

# Using WebAssembly to run, extend, and secure your .NET application

Niels Tanis

NDC { Security }

01 VERACODE 01



0101  
0101

# Who am I?

- Niels Tanis
- Sr. Principal Security Researcher @ Veracode
  - Background .NET Development,  
Pentesting/ethical hacking,  
and software security consultancy
  - Research on static analysis for .NET apps



0101  
0101

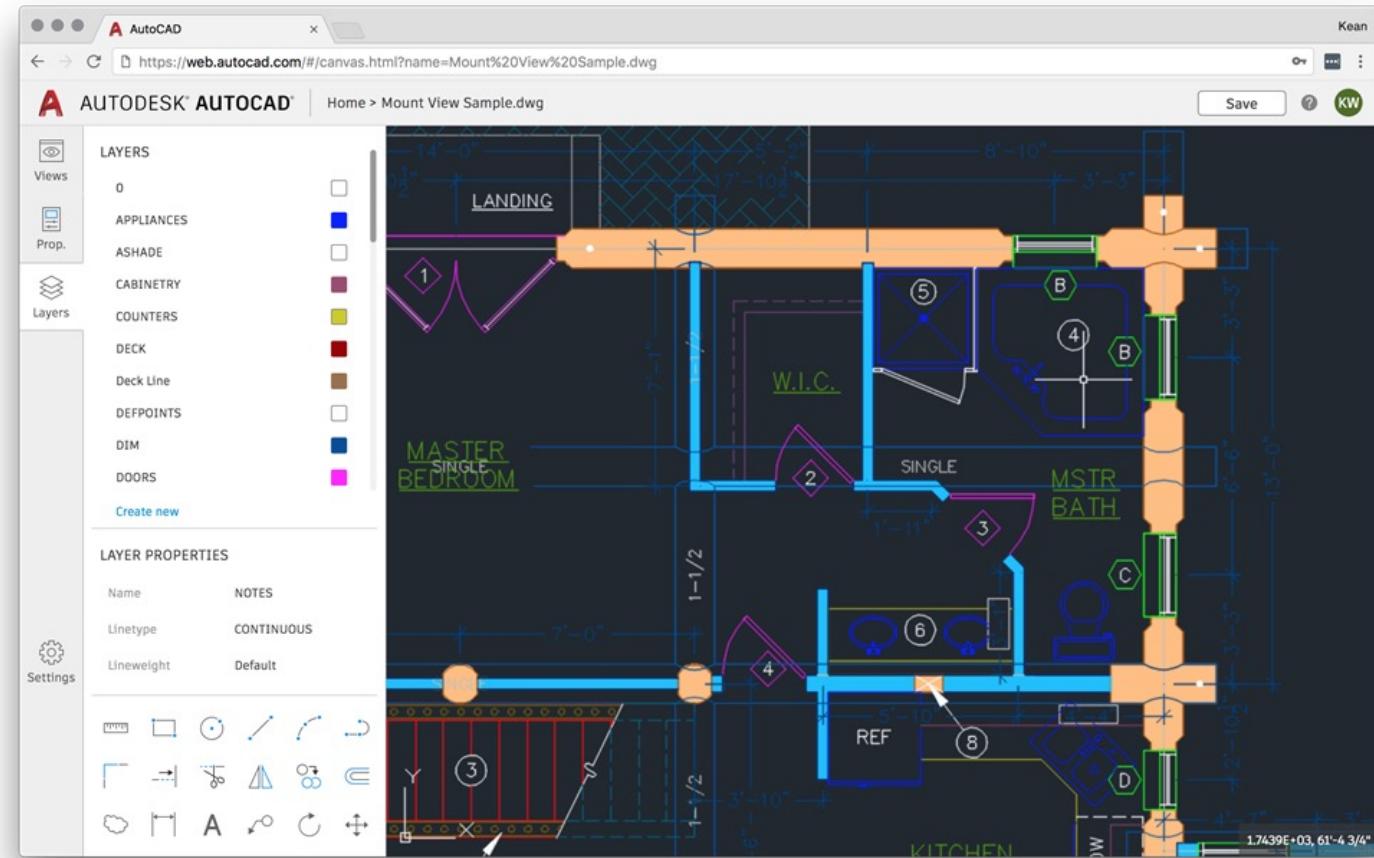
# WebAssembly

The screenshot shows a web browser window displaying the [WebAssembly High-Level Goals](https://webassembly.org/docs/high-level-goals/) page. The page has a dark header with a purple 'WA' logo and the word 'WEBASSEMBLY'. Below the header, a green banner states 'WebAssembly 1.0 has shipped in 4 major browser engines.' followed by icons for Firefox, Chrome, Opera, and Edge, with a 'Learn more' link. The main content area is titled 'WebAssembly High-Level Goals' and lists three main goals:

