



Owned or pwned? no peekin' or tweakin'!

By Richard Zak and Nick Vidal



TABLE OF CONTENTS



01

Confidential Computing

02

Hardware

03

Enarx

04

Demo



Confidential Computing



Confidential Computing

“Confidential Computing protects data in use by performing computation in a hardware-based Trusted Execution Environment. These secure and isolated environments prevent unauthorized access or modification of applications and data while in use, thereby increasing the security assurances for organizations that manage sensitive and regulated data.”

✦ Protections for Data...

At Rest

File encryption on
HD/SSD or Cloud
Storage



In Use

CPU/Memory (TEEs)



In Transit

HTTPS (Browser ↔ Server)
End-to-End





Data Protection



Data Integrity



Data Confidentiality



Code Integrity



Code Confidentiality

Sectors



Defense



Banking/Finance



Healthcare



Government



Customer Data



Telecom

Advantages



No Peekin'

Data & Code Confidentiality



No Tweakin'

Data & Code Integrity

**Even if host is compromised
Internal/external threats**



Hardware

Architectures



Process Based

Intel SGX
RISC-V Sanctum



VM Based

AMD SEV
IBM PEF
ARM Realms
Intel TDX



CPUs (early gen)



2012

ARM TrustZone

Secure OS
(Two Worlds)



2015

Intel SGX

Isolated Apps
(Enclave)



2020

AMD SEV

Isolated VMs



2021

IBM PEF

Isolated VMs



CPUs/GPUs (next gen)



AMD SEV-SNP
EPYC Milan



Intel SGX
Xeon Ice Lake



NVIDIA
H100 Hopper



Cloud: next gen general availability



Azure



Google Cloud





Enarx

Enarx



Open Source

Leading framework for running applications in TEEs (Trusted Execution Environments).



Easy Deployment

Provides a run-time TEE based on WebAssembly, allowing developers to deploy apps from various languages



Hardware Neutral

CPU-architecture independent, letting developers deploy the same app binary transparently across multiple targets, different architectures.

WebAssembly



Security

Its sandbox model offers an extra layer of protection, isolating the application from the host.



Performance

It's statically typed and designed to be encoded in a size- and load-time-efficient binary format highly optimized for performance.



Portability

It's designed to be executable efficiently on a variety of operating systems and instruction set architectures, on the Web and off.

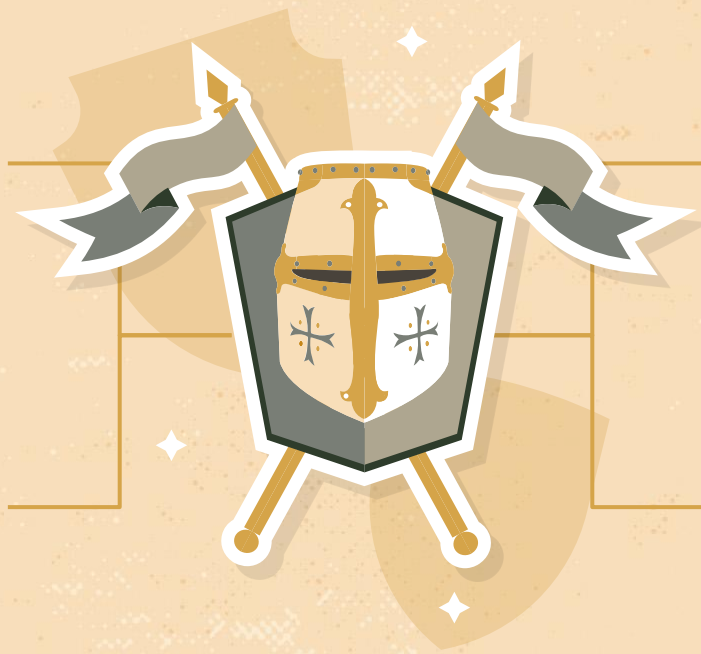
Enarx Architecture

Enarx Keep

Sets up the TEE & runs the app

Drawbridge

Acts as a repo for WebAssembly apps



Steward

Performs attestation & provides crypto cert

Host

Hosts app & data, but is untrusted

Security Principles of Enarx

1. Minimal Trusted Computing Base
2. Minimum trust relationships
3. Deployment-time portability
4. Network stack outside TCB
5. Security at rest, in transit, and in use
6. Auditability
7. Open source
8. Open standards
9. Memory safety
10. No backdoors



