Security and confidential computing

axel simon
office of the CTO

enarx.io

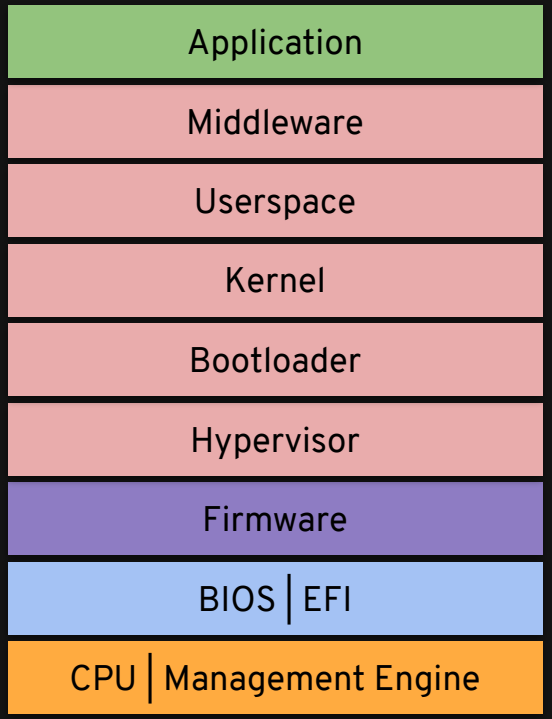
Red Hat

The Problem

The Need for Confidentiality and Integrity

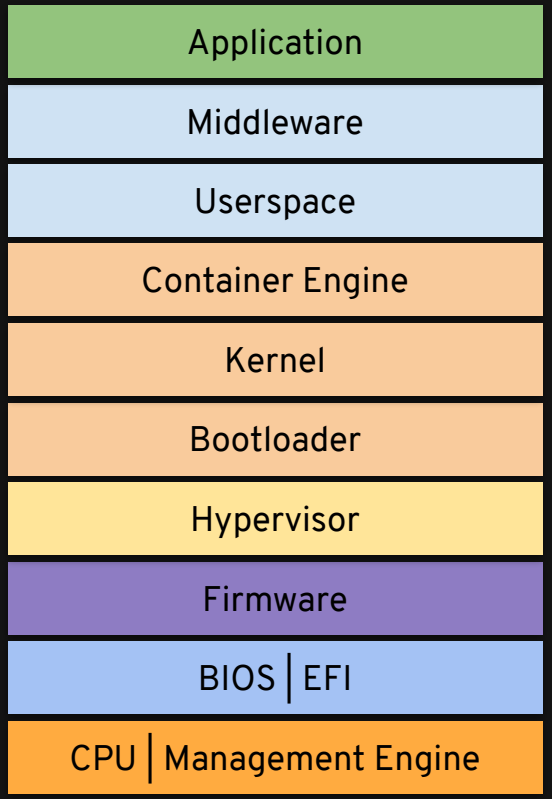
- Banking & Finance
- Government & Public Sector
- Telco
- IoT
- HIPAA
- GDPR
- Sensitive enterprise functions
- Defense
- Human Rights NGOs
- ...

Virtualization Stack



Red Hat

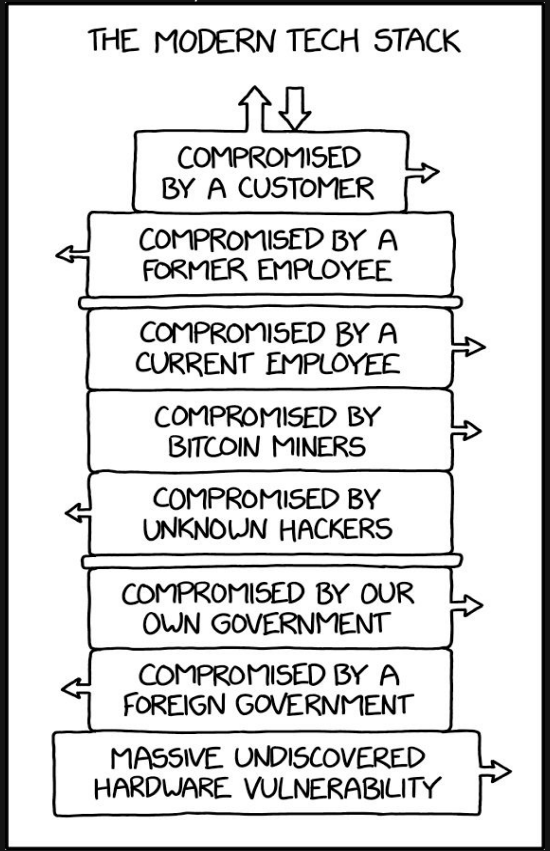
Container Stack



Red Hat

Virtualization Stack

as seen by xkcd (xkcd.com/2166)

