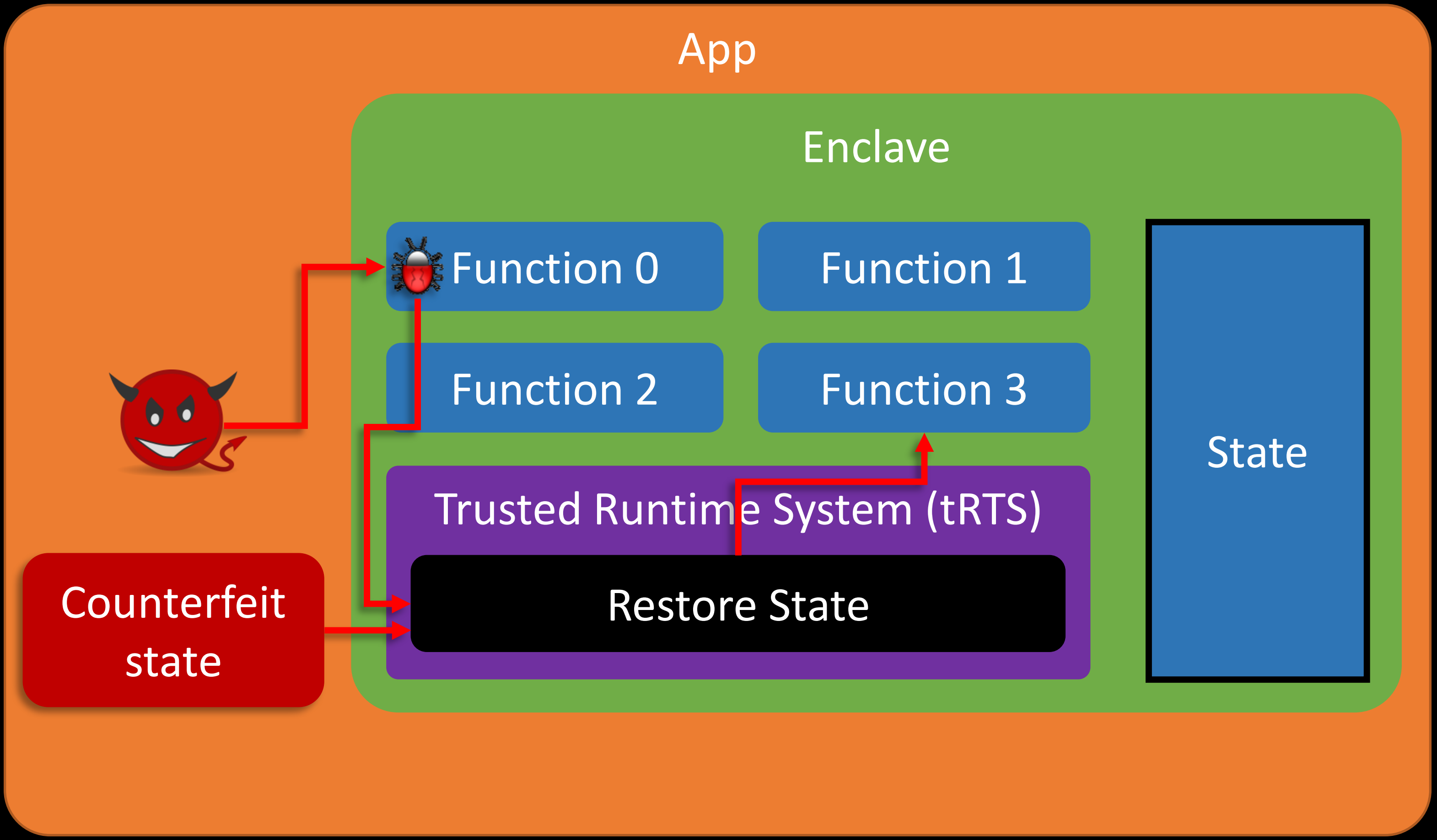
Motivation

Widespread SDK usage