1. Define a portable, size- and load-time-efficient binary format to serve as a compilation target which can be compiled to execute at native speed by taking advantage of common hardware capabilities available on a wide range of platforms, including mobile and IoT.
2. Specify and implement incrementally:
  - a Minimum Viable Product (MVP) for the standard with roughly the same functionality as `asm.js`, primarily aimed at C/C++;
  - additional features 🎉, initially focused on key features like `threads`, `zero cost exceptions`, and `SIMD`, followed by additional features prioritized by feedback and experience, including support for languages other than C/C++.
3. Design to execute within and integrate well with the existing Web platform:
  - maintain the versionless, feature-tested and backwards-compatible evolution story of the Web;



# WebAssembly - AutoCAD



0101  
0101

# WebAssembly - SDK's

A screenshot of a Medium article page. The title is "Introducing the Disney+ Application Development Kit (ADK)" by Mike Hanley. The article was published on Sep 8, 2021, and has a 10 min read time. It features a profile picture of Mike Hanley, social sharing icons (Twitter, Facebook, LinkedIn), and a "Get started" button. The background shows a blurred version of the article content.

A screenshot of a Medium article page. The title is "How Prime Video updates its app for more than 8,000 device types" by Alexandru Ene. The article is categorized under "CLOUD AND SYSTEMS". It includes a large graphic of overlapping triangles, a "Subscribe" button, and a "Share" button. The background shows a blurred version of the article content.



# Agenda

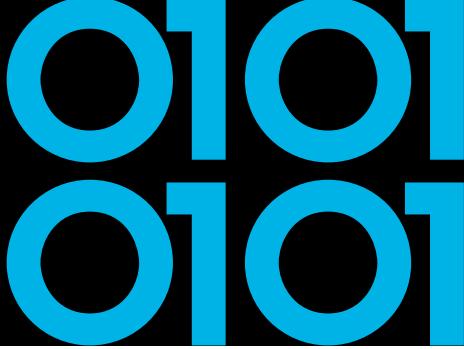
- Introduction
- WebAssembly 101
- Running .NET on WebAssembly
- Extending .NET with WebAssembly
- Securing .NET with WebAssembly
- Conclusion
- Q&A



# WebAssembly Design

- **Be fast, efficient, and portable**
  - Executed in near-native speed across different platforms
- **Be readable and debuggable**
  - In low-level bytecode but also human readable
- **Keep secure**
  - Run on sandboxed execution environment
- **Don't break the web**
  - Ensure backwards compatibility





# WebAssembly

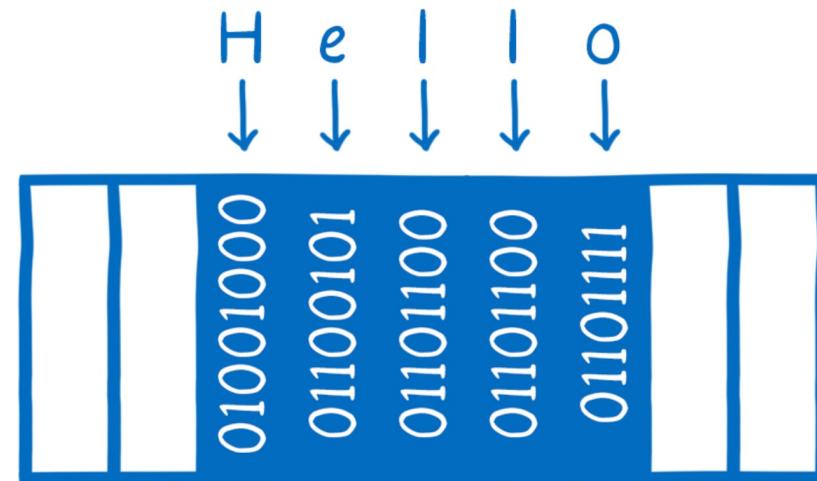
- Binary instruction format for stack-based virtual machine similar to .NET running MSIL
- Designed as a portable compilation target
- The security model of WebAssembly:
  - Protect users from buggy or malicious modules
  - Provide developers with useful primitives and mitigations for developing safe applications