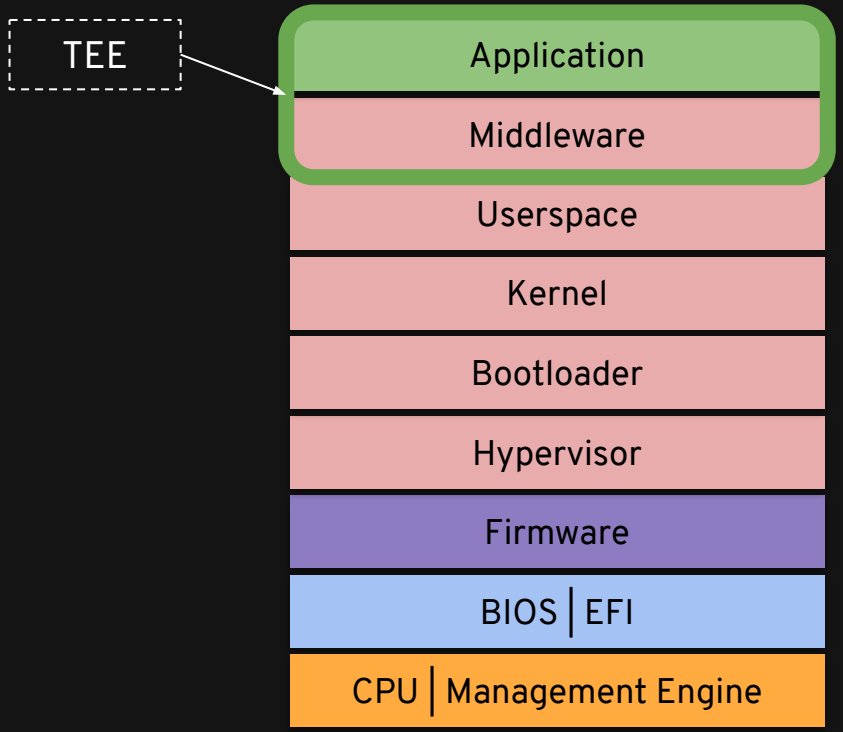


Red Hat

Trusted Execution Environments

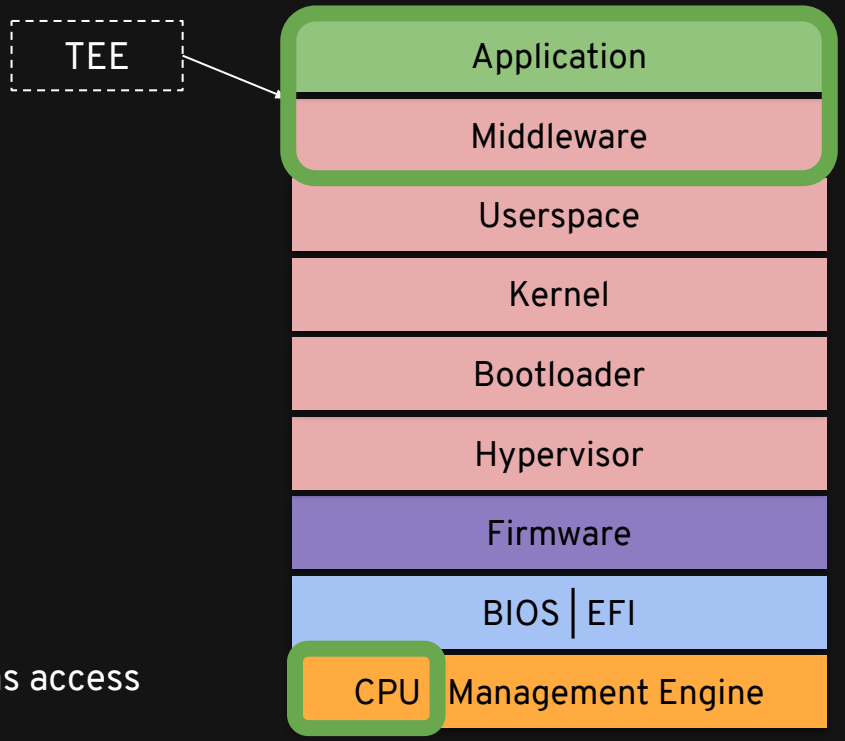
Red Hat

What's a TEE?



Red Hat

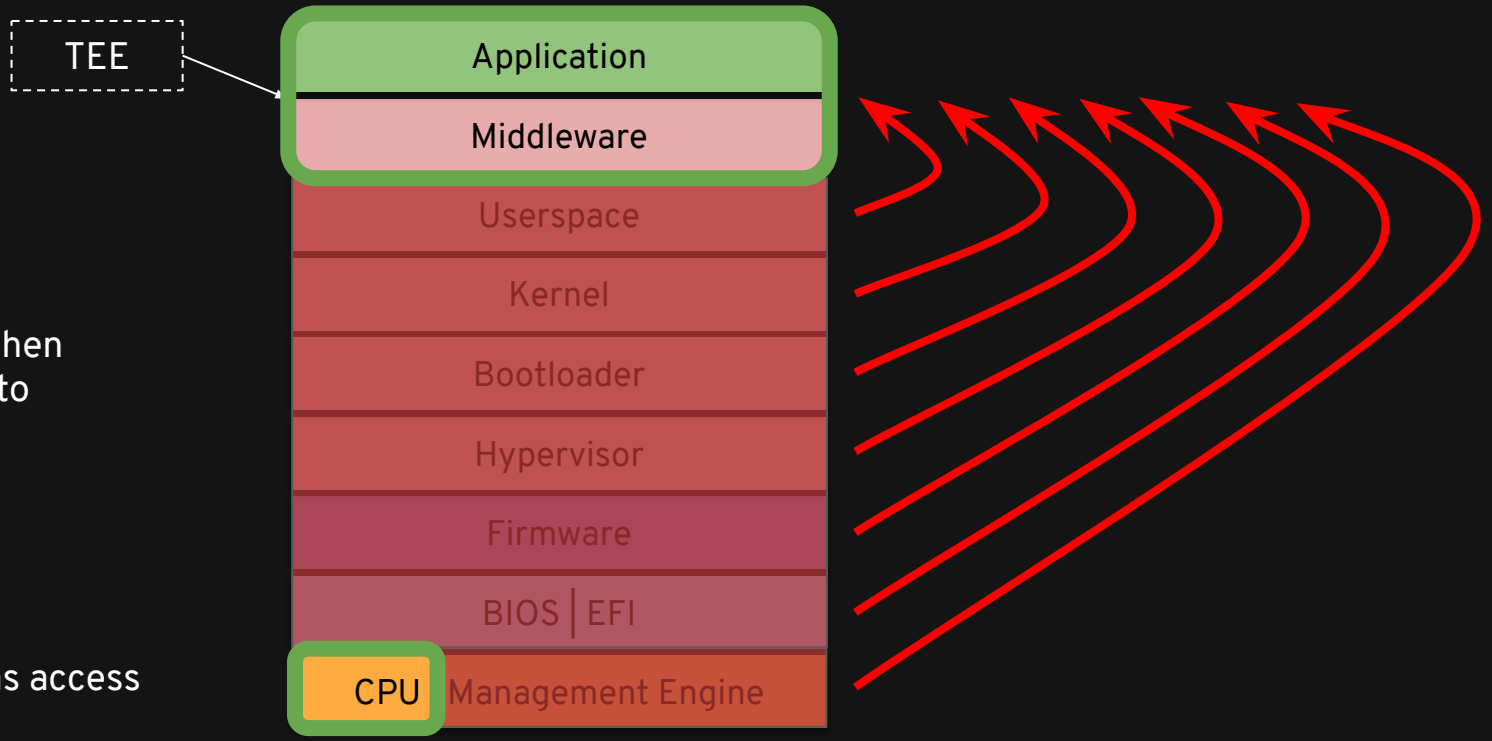
What's a TEE?



Only the CPU has access

Red Hat

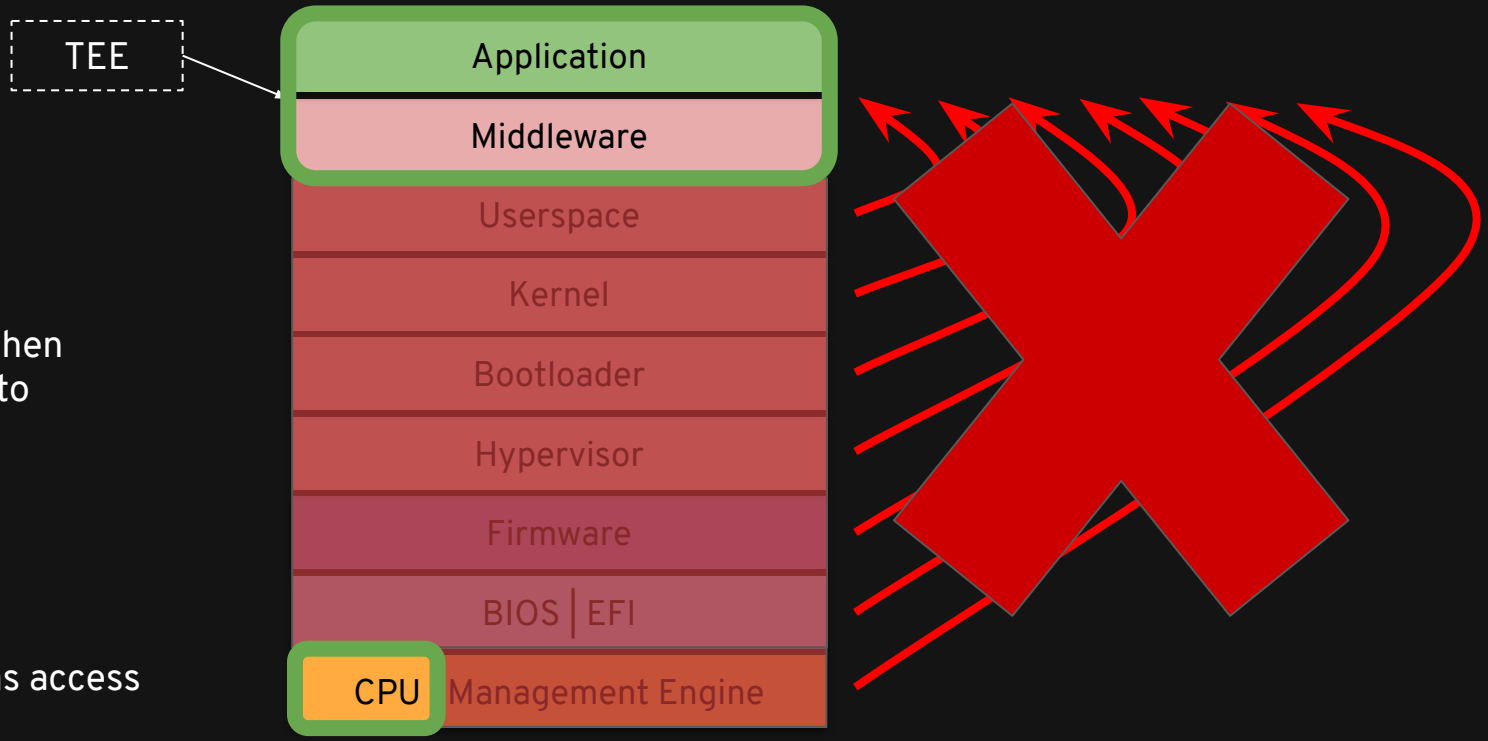
What's a TEE?