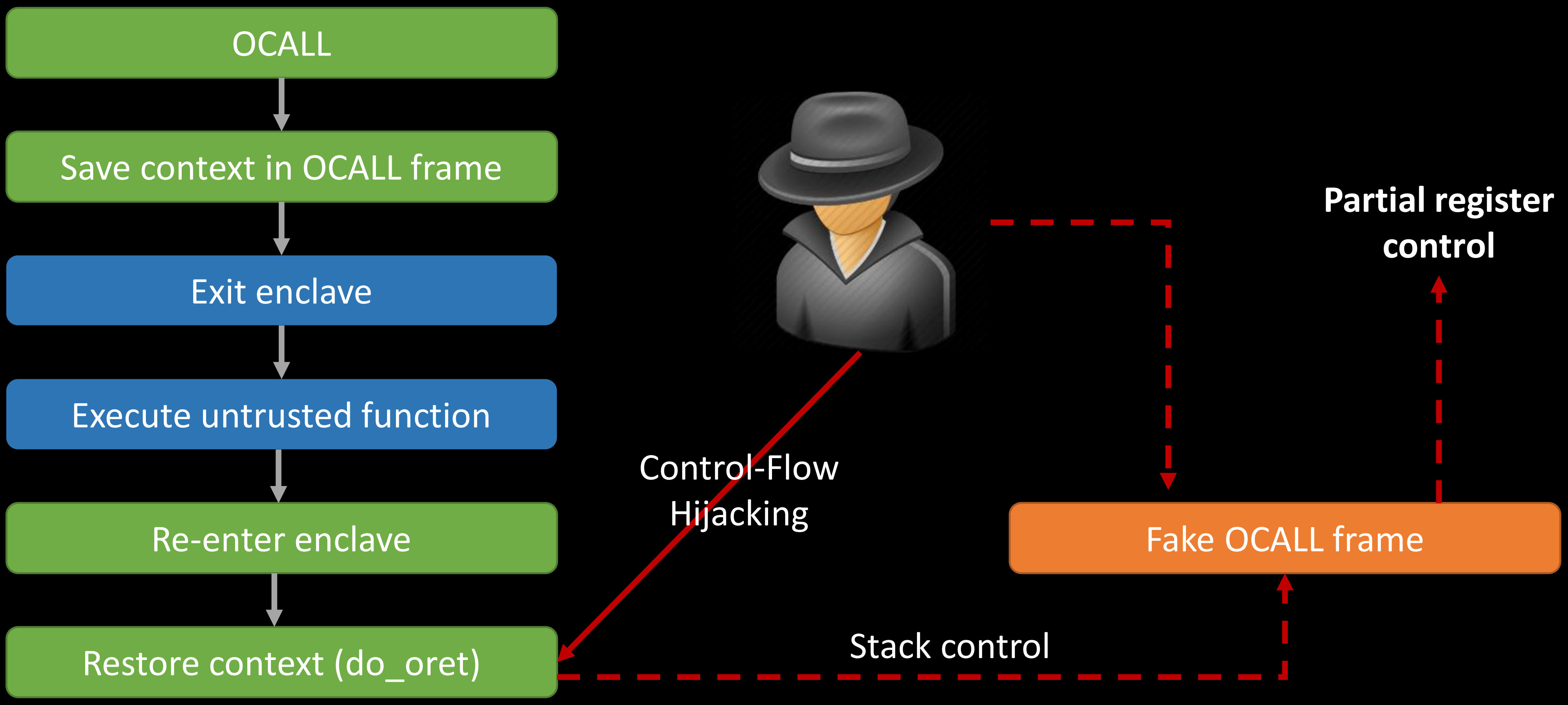
Easier exploitation

Existing hardening