KubeCon | CloudNativeCon Europe 2022

Attacking & Defending Kubernetes TEE Enclaves in Critical Infrastructure

Robert Ficcaglia, Co-Chair Kubernetes Policy WG

KubeCon | CloudNativeCon Europe 2022

Robert Ficcaglia

Co-Chair Kubernetes Policy Workgroup

Kubernetes SIG-Security Volunteer

CNCF Security TAG Volunteer

Day jobs have included designing spacecraft, cryptographic software, hardware, firmware, drivers, APIs, formal verification of systems, >1B user systems, >25 years of private and public sector engineering.

rficcaglia@gmail.com @rficcaglia

Fan of Flamenco, fino and fideuà!

What is a TEE?

“Confidential Computing is the **protection of data in use** by performing computation in a hardware-based *T*rusted *E*xecution *E*nvironment.”

“These [TEEs] **prevent unauthorized access or modification** of applications and data **while they are in use**.”

“If you can’t trust hardware, the kernel, the OS, operators…all bets are off!!!”

- Everyone at some point in threat modeling

Who really needs this?

Use Cases

- Scalable replacement for dedicated HSM
- Cloud Key Management Services/APIs
- Compliance - e.g. process private data, zero knowledge proofs
- Regulatory - utilize clouds that may be untrusted
- Smart Contracts, Blockchain, Proprietary Algorithms (ML)
- IoT, Edge, Vehicles

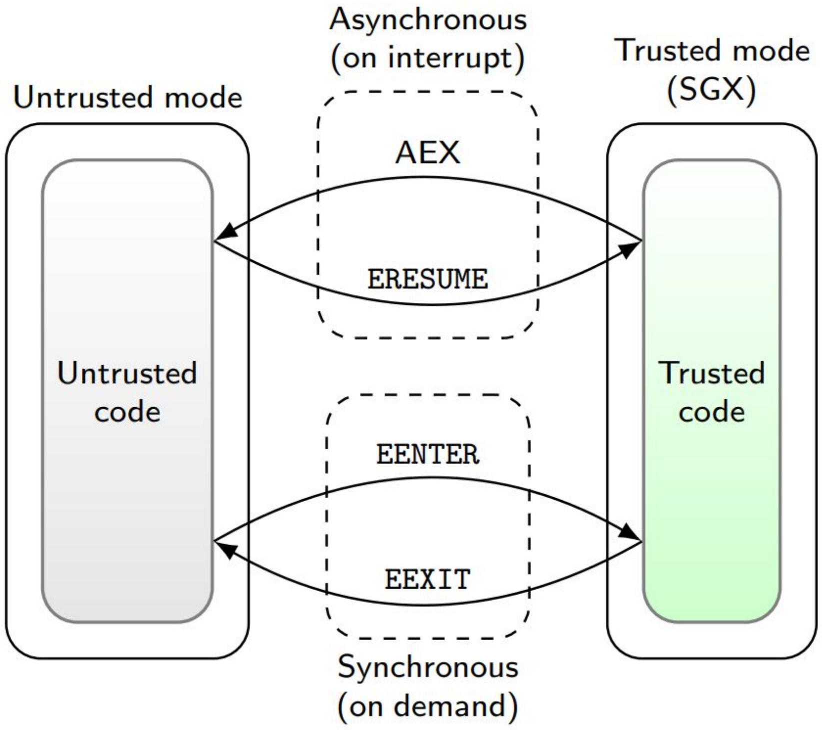
"[TEEs] **prevent unauthorized access or modification** of applications and data **while they are in use**."

How?

“Hardware … remove the operating system and device driver vendors, platform and peripheral vendors, and service providers and their admins, from the list of required trusted parties, thereby reducing exposure to potential compromise” [1]

Deeper Dive

Privacy and isolation of data ***in use*** relies on trusted hardware (and μcode) and special memory management on CPUs (eg. MEE)