What happens when other layers try to access?

Only the CPU has access

What's a TEE?



What happens when other layers try to access?

Blocked by CPU

Only the CPU has access

Trusted Execution Environments

TEE is a protected area within the host, for execution of sensitive workloads

Trusted Execution Environments

TEE is a protected area within the host, for execution of sensitive workloads

TEE provides:

- Memory Confidentiality
- Integrity Protection
- General compute
- HWRNG

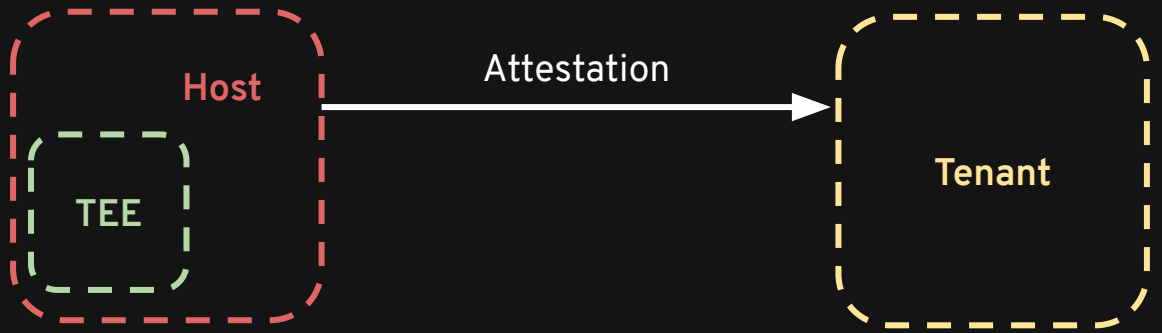
Trusted Execution Environments

Q. “But how do I know that it’s a valid TEE?”

TEE provides:

- Memory Confidentiality
- Integrity Protection
- General compute
- HWRNG

Trusted Execution Summary



Q. “But how do I know that it’s a valid TEE?”
A. Attestation

TEE provides:

- Memory Confidentiality
- Integrity Protection
- General compute
- HWRNG

Trusted Execution Summary

Attestation includes:

- Diffie-Hellman Public Key
- Hardware Root of Trust
- TEE Measurement

TEE provides:

- Memory Confidentiality
- Integrity Protection
- General compute
- HWRNG

Trusted Execution Models

Process-Based

- Intel SGX (not upstream)
- RISC-V Sanctum (no hardware)

VM-Based

- AMD SEV
- IBM PEF (no hardware)
- Intel MKTME (no attestation[1])

Not a TEE: TrustZone, TPM

1. Attestation is discussed here: https://patents.google.com/patent/US20190042463A1/en?oq=20190042463

Trusted Execution: Process-Based

PROS

- Access to system APIs from Keep

CONS

- Unfiltered system API calls from Keep
- Application redesign required
- Untested security boundary
- Fantastic for malware
- Lock-in

Trusted Execution: Virtual Machine-Based

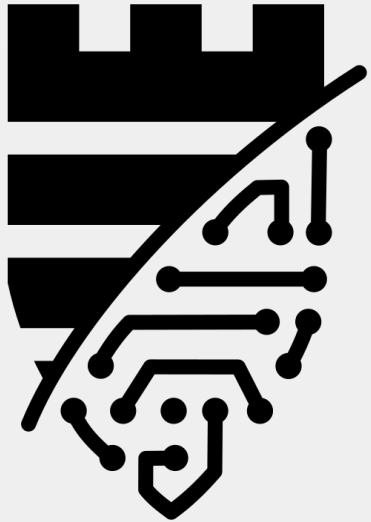
PROS

- Strengthening of existing boundary
- Run application on existing stacks
- Bidirectional isolation
- Limits malware

CONS

- Hardware emulation
- Heavy weight for microservices
- CPU architecture lock-in
- Duplicated kernel pages
- Host-provided BIOS

Introducing Enarx



Red Hat

The Enarx 5-bullet overview

- Uses TEEs (SGX, SEV, etc.) for confidential workloads

The Enarx 5-bullet overview

- Uses TEEs (SGX, SEV, etc.) for confidential workloads
- Easy development and deployment using Wasm

The Enarx 5-bullet overview

- Uses TEEs (SGX, SEV, etc.) for confidential workloads
- Easy development and deployment using Wasm
- Strong security design principles

The Enarx 5-bullet overview

- Uses TEEs (SGX, SEV, etc.) for confidential workloads
- Easy development and deployment using Wasm
- Strong security design principles
- Cloud-native → Openshift, kubernetes

The Enarx 5-bullet overview

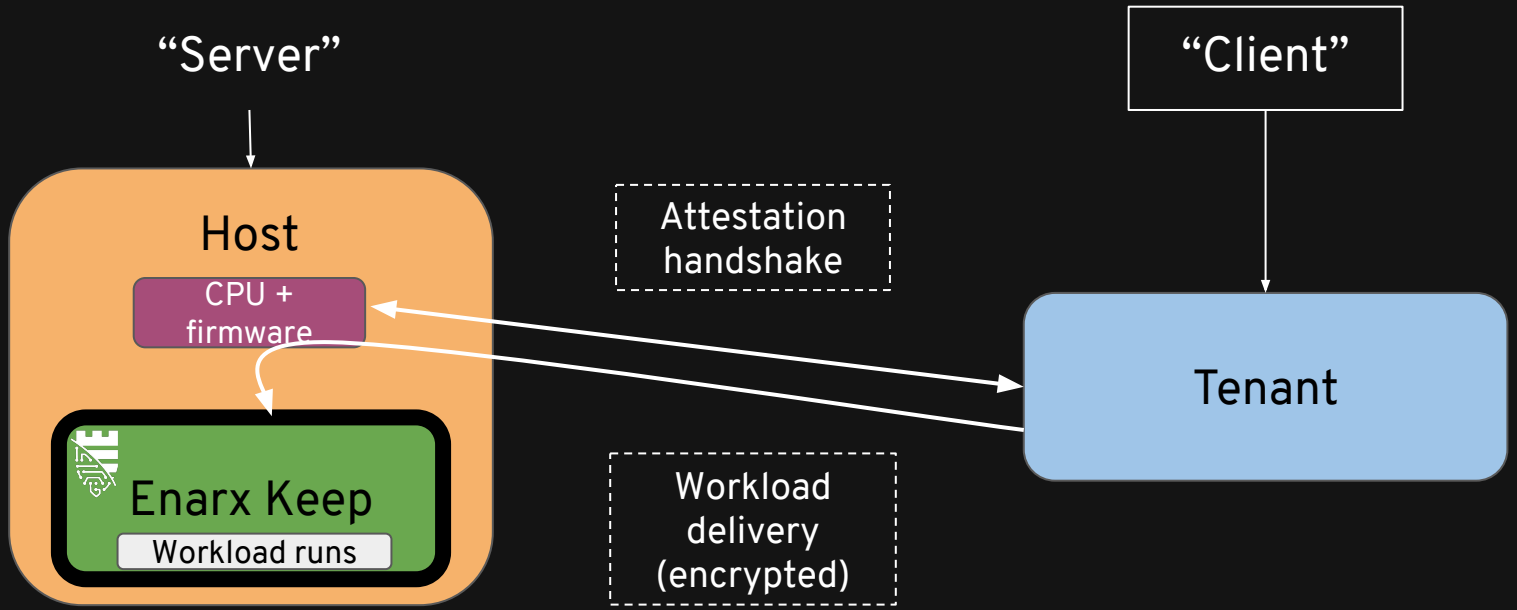
- Uses TEEs (SGX, SEV, etc.) for confidential workloads
- Easy development and deployment using Wasm
- Strong security design principles
- Cloud-native → Openshift, kubernetes
- Open source: *project*, not production-ready (yet)