Demo

Try Enarx @ try.enarx.dev



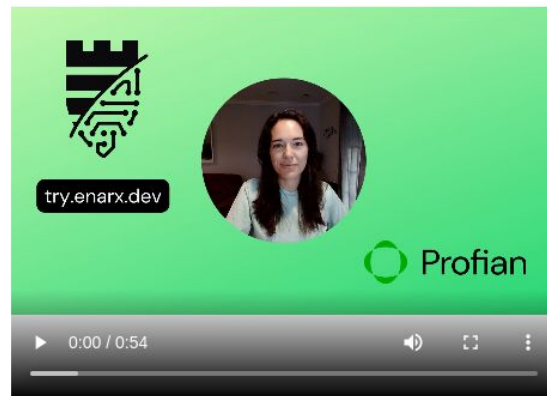
 **Enarx** secured by  **Profian**

 [GitHub](#)  [Chat](#)  [Blog](#)  [Twitter](#)

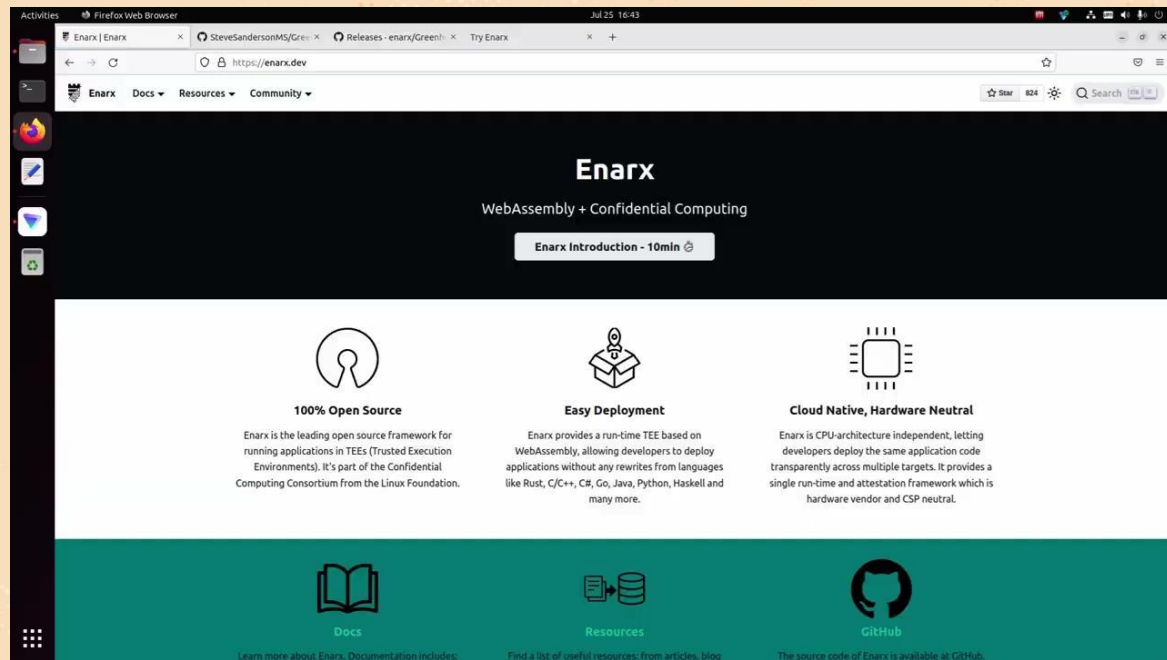
Try Enarx

Run a **Wasm** workload using **Confidential Computing**.
Watch the video and start now or choose a **platform**.