does not cover tRTS

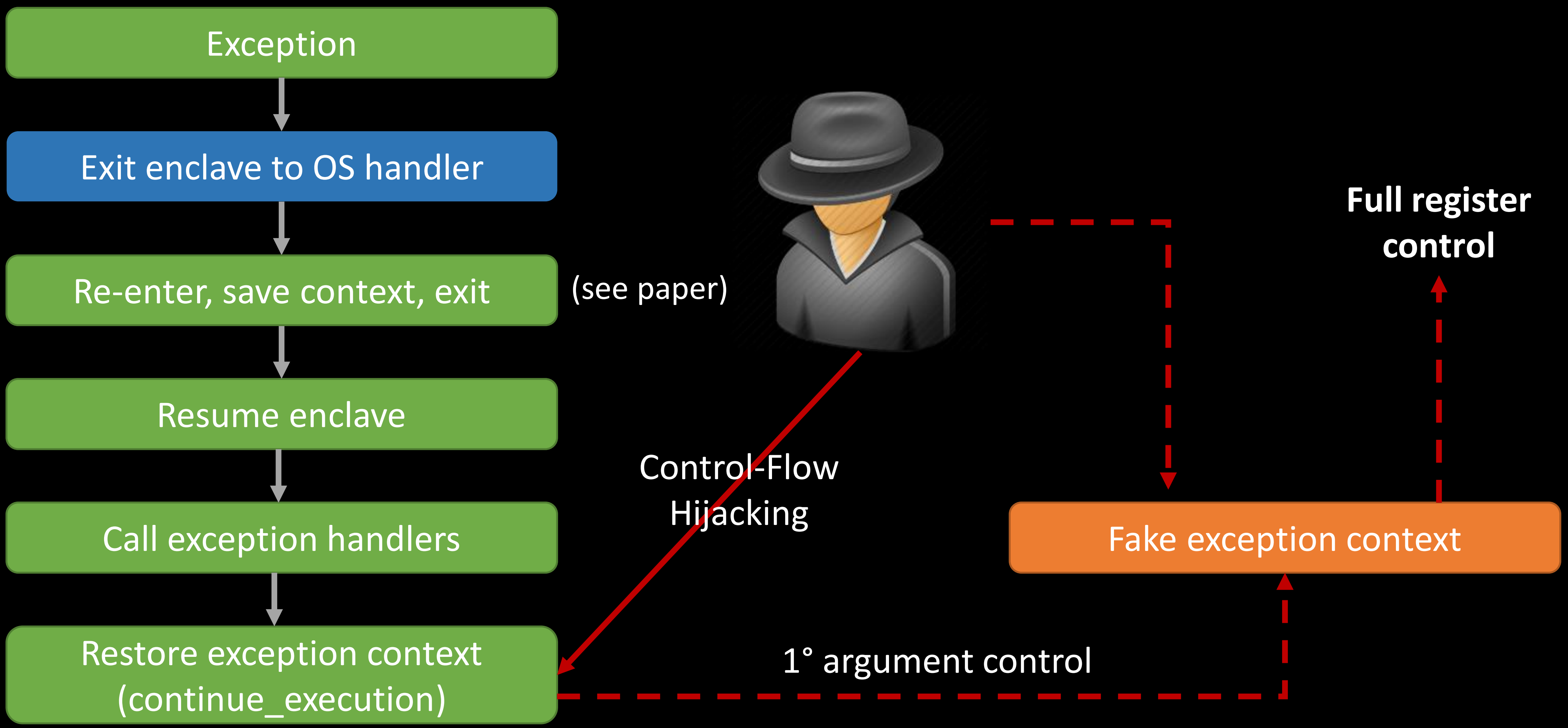
The Basic Idea