CONFIDENTIAL COMPUTING
CONSORTIUM

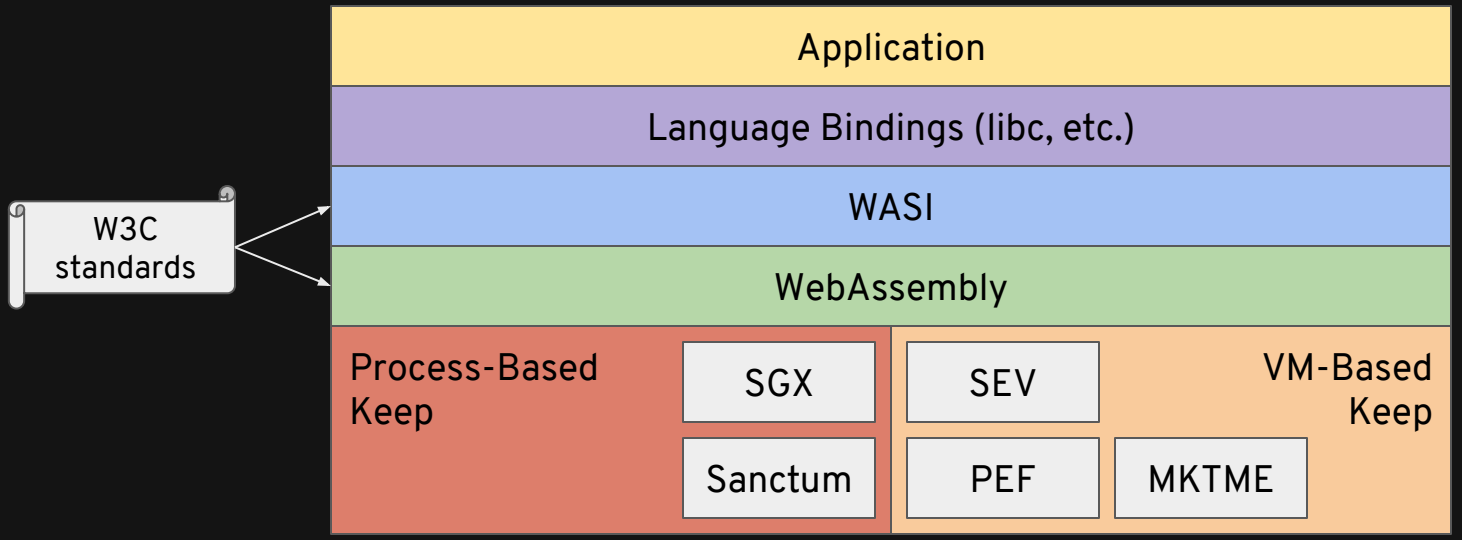
Red Hat

Where do we want to be?

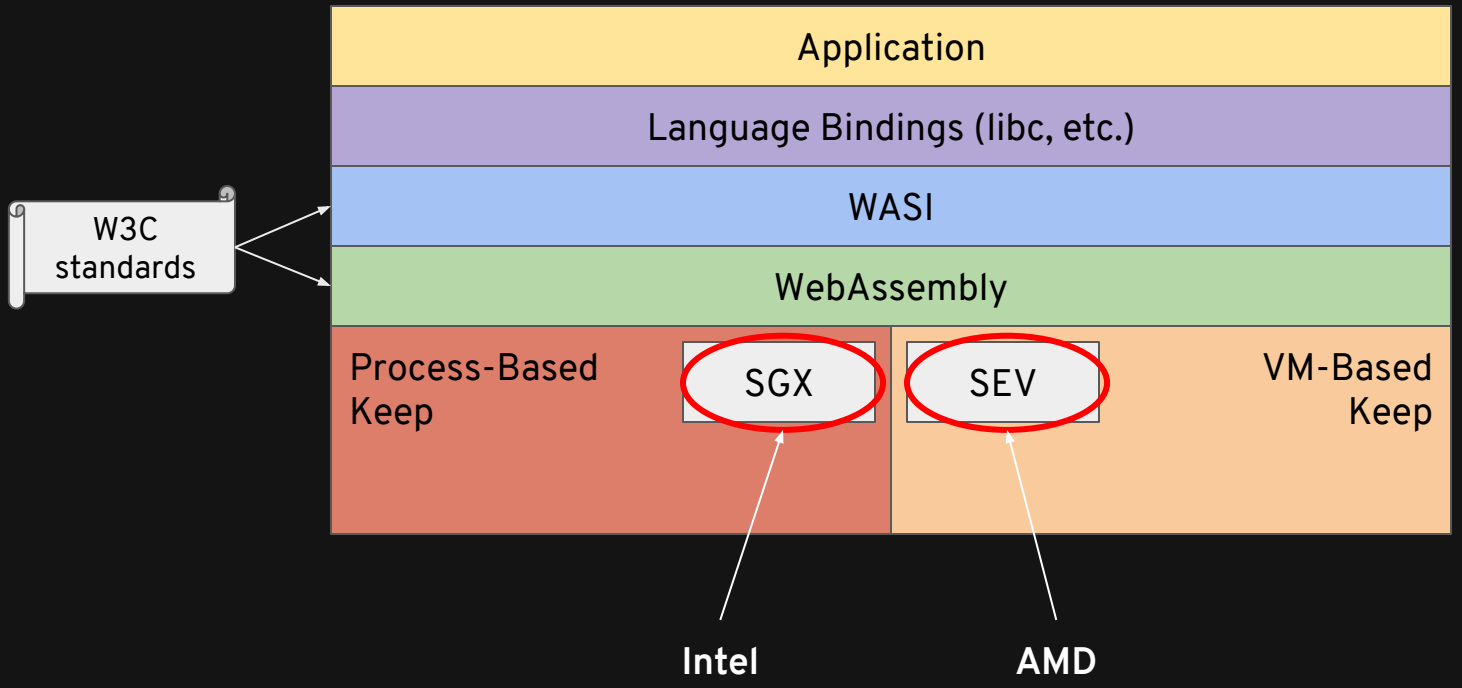
What's the full picture?