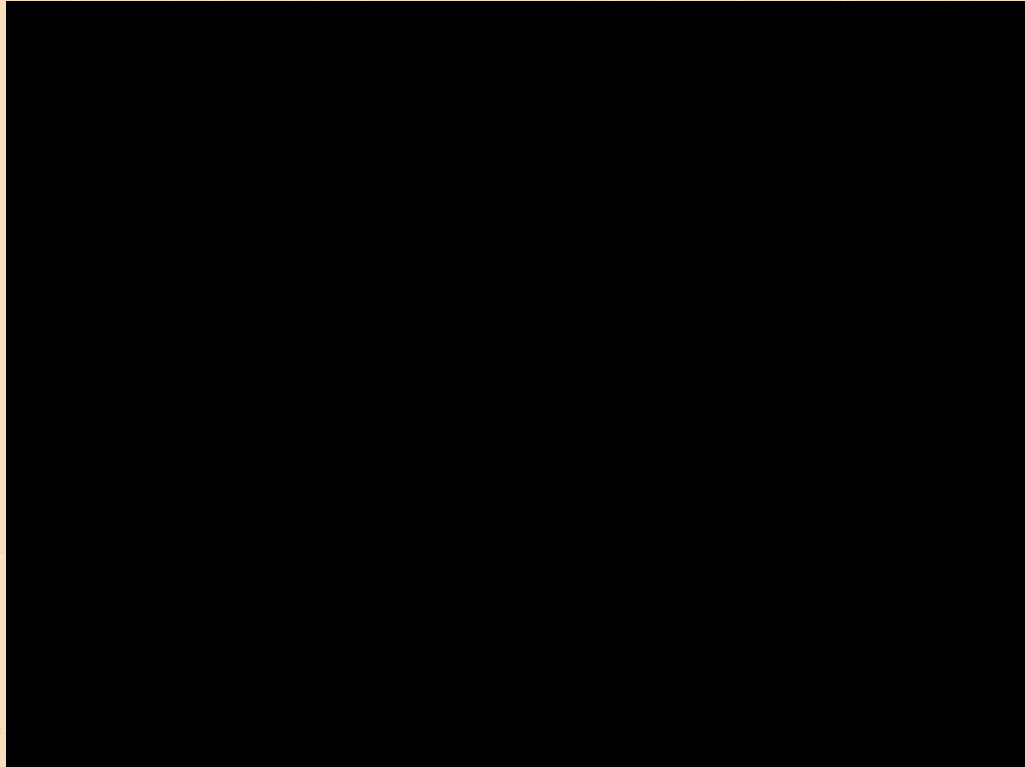
[Start now](#)



Greenhouse Monitor



Cryptle



The image is a promotional banner for the 'Cryptle Hack Challenge'. It features a central white banner with the word 'Cryptle' in a large, dark blue, stylized font. Below this, a green ribbon contains the words 'Hack Challenge' in a white, serif font. The background is a textured, light orange color. Decorative elements include a large, faint shield shape in the background, a small white star in the top left, a stylized grey and white shield with a cross in the top right, and a stylized orange and white shield with a cross in the bottom left. There are also several small white stars scattered throughout the background.

Cryptle

Hack Challenge



THANKS!

Please star our project:
github.com/enarx/enarx



CREDITS: This presentation template was created by Slidesgo,
including icons by Flaticon, and infographics & images by Freepik



Why WebAssembly?

1. Minimal Trusted Computing Base
2. Minimum trust relationships
3. Deployment-time portability
4. Network stack outside TCB
5. Security at rest, in transit and in use
6. Auditability
7. Open source
8. Open standards
9. Memory safety
10. No backdoors

Why Rust?

1. Minimal Trusted Computing Base
2. Minimum trust relationships
3. Deployment-time portability
4. Network stack outside TCB
5. Security at rest, in transit and in use
6. Auditability
7. Open source
8. Open standards
9. Memory safety
10. No backdoors

Why Open Source?

1. Minimal Trusted Computing Base
2. Minimum trust relationships
3. Deployment-time portability
4. Network stack outside TCB
5. Security at rest, in transit and in use
6. Auditability
7. Open source
8. Open standards
9. Memory safety
10. No backdoors