Authenticated

Data Integrity

Data Confidentiality

Isolation

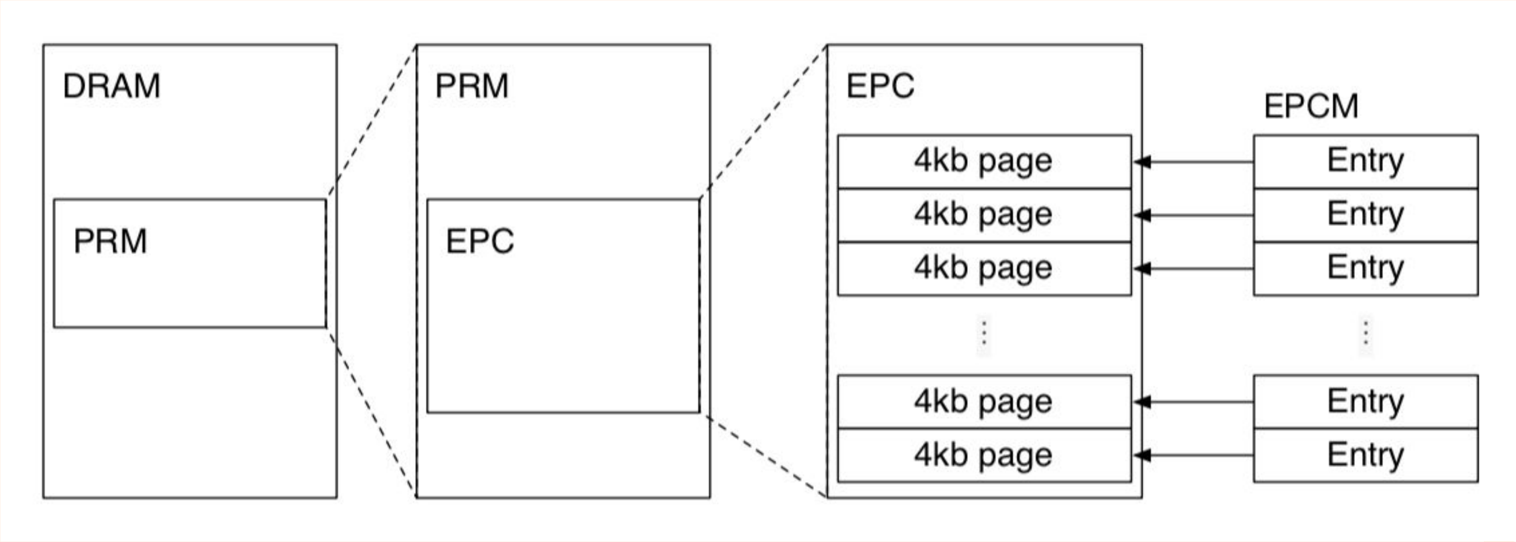
Attestability

Fine Print: Code confidentiality? Performance? Availability? Scalability ? Anonymity?

Example Implementation (SGX)

Enclave Page Cache (EPC) - memory pages for an “enclave” in BIOS-reserved region of physical memory

Access via SGX instructions

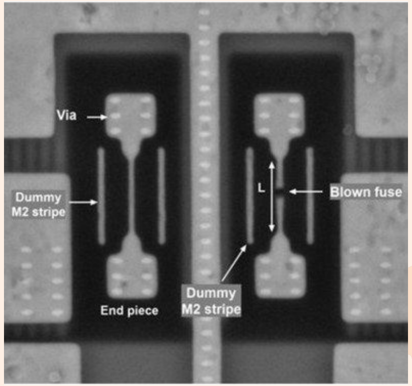


Measurement and Attestation

Code Integrity — hashing and asymmetric encryption

Enclave TCB produces a hash-based “measurement” of the initial state (eg. code, private heap and stack) in all attested statements.

Relies on the trust of special pre-provisioned “architectural” enclaves - keys* hard coded in CPU to verify these at runtime (eg eFuse)



*Root keys are not directly accessed - key derivation methods used

Init() Securely

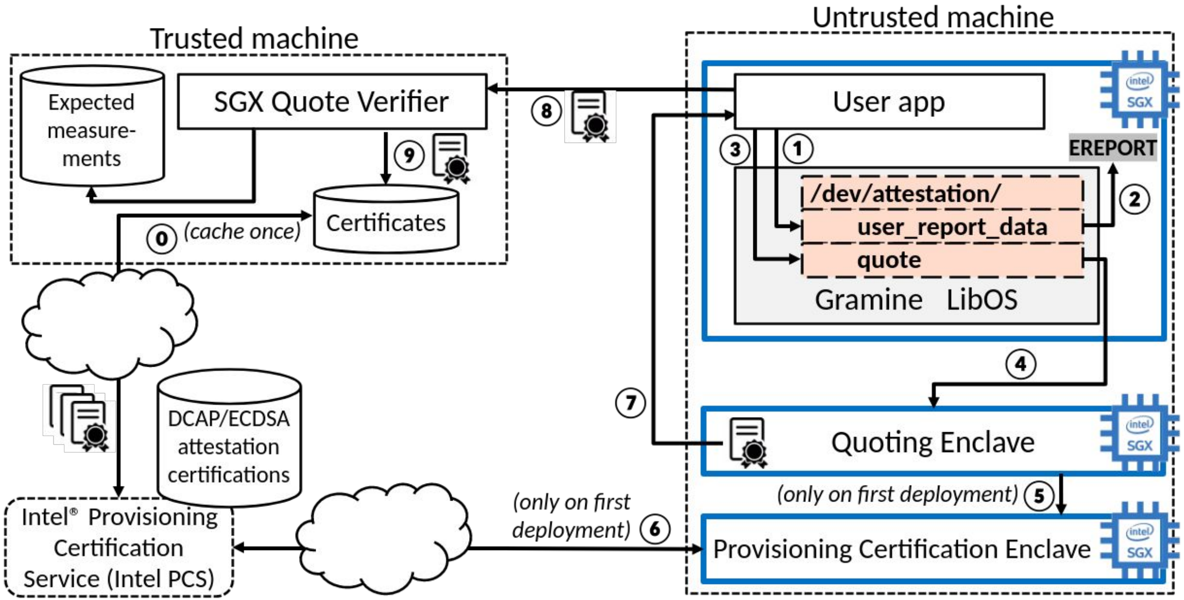
Code, data cleartext before enclave initialization

Secrets/keys and data come from the outside

Enclave must prove that it is legitimate and untampered

Local Attestation: two TEEs run on the same physical machine

Remote attestation: a TEE running on a remote physical machine.