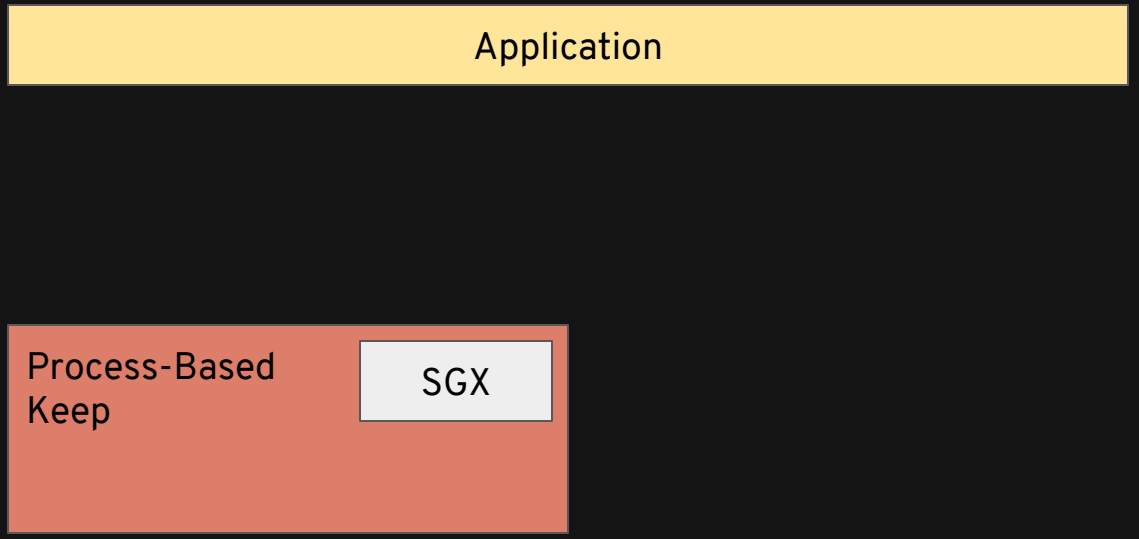
Enarx Architecture



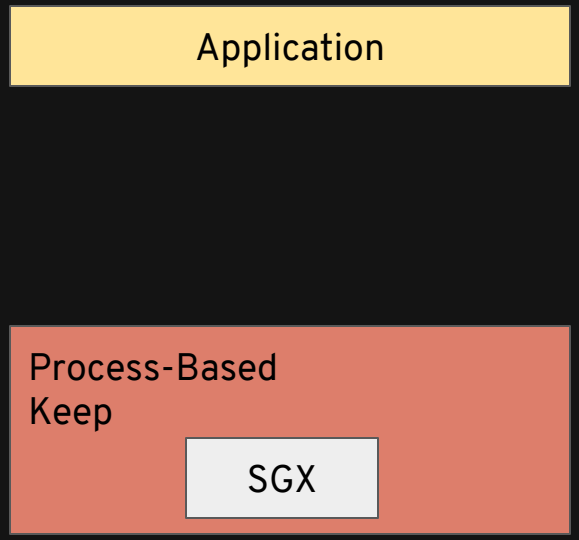
Enarx Architecture



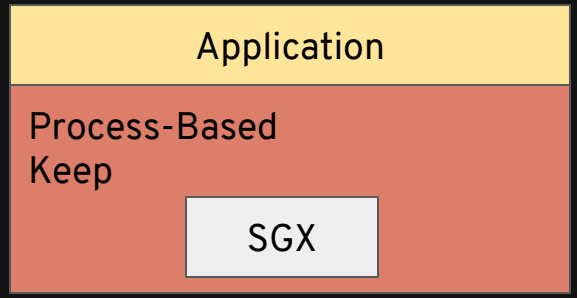
Breaking things down with SGX



Breaking things down with SGX

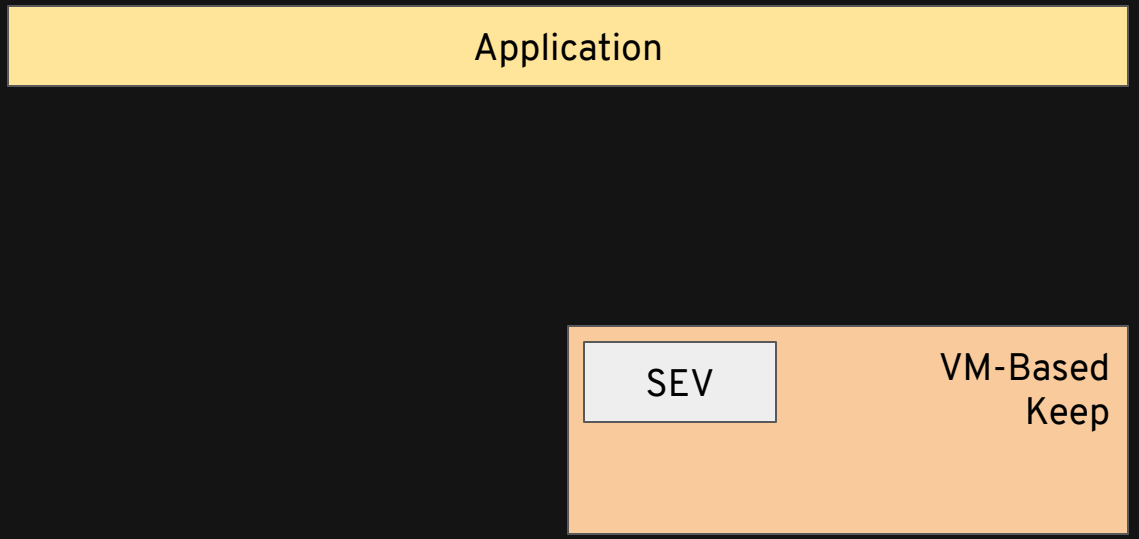


Breaking things down with SGX

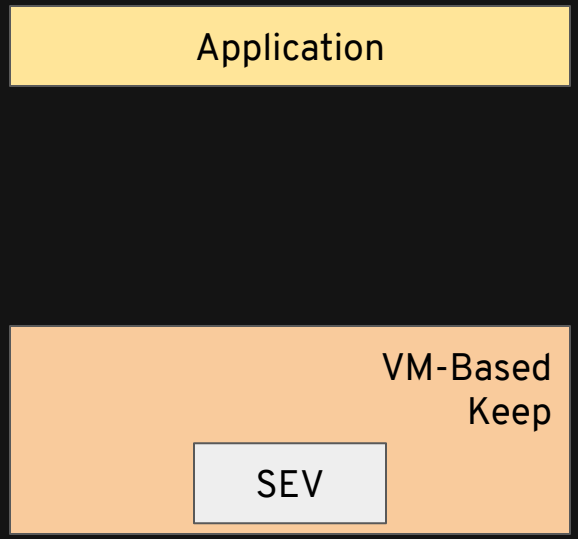


SGX demo

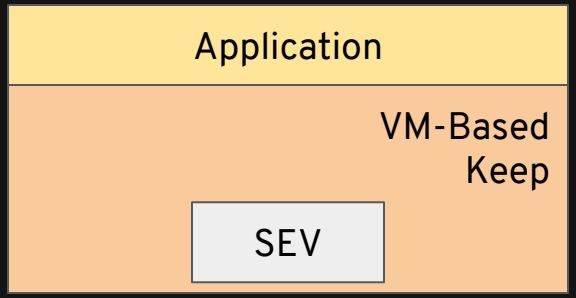
Breaking things down with SEV



Breaking things down with SEV

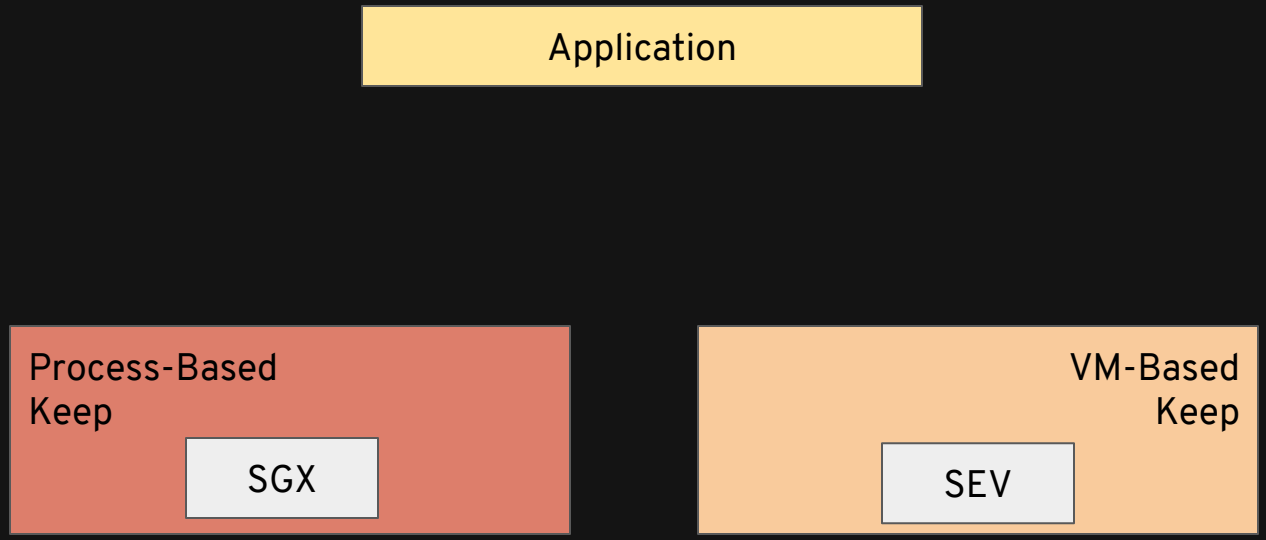


Breaking things down with SEV

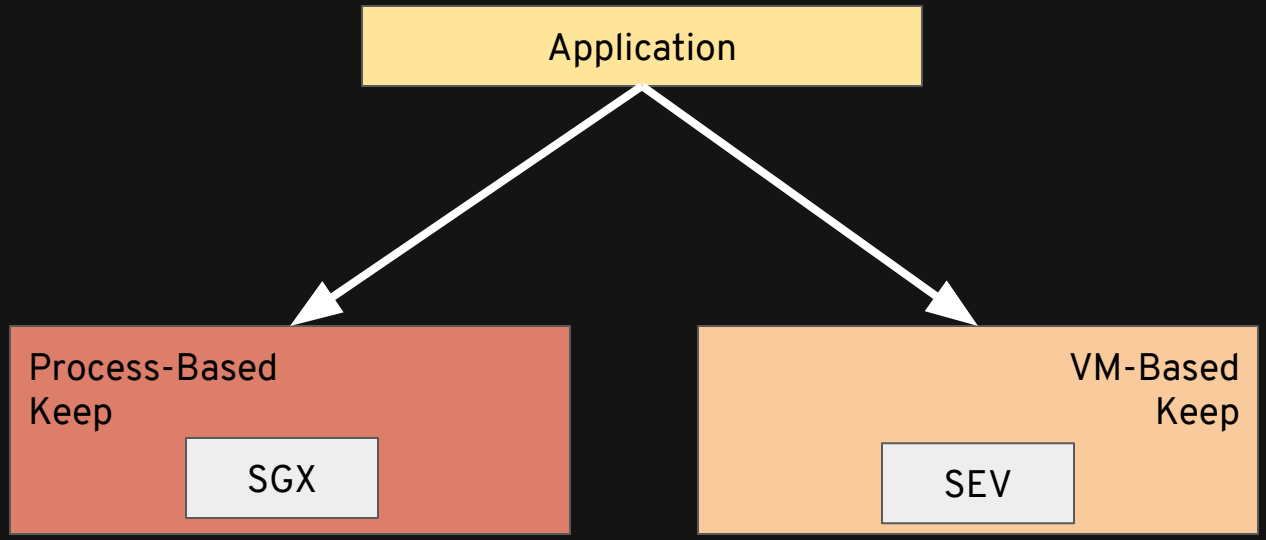


SEV demo

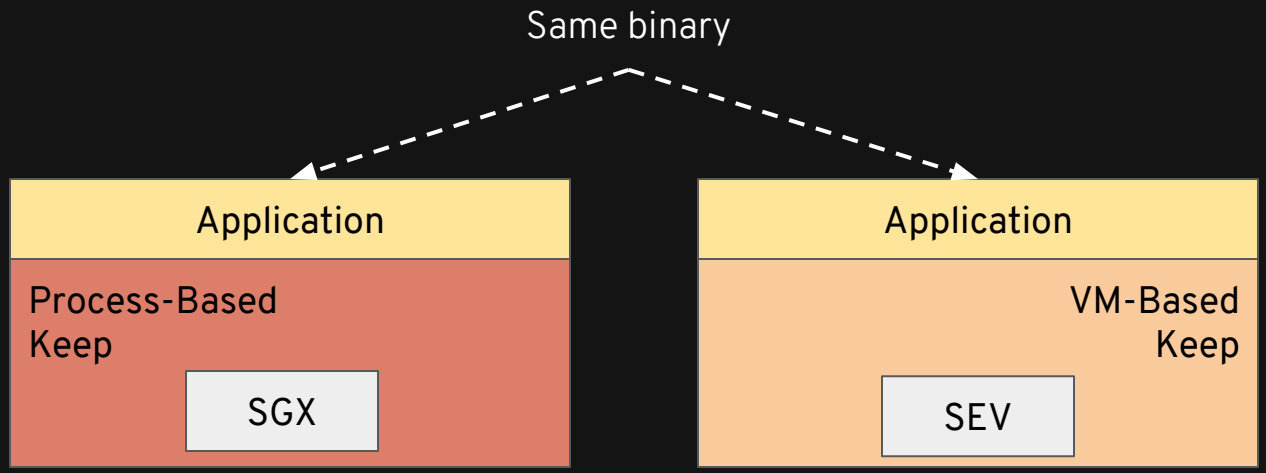
Where we'd like to be



Where we'd like to be

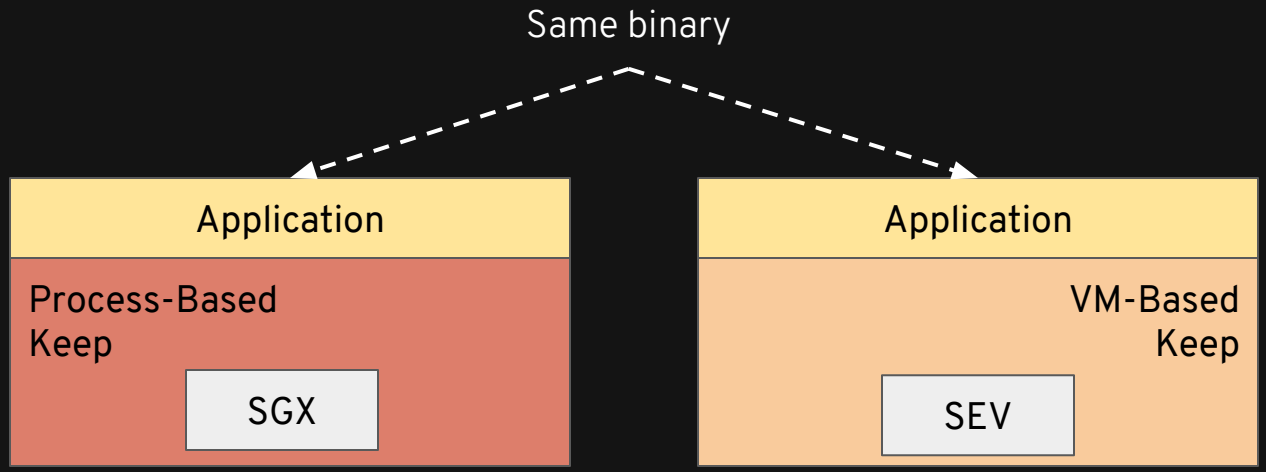


Where we'd like to be



Where ~~I'd like~~ to be

Where we are



Red Hat

```
[nathaniel@localhost enarx]$ cat ~/test.c
#include <stdio.h>

int
main(int argc, char *argv[])
{
    printf("Good morning, that's a nice tnetennba!\n");
    return 0;
}
[nathaniel@localhost enarx]$ musl-gcc -fPIE -static-pie -o ~/test ~/test.c
[nathaniel@localhost enarx]$ file ~/test
/home/nathaniel/test: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), statically linked, BuildID[sha1]=f