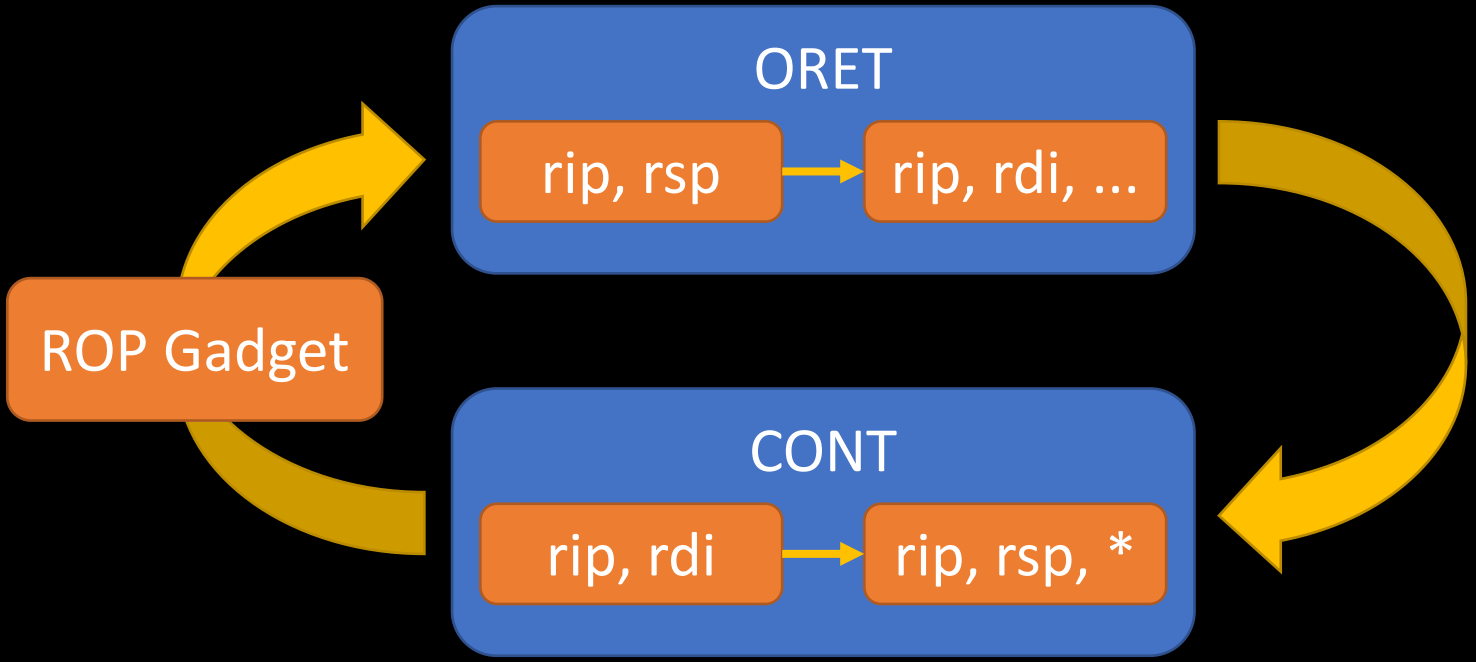
The ORET Primitive