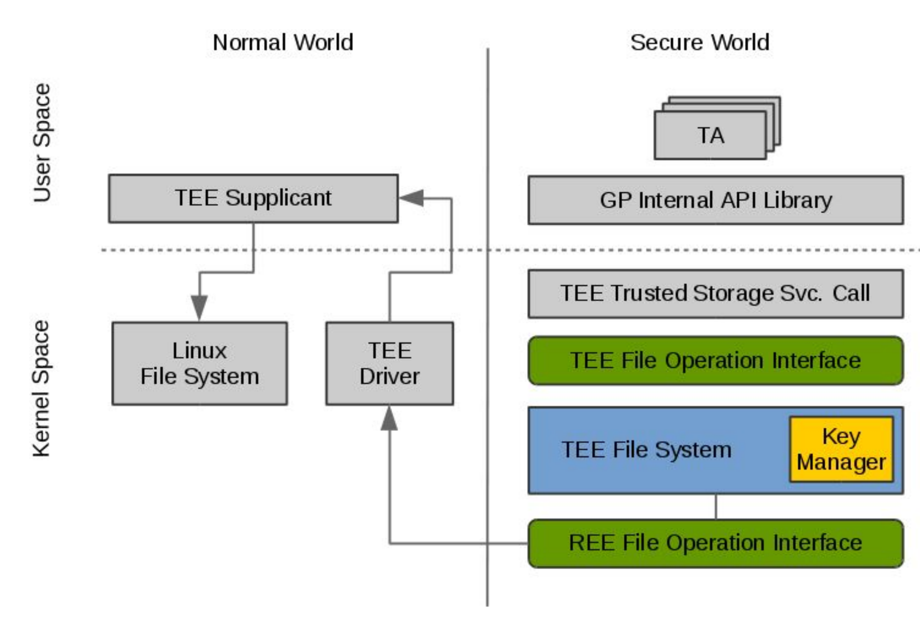
Data Persistence Outside TEE

Running enclave code, data ephemeral.

Encrypt data as it exits the enclave

Encryption key specific to enclave

Sealing (SGX Specific)
[OP-TEE](#) (ARM)



How to DevOps?

3 emerging deployment options:

1. Carefully partitioning application code into trusted and untrusted components and marshall calls between these
2. Deploy unmodified applications into TEE-resident libOS processes or containers
3. Cross-TEE portable application bytecode

What would El Profesor do?

A) A direct assault on the boundary security?

B) Bribe, coerce internal actors in every supply chain?

C) Use the system features?

D) Use the system’s defenses?

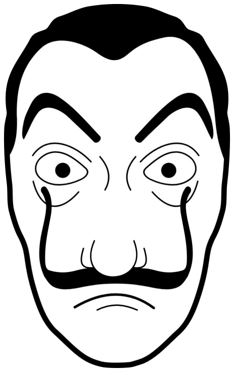
E) Target the human operators?

F) Inject chaos into the system and operator pathways?

G) Misdirect and intercept signals - replace them with fakes?

H) Expect countermeasures to all of the above?