ba649daba150448a9ea182aa87d16349f04e1de, with debug_info, not stripped
[nathaniel@localhost enarx]$ target/x86_64-unknown-linux-musl/debug/enarx-keep-sgx --shim target/x86_64-unknown-
linux-musl/debug/enarx-keep-sgx-shim --code ~/test
```

```
[nathaniel@localhost enarx]$ cat ~/test.c
#include <stdio.h>

int
main(int argc, char *argv[])
{
    printf("Good morning, that's a nice tnetennba!\n");
    return 0;
}
[nathaniel@localhost enarx]$ musl-gcc -fPIE -static-pie -o ~/test ~/test.c
[nathaniel@localhost enarx]$ file ~/test
/home/nathaniel/test: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), statically linked, BuildID[sha1]=f
ba649daba150448a9ea182aa87d16349f04e1de, with debug_info, not stripped
[nathaniel@localhost enarx]$ target/x86_64-unknown-linux-musl/debug/enarx-keep-sev --shim target/x86_64-unknown-
linux-musl/debug/enarx-keep-sev-shim --code ~/test
```

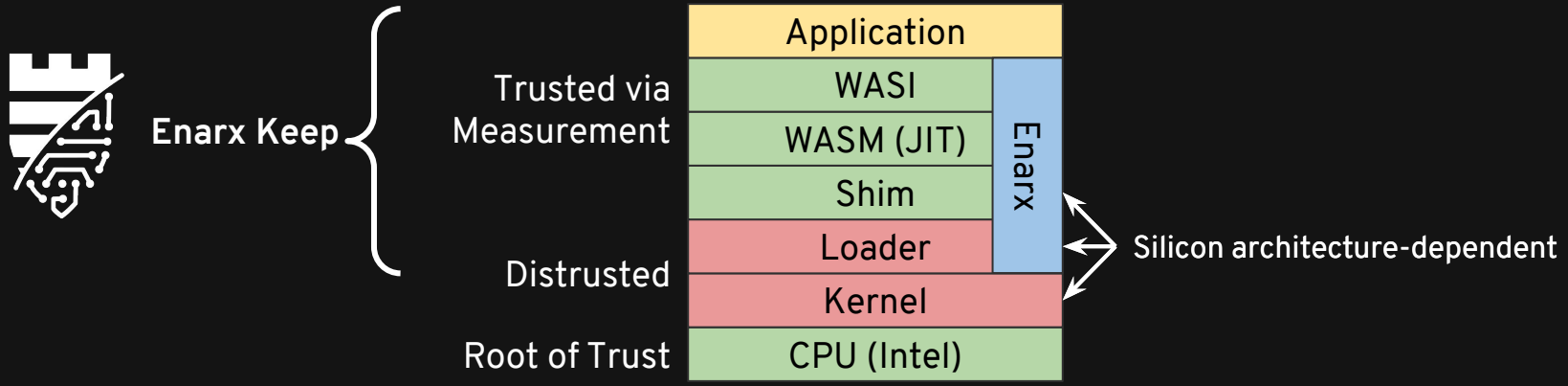
```
[nathaniel@localhost enarx]$ cat ~/test.c
#include <stdio.h>

int
main(int argc, char *argv[])
{
    printf("Good morning, that's a nice tnetennba!\n");
    return 0;
}