0101  
0101

# WebAssembly Memory

- Isolated per WASM module
- A contiguous, mutable array of uninterpreted bytes





# WebAssembly Control-Flow Integrity

```
int number = Convert.ToInt32(Console.ReadLine());
Console.WriteLine($"Number {number}");
if (number>5)
{
    Console.WriteLine("Number is larger than 5");
}
else
{
    Console.WriteLine("Number is smaller than 5");
}
Console.WriteLine("Done!");
```



# WebAssembly Control-Flow Integrity

```
int number = Convert.ToInt32(Console.ReadLine());  
Console.WriteLine($"Number {number}");  
if (number>5)
```

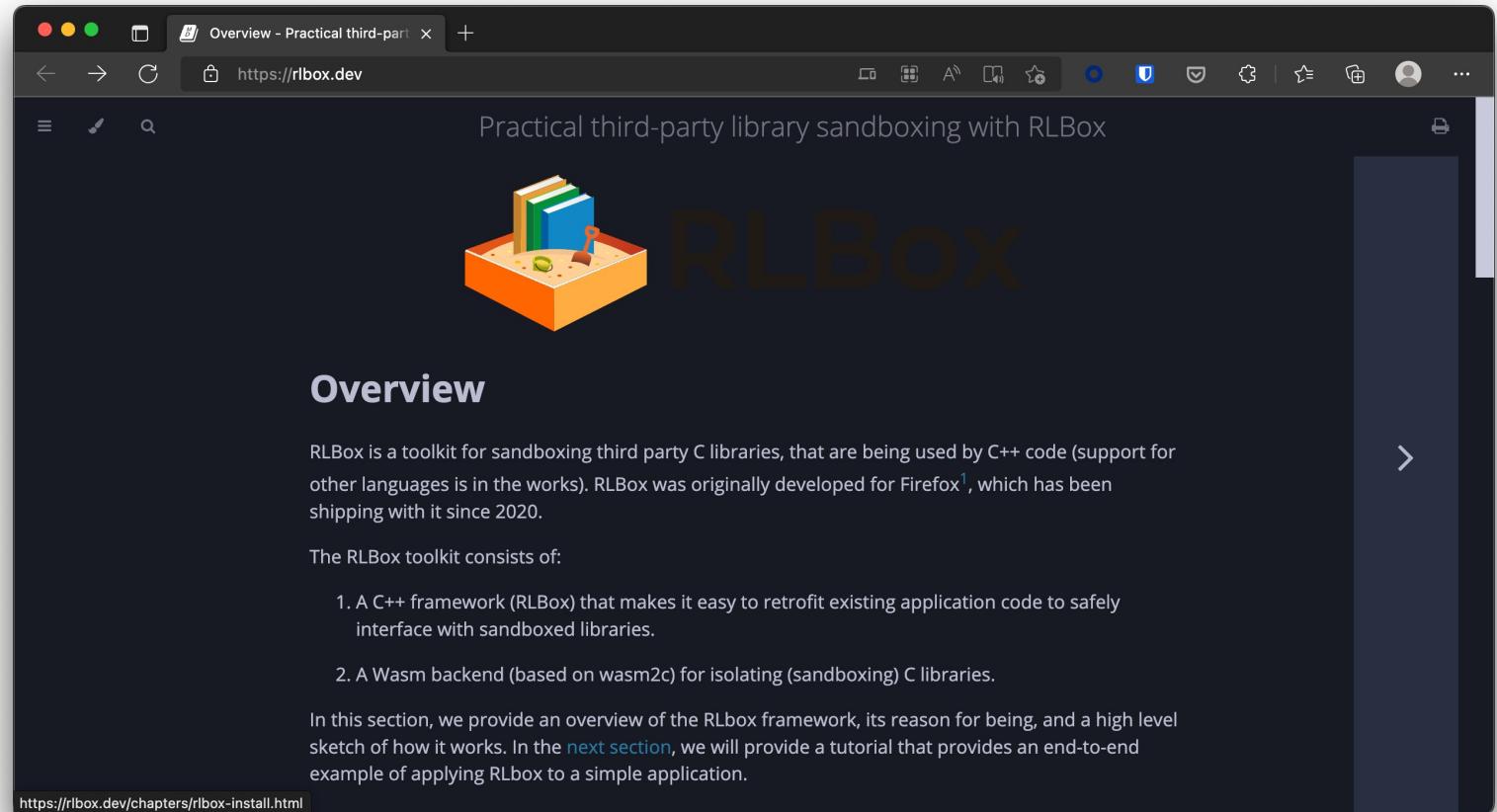
```
Console.WriteLine("Number is larger than 5");
```

```
Console.WriteLine("Number is smaller than 5");
```

```
Console.WriteLine("Done!");
```