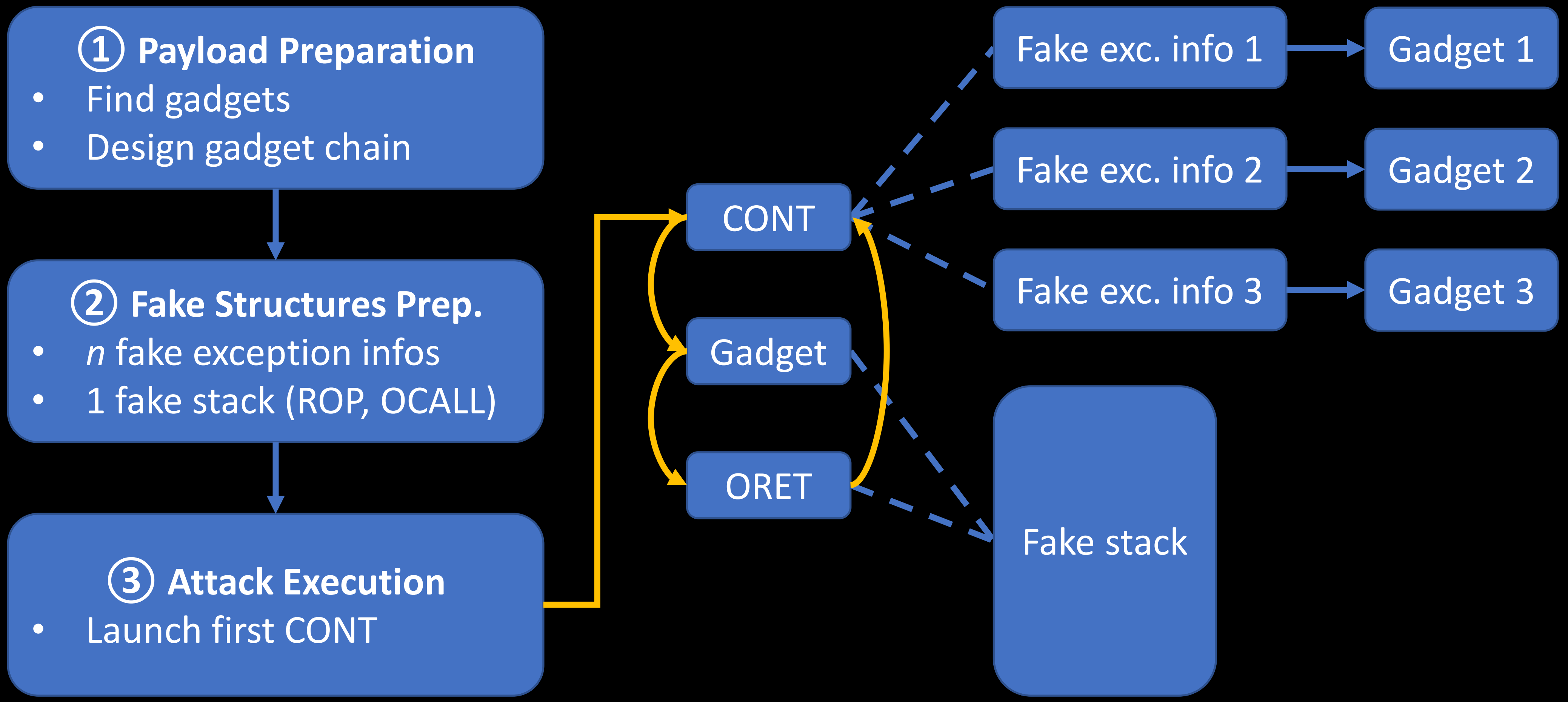
The CONT Primitive



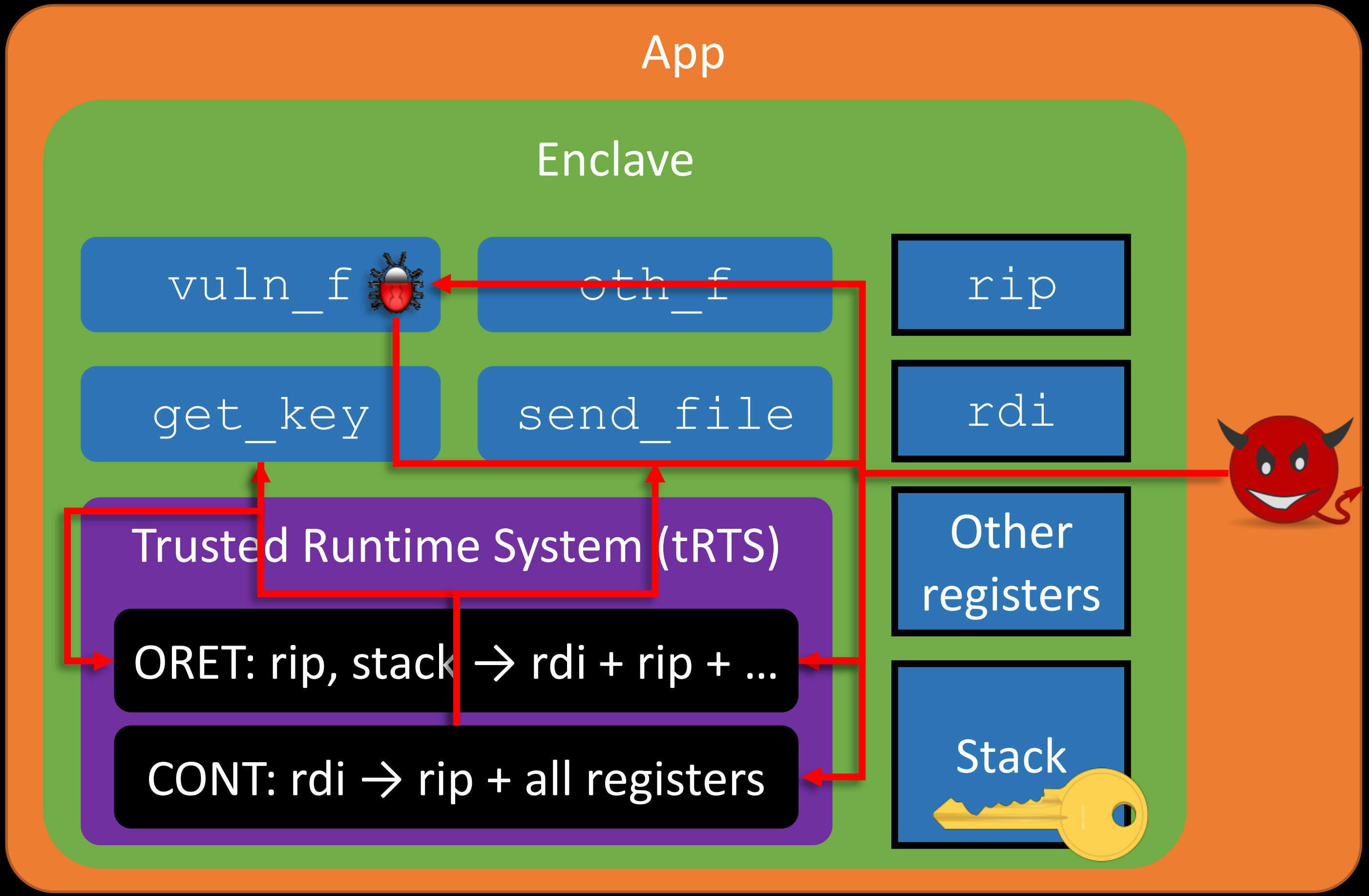