[nathaniel@localhost enarx]$ musl-gcc -fPIE -static-pie -o ~/test ~/test.c
[nathaniel@localhost enarx]$ file ~/test
/home/nathaniel/test: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), statically linked, BuildID[sha1]=f
ba649daba150448a9ea182aa87d16349f04e1de, with debug_info, not stripped
[nathaniel@localhost enarx]$ target/x86_64-unknown-linux-musl/debug/enarx-keep-sgx --shim target/x86_64-unknown-
linux-musl/debug/enarx-keep-sgx-shim --code ~/test
```

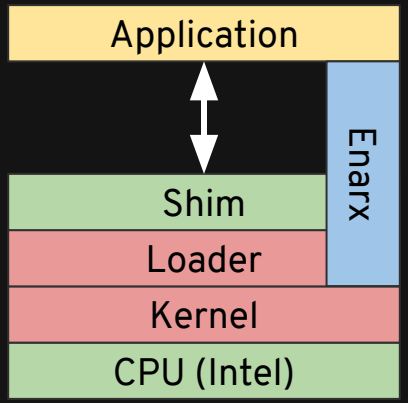
```
[nathaniel@localhost enarx]$ cat ~/test.c
#include <stdio.h>

int
main(int argc, char *argv[])
{
    printf("Good morning, that's a nice tnetennba!\n");
    return 0;
}
[nathaniel@localhost enarx]$ musl-gcc -fPIE -static-pie -o ~/test ~/test.c
[nathaniel@localhost enarx]$ file ~/test
/home/nathaniel/test: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), statically linked, BuildID[sha1]=f
ba649daba150448a9ea182aa87d16349f04e1de, with debug_info, not stripped
[nathaniel@localhost enarx]$ target/x86_64-unknown-linux-musl/debug/enarx-keep-sev --shim target/x86_64-unknown-
linux-musl/debug/enarx-keep-sev-shim --code ~/test
```