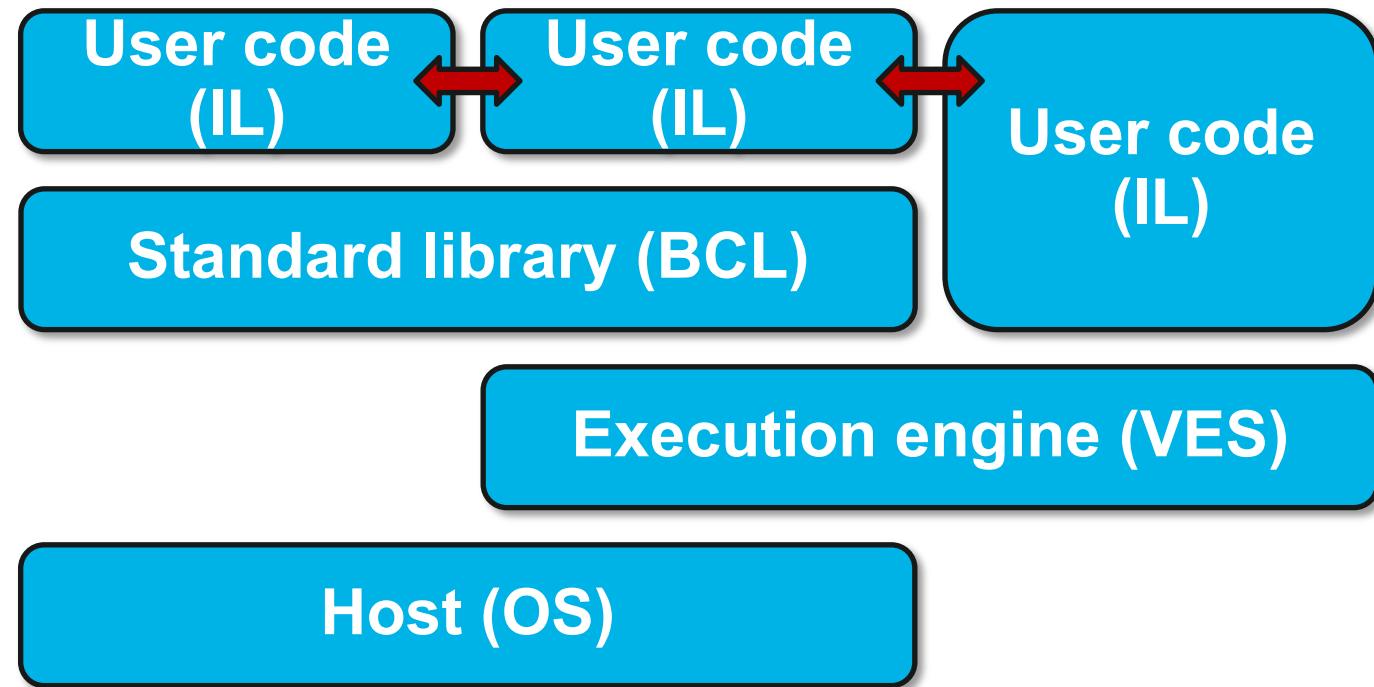
0101  
0101

# FireFox RLBox



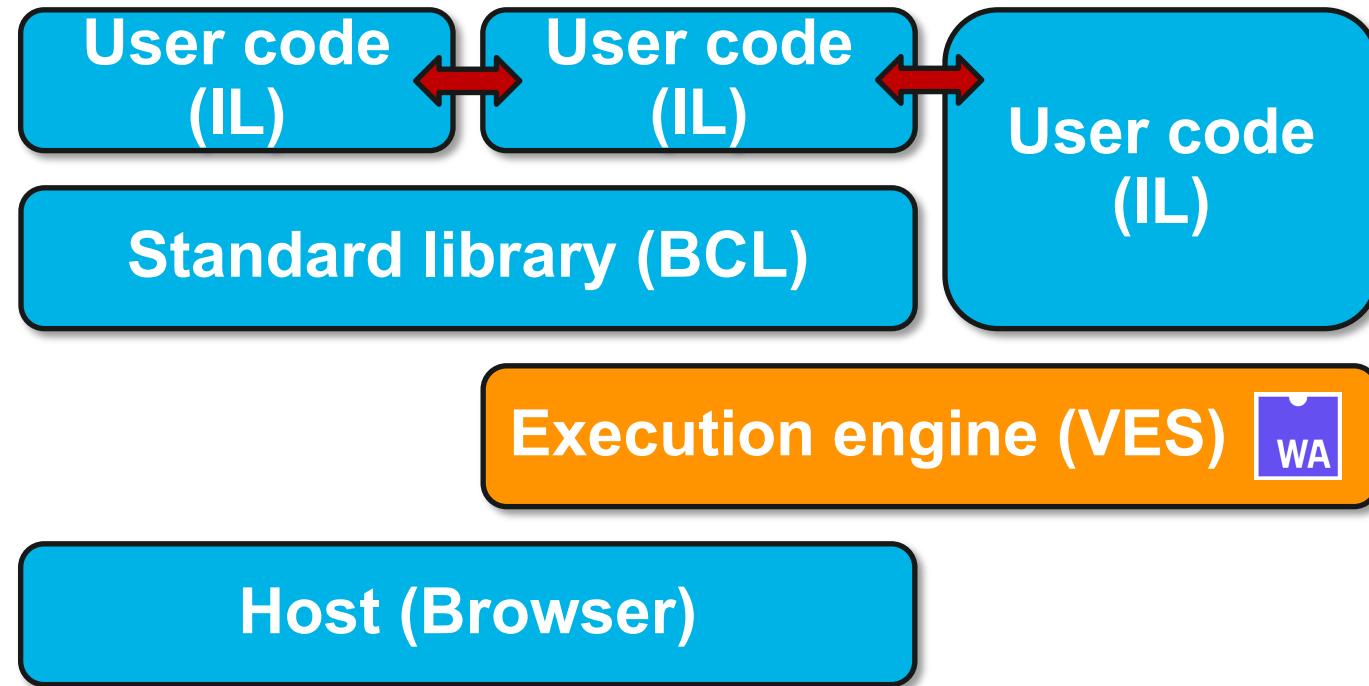


# Running .NET on WebAssembly



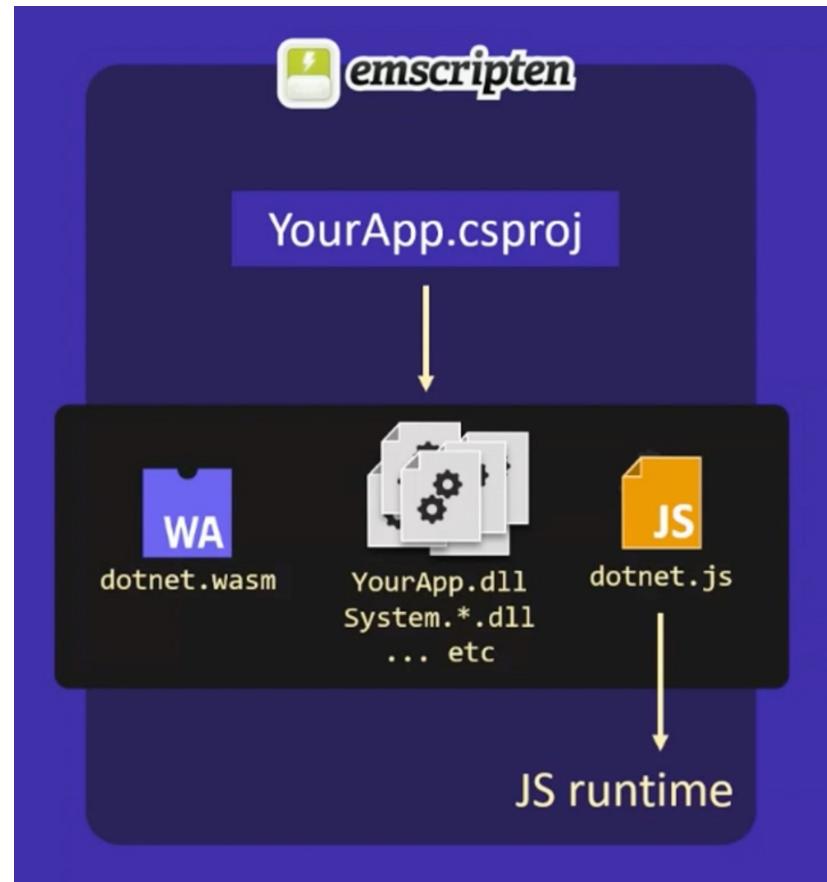