I) **PATIENTLY RESEARCH AND TEST ALL OF THE ABOVE**



Every Feature Holds a Key

CPU, Cache, Branch Predictor

DRAM

Bus, Board, BIOS

HW BOM

Firmware/µcode

Kernel

OS

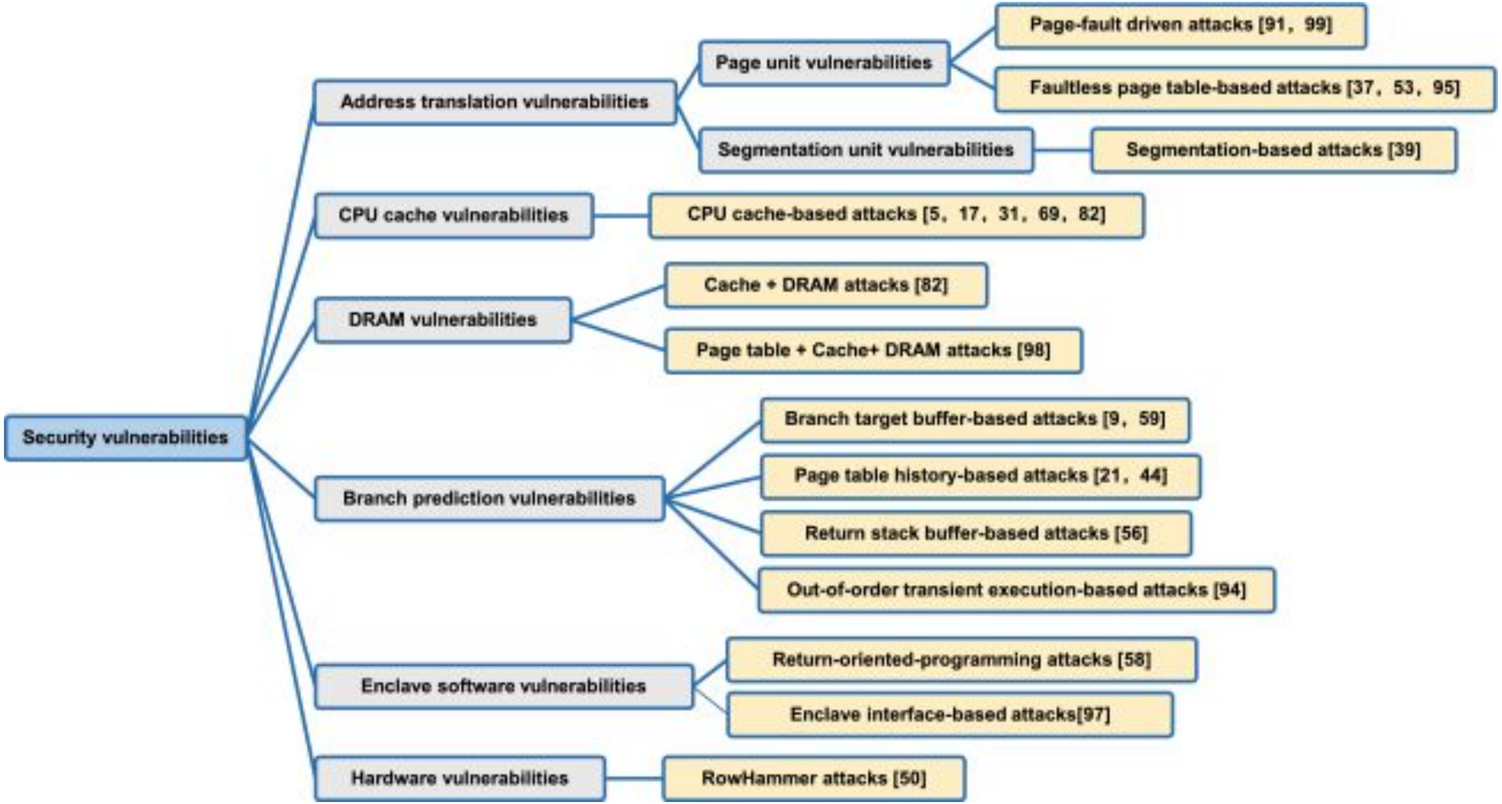
Drivers

SBOM

Compiler features

Side Channels…V=IR

Threats


Security vulnerabilities
Address translation vulnerabilities
Page unit vulnerabilities
Page-fault driven attacks [91, 99]
Faultless page table-based attacks [37, 53, 95]
Segmentation unit vulnerabilities
Segmentation-based attacks [39]
CPU cache vulnerabilities
CPU cache-based attacks [5, 17, 31, 69, 82]
DRAM vulnerabilities
Cache + DRAM attacks [82]
Page table + Cache+ DRAM attacks [98]
Branch prediction vulnerabilities
Branch target buffer-based attacks [9, 59]
Page table history-based attacks [21, 44]
Return stack buffer-based attacks [56]
Out-of-order transient execution-based attacks [94]
Enclave software vulnerabilities
Return-oriented-programming attacks [58]
Enclave interface-based attacks[97]
Hardware vulnerabilities
RowHammer attacks [50]

Trust

BigCo Brand Name?

Assessment by third-party evaluation laboratories?

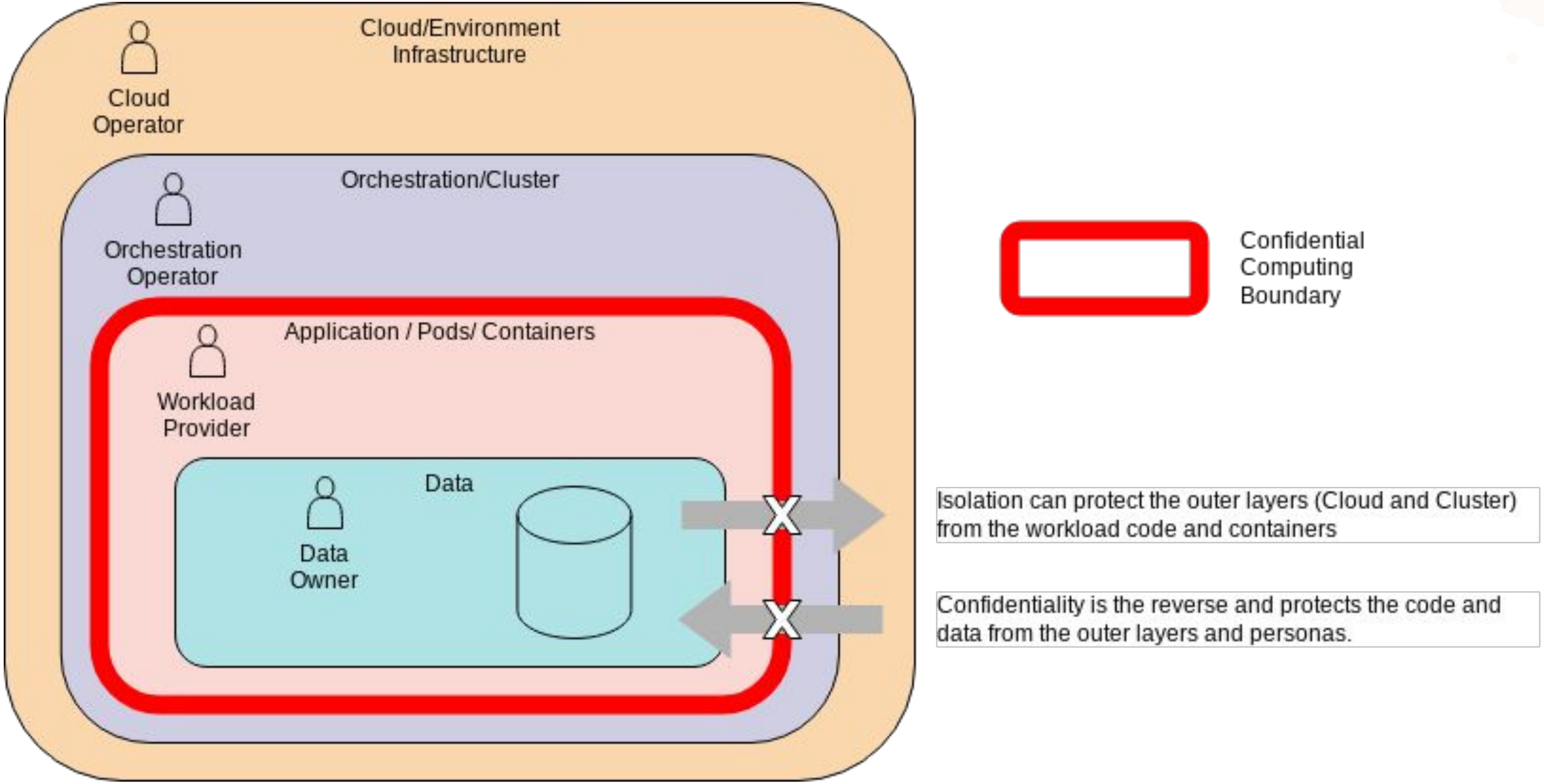
Open source hardware, firmware, and software?