The ORET+CONT Loop



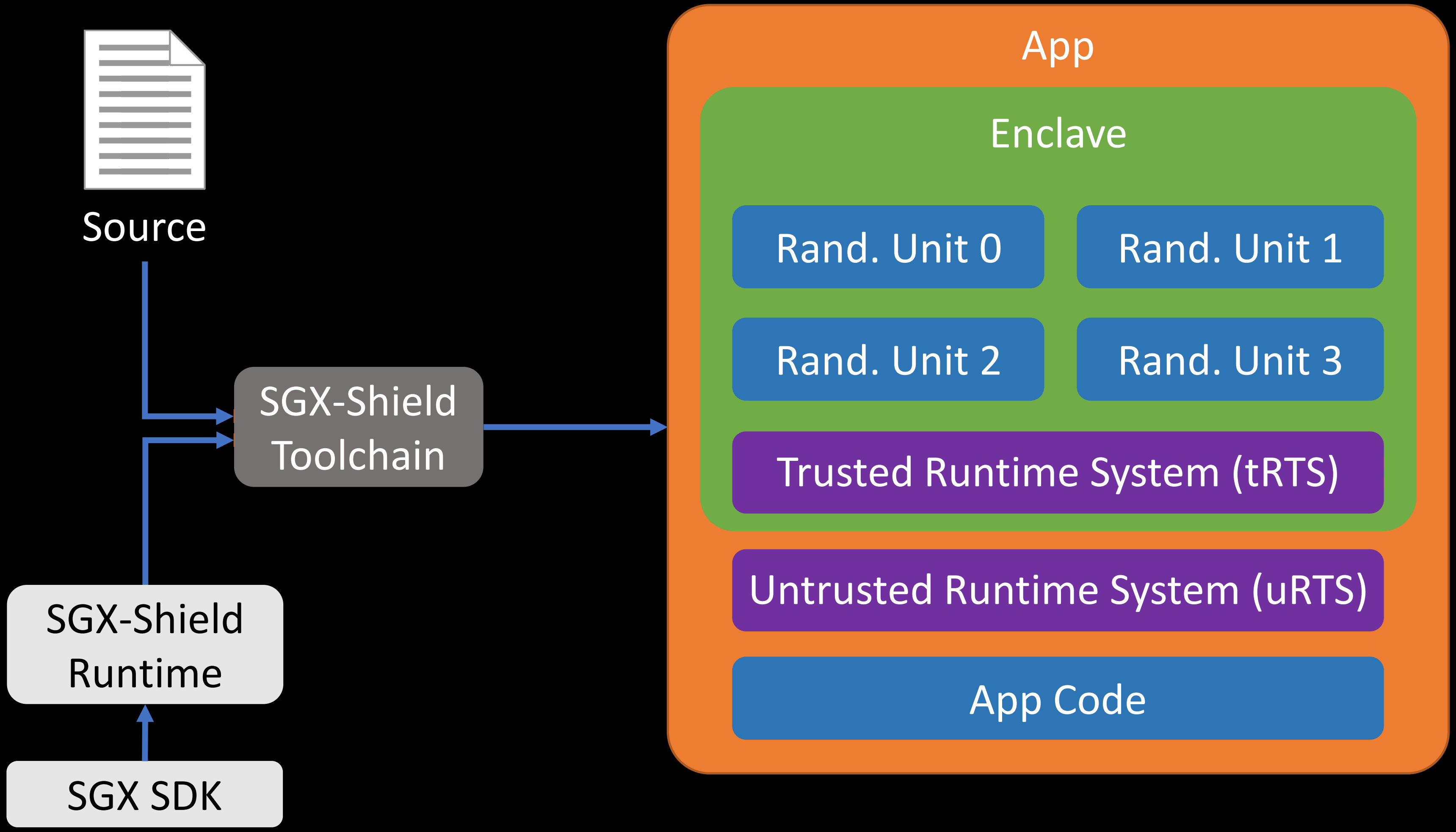
Attack Overview



Example Attack



SGX-Shield [Seo et al., NDSS 2017]