# Running .NET on WebAssembly



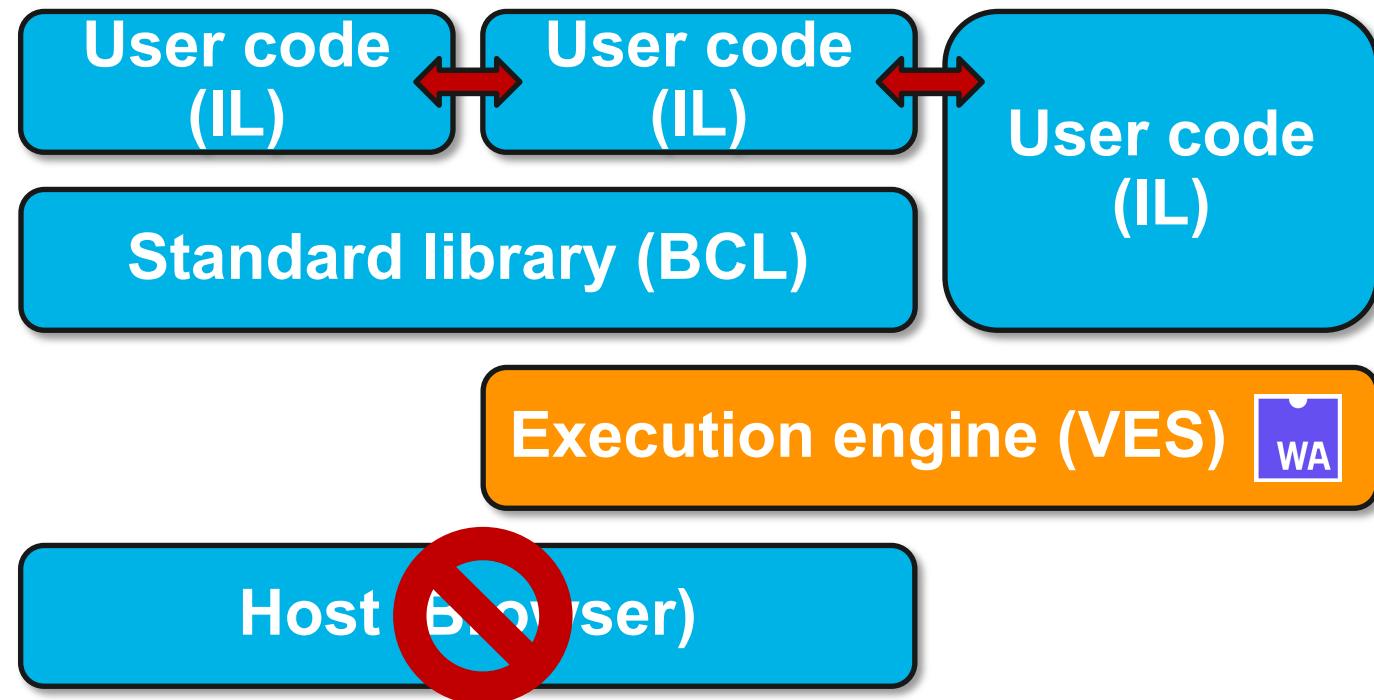


# Blazor WebAssembly





# Running .NET on WebAssembly



# WebAssembly System Interface WASI



- Introduced in March 2019 by Bytecode Alliance
- WasmTime implementation as reference
- POSIX inspired, engine-independent, non-Web system-oriented API for WebAssembly