Papers by academic or standards bodies?

HW TEEs - A Silver Bullet?

"Security is only as strong as the layers below it, since security in any layer of the compute stack could potentially be circumvented by ~~a breach~~ at an underlying layer. This drives the need for security … down to the silicon." [1]

Where Does Kubernetes Fit into this?



Kubernetes TEE Dev(Sec)Ops Tasks

- Develop container image and enclave compatible app
- Attestation and Secret Provisioning
- Runtime image verification
- Secure Data Persistence
- Do multiple TEE containers communicate with each other?
- Workload placement and management in cluster?
- Migration of workloads between TEE environments?
- Integrity failure monitoring? Availability? DoS?
- App, Image, TEE SDK and open source patch management?
- Revocation of keys or attestation certs?
- Firmware or hardware updates required?
- Kubernetes release compatibility?
- Monitoring which apps are loaded in which enclaves on which nodes?
- Detecting an enclave creating other enclaves?
- Testing?
- Enclave error handling/lock down?
- Distributing to other groups?
- Performance monitoring and tuning?
- I give up…

(Some) Open Source TEE Help

Each Take a different approach to Learning Curve, Ease of Use, Performance, Code Migration Requirements, I/O

CONFIDENTIAL CONTAINERS

MarbleRun

OpenEnclave - Write Code

✔Enclave creation and management

Function calls to manage the lifecycle of an enclave within your application

✔Enclave measurement and identity

Hash of attributes and the position, content and protection of memory pages. Two enclaves with the same hash are identical. Also the hash of the public key of the author.

✔Communication

Mechanisms for defining call-ins and call-outs and the data marshalling associated with them

✔System primitives

System primitives exposed by enclave runtime, such as thread and memory management

✔Sealing

Functions to support persistence of secrets

✔Attestation

Functions to support verification of identity

✔Runtime and cryptographic libraries

Pluggable libraries to provide the necessary language and cryptographic support within an enclave

LibOS TEE - Wrap a Process

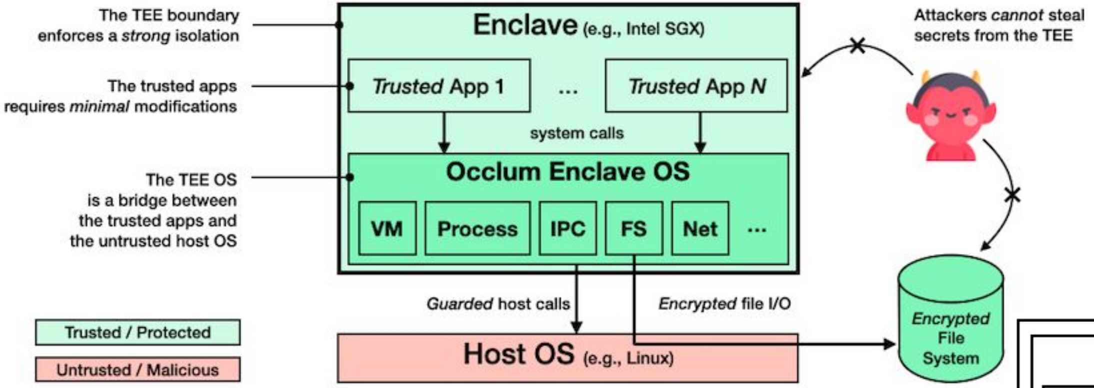
Occlum light-weight LibOS processes share the same SGX enclave; legacy applications run on SGX with little to no modifications to source code - run in pod (requires SGX DCAP driver and Intel FSGSBASE enablement patch)