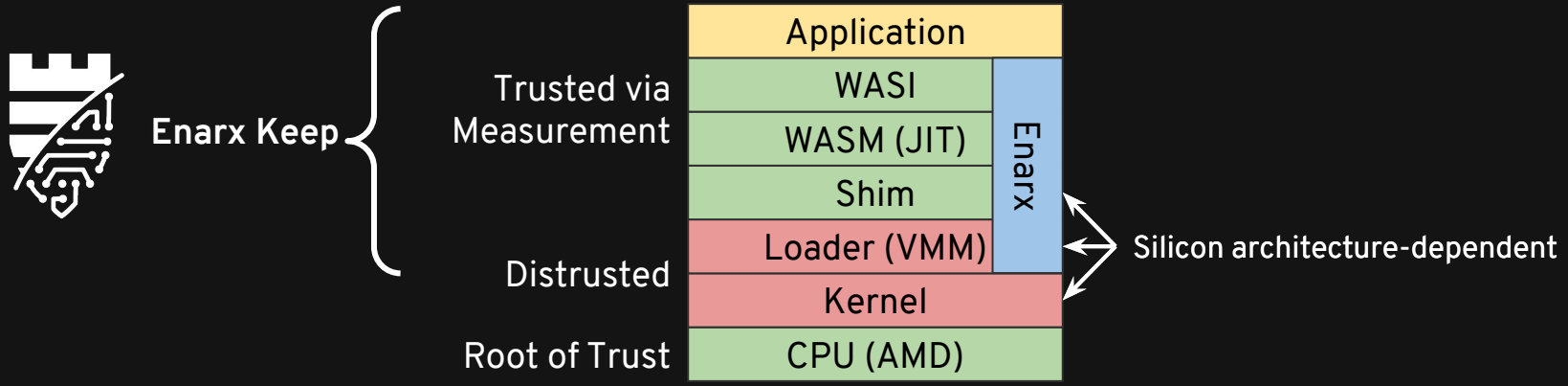
Layers - process-based Keep



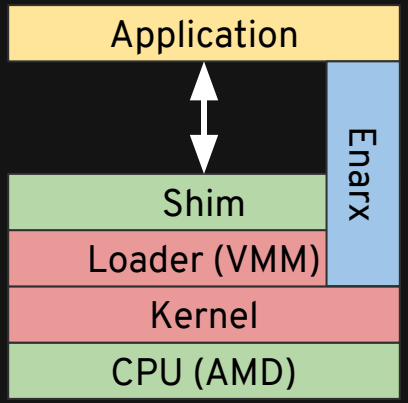
Layers (now) - process-based Keep



Layers - VM-based Keep

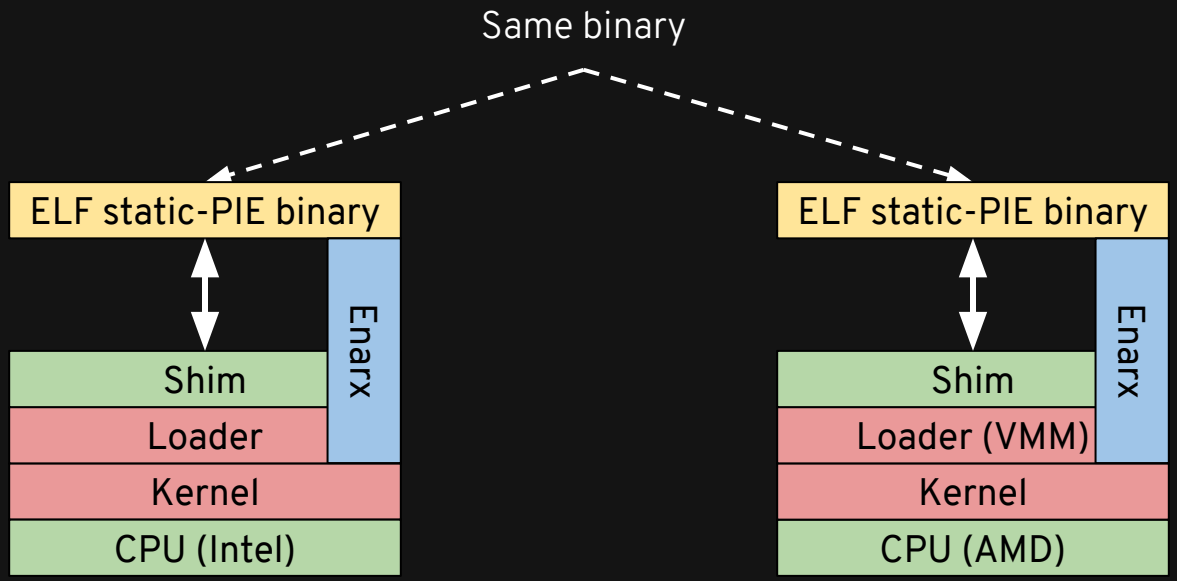


Layers (now) - process-based Keep

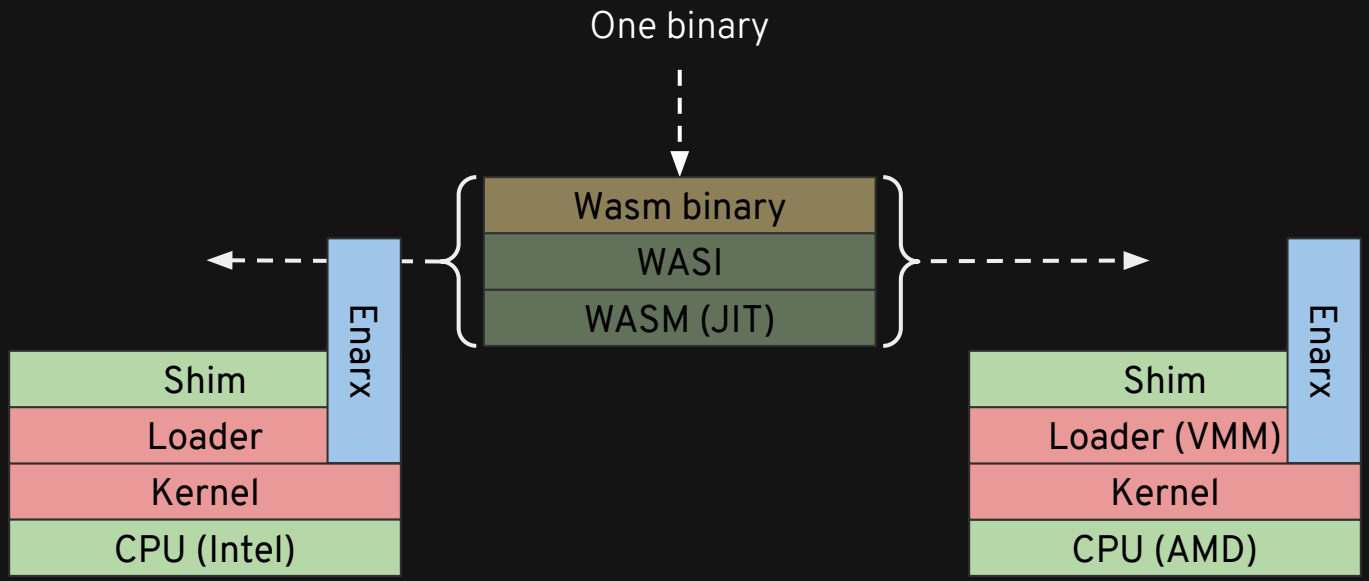


~~Where we would like to be~~

Where we are



Where we'd like to be *next*



We are an open project

• Code	✓ GitHub	
• Wiki	✓ GitHub	
• Design	✓ GitHub	
• Issues & PRs	✓ GitHub	
• Chat	✓ Rocket.Chat	(Thank you!)
• CI/CD resources	✓ Packet.io	(Thank you!)
• Stand-ups	✓ Open to all	
• Diversity	✓ Contributor Covenant CofC	

We Need Your Help!

Website: https://enarx.io

Code: https://github.com/enarx

License: Apache 2.0

Language: Rust

Daily stand-ups open to all! Check the website wiki for details.

We Need Your Help!

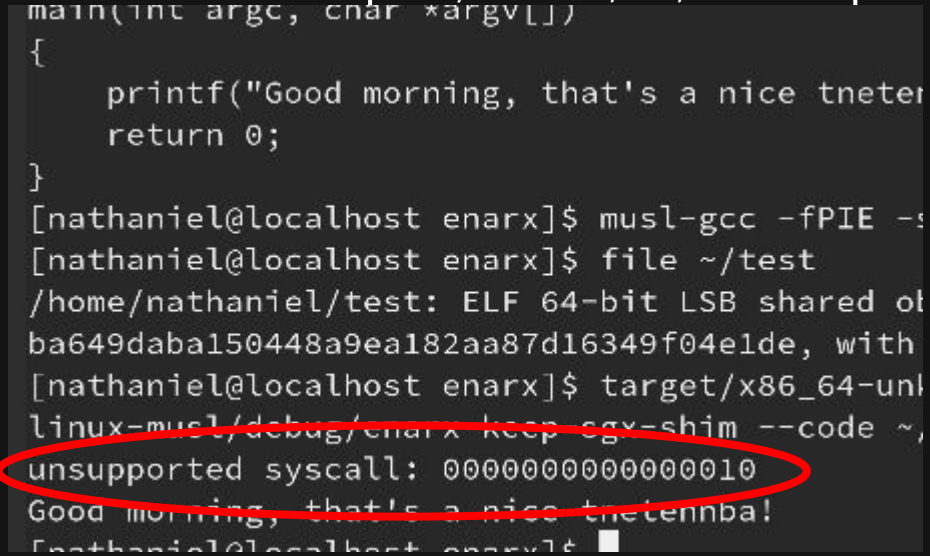
Website: https://enarx.io

Code: https://github.com/enarx

License: Apache 2.0

Language: Rust

Daily stand-ups open to all!



Red Hat

Questions?

https://enarx.io

Red Hat