Fine-grained code randomization

tRTS is not randomized

Attacking SGX-Shield

Fine-grained code randomization

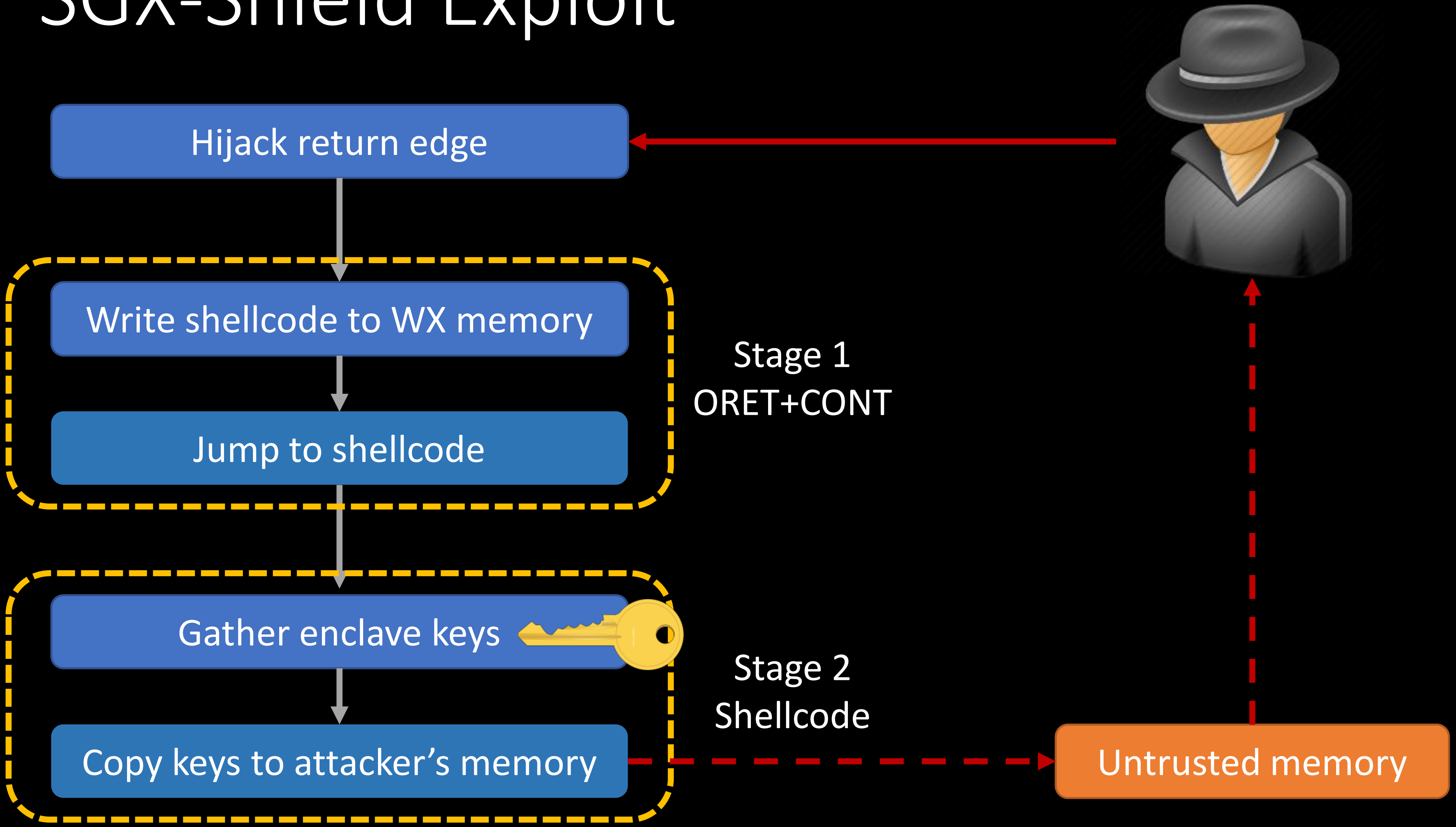
Reusing tRTS code (not randomized)

Coarse-grained Control Flow Integrity

Return edges are not properly instrumented
→ ORET is possible

Other mitigations (SFI, W⊕X) assume CFI

SGX-Shield Exploit



Mitigations

SDK Hardening

Secret canaries in contexts

Mangling context data

External Hardening

Randomization of SDK code

Stronger CFI

Lessons Learned

- SGX presents significant hardening challenges
  - Strong attacker
- The SDK can increase an enclave’s attack surface
  - Powerful code-reuse primitives
  - Low-level code hidden from sight

Conclusion

- We presented a novel code-reuse attack on Intel SGX
- Using «forgotten» code to bypass SoA hardening
- Underlines the need to consider implications of SDK usage

Questions?