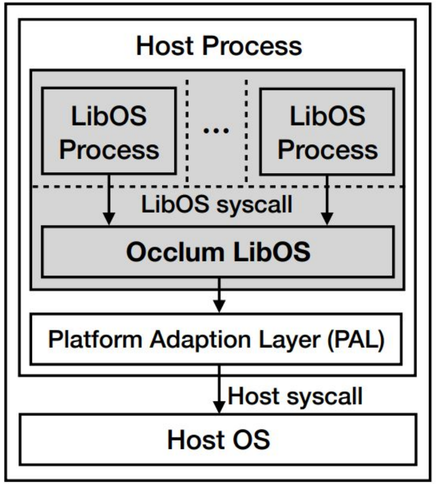
FINE PRINT: "In the current implementation, the customer binaries only be signed with signing tool. It is not verified. So user should take the responsibly to make sure the binaries are good citizens." [2]

Trampoline code *("a piece of carefully-written assembly, which performs sanity checks")* that jumps to the entry point of the LibOS [3],[4], [5] ie is it atomic? [8] - relies on "verifier" (not formally verified [6]), also the Verifier scheme relies on MPX but *"MPX is not supported by Intel anymore."*

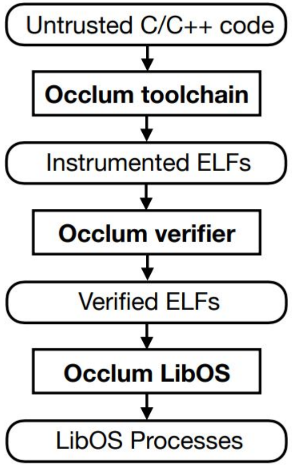
Gramine (formerly called Graphene) is a lightweight library OS, designed to run a single application with minimal host requirements. Containers via **Gramine Shielded Containers** (GSCs).

LibOS (continued)





(a) Occlum LibOS



(b) Occlum workflow

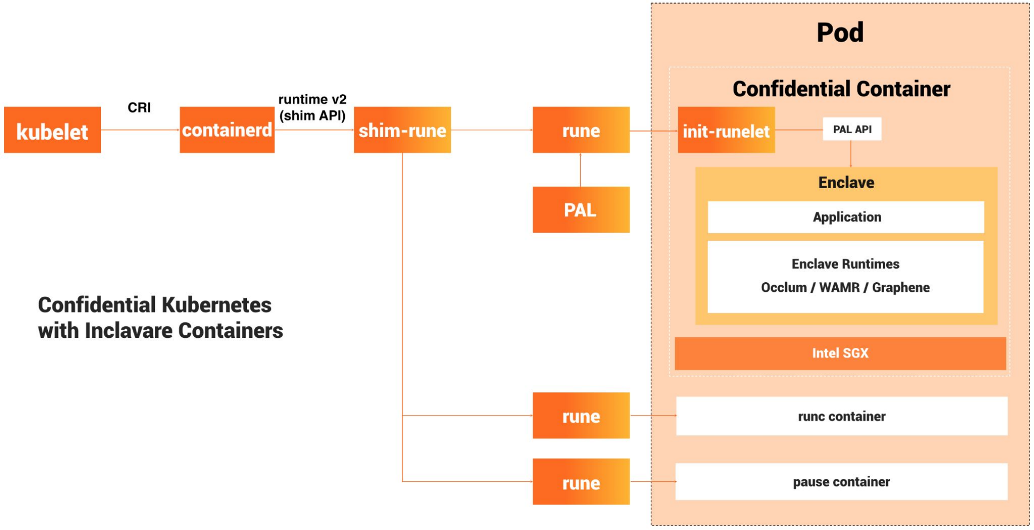
Inclavare - TEE Containers

Container applications can run in the TEE

Enclave runtime is responsible for loading and running applications

Interface between containers and enclave runtime is [Enclave Runtime PAL API]() (through runE)

Currently supports occlum, WAMR, and gramine [1]-[3]



MarbleRun - Deploy all the Things

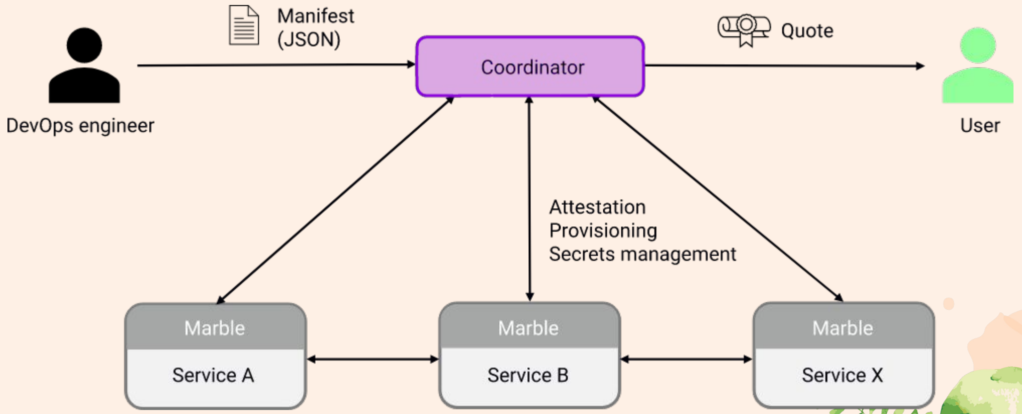
Control plane designed to run TEE containers (marbles) on Kubernetes

Manages secrets

Establishing enclave-to-enclave mTLS (CA).

No change to tools or application code

Remote Attestation manifest defined in JSON



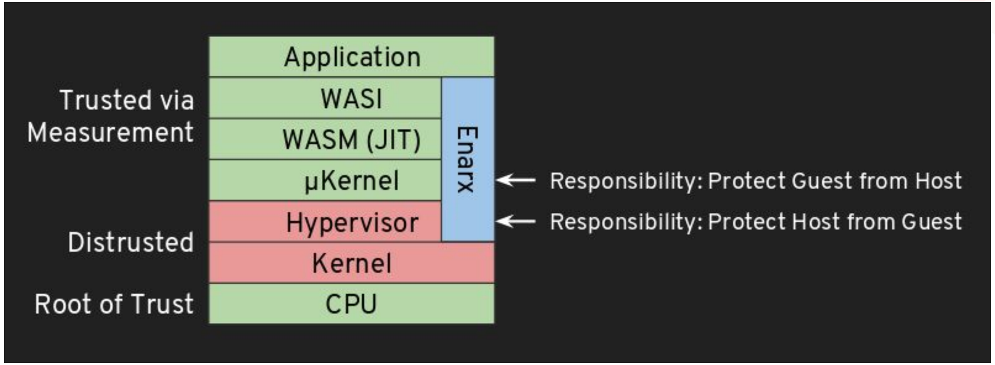
https://github.com/edgelesssys/marblerun

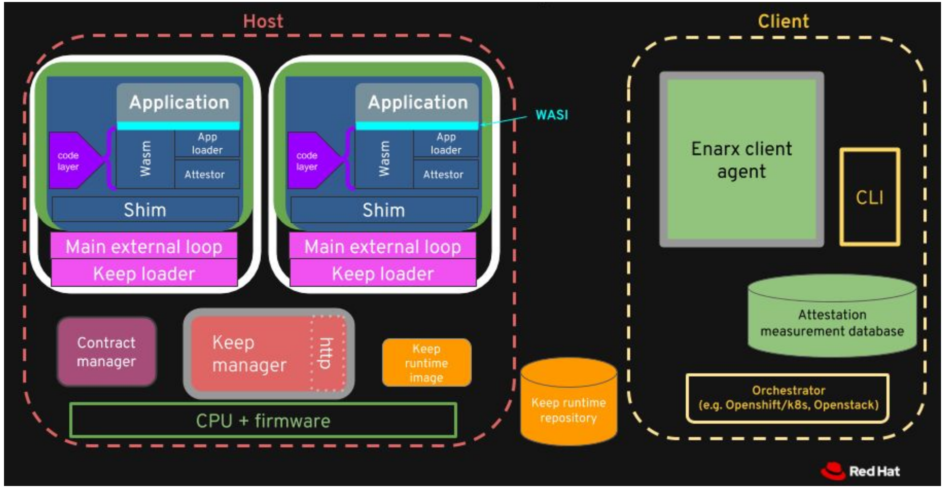
And Now For Something Completely Different - Enarx

Support for Intel platforms (SGX), AMD platforms (SEV)

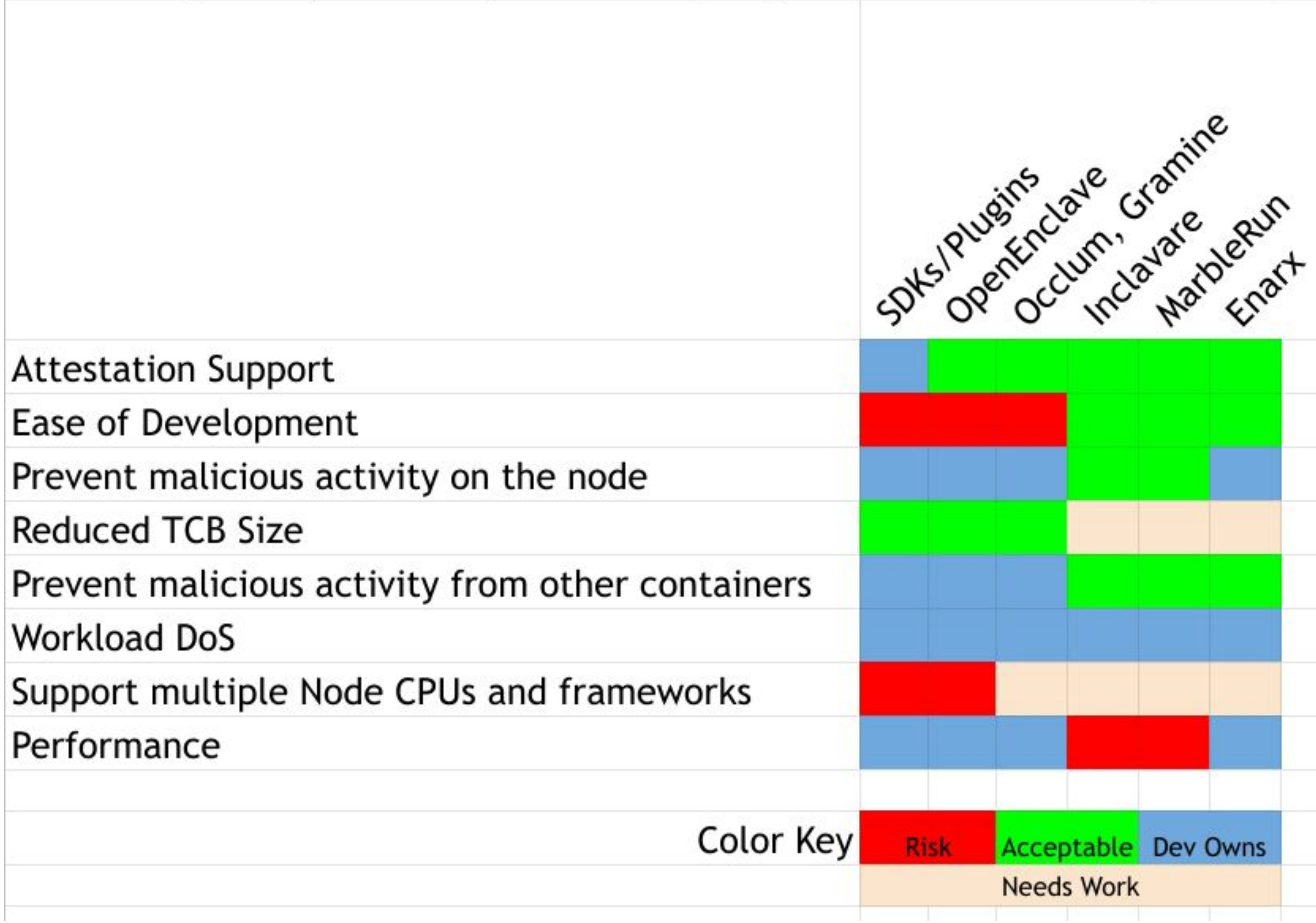
TEEs (Keeps) based on WebAssembly

1. Virtual Memory Manager
2. microkernel (µkernel)
3. WASM runtime
4. WASI implementation





Project Capabilities (not attesting to production readiness)



How to Defend?

Attack surface minimization

Formal Verification of (critical parts of) TCB

Iago attacks - software supervisor (check buffer sizes, malicious pointers, etc)

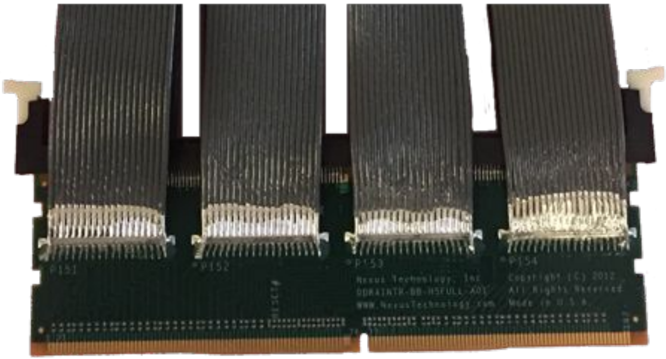
Mo’ better memory page layout control - ROP jails, JOP landing pads

Control-Flow Integrity/Graph?

Prevent malicious tenant abuse of node EPC

TDX? - multi-key memory encryption (TME-MK)

- Use of a specific key for a page of memory
- Allows either CPU-generated keys or tenant-provided keys
- Containers can be encrypted separately
- Support on AKS [1]



TEEs Can Aid Compliance

NIST 800-53 Rev 5 High Baseline Controls

CM-8(3) - Automated Unauthorized Component Detection

SC-3 - Security Function Isolation

SC-4 - Information in Shared System Resources

SC-7(21) - Isolation of System Components

SC-39 - Process Isolation

SI-16 - Memory Protection

Más Compliance

Additional Controls

PL-8(1) - Security and Privacy Architectures | Defense in Depth

SA-8(5) - Efficiently Mediated Access

SA-8(6) - Security and Privacy Engineering Principles | Minimized Sharing

SA-8(7) - Security and Privacy Engineering Principles | Reduced Complexity

SA-8(13) - Security and Privacy Engineering Principles | Minimized Security Elements

SA-8(26) - Performance Security

SA-17(1) - Developer Security and Privacy Architecture and Design | Formal Policy Model

SA-17(2) -Developer Security and Privacy Architecture and Design | Security-relevant Components

SC-3(1) - Security Function Isolation | Hardware Separation

SC-3(5) - Security Function Isolation | Layered Structures

SC-7(20) - Boundary Protection | Dynamic Isolation and Segregation

SC-34 -Non-modifiable Executable Programs

SC-39(1) - Process Isolation | Hardware Separation

SC-39(2) - Process Isolation | Separate Execution Domain Per Thread

¿A dónde vamos ahora?

For a High Baseline Kubernetes project I would …

If I had to attack that solution I would …

Follow the progress of a real world build-out and attack lab @SunStoneSecure

KubeCon | CloudNativeCon

Europe 2022

THANK YOU!

rficcaglia@gmail.com

Appendix: Honorable Mentions (aka I ran out of Time)

Teaclave - TEE aware function-as-a-service platform.

Confidential Containers Operator and https://github.com/confidential-containers/enclave-cc

SEV Kata Containers

Mystikos - musl runtime and a set of tools for TEEs

Keystone - Keystone is an open-source project for building trusted execution environments (TEE) with secure hardware enclaves, based on the RISC-V architecture.