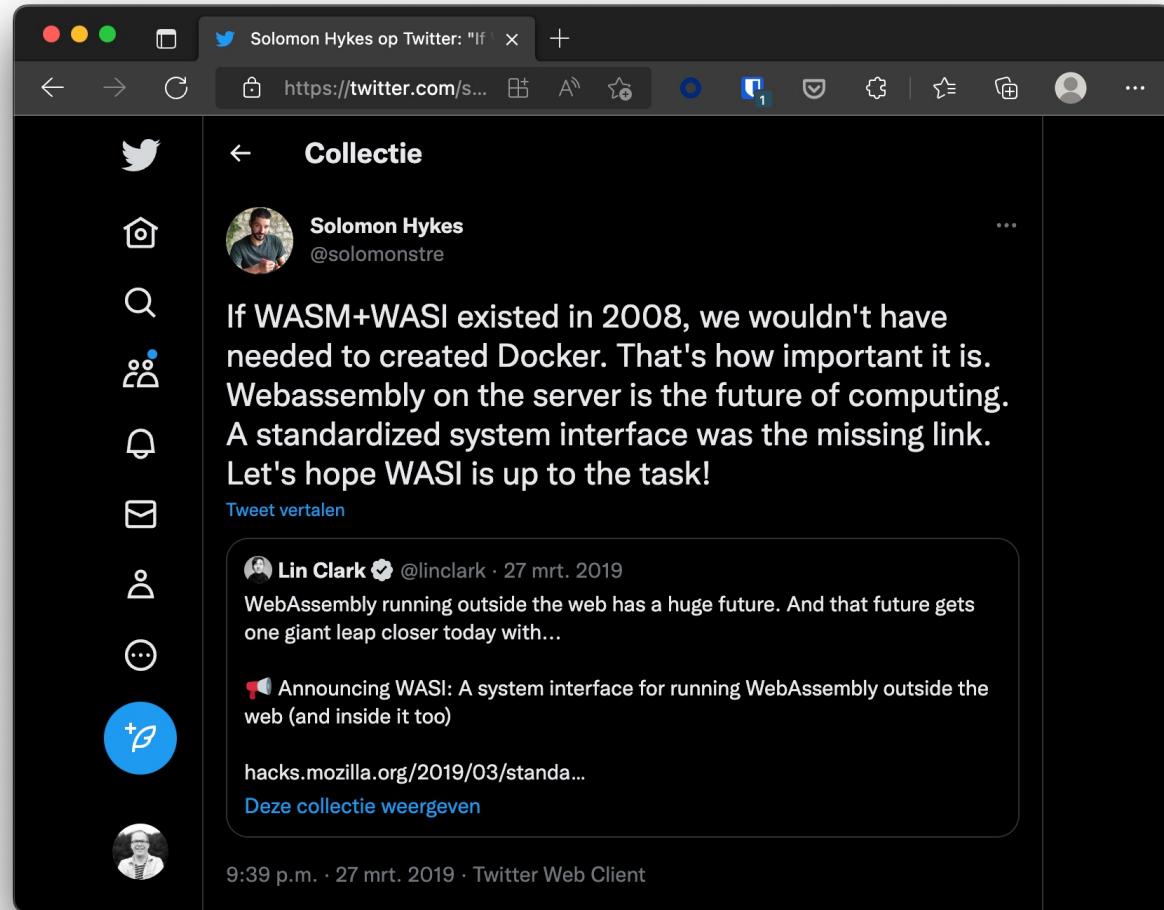
# WebAssembly System Interface WASI



- Strong sandbox with Capability Based Security
- Right now, supports e.g. FileSystem actions
- Future support for sockets and other system resources.
- Anyone recall .NET Standard? ☺

0101  
0101

# Docker vs WASM & WASI



0101  
0101

# Docker vs WASM & WASI

A screenshot of a Twitter web application window. The tweet is from Solomon Hykes (@solomonstre) dated March 27, 2019. The tweet content is: "So will wasm replace Docker?" No, but imagine a future where Docker runs linux containers, windows containers and wasm containers side by side. Over time wasm might become the most popular container type. Docker will love them all equally, and run it all :)"

The tweet includes a reply from Solomon Hykes (@solomonstre) dated March 28, 2019, stating: "If WASM+WASI existed in 2008, we wouldn't have needed to created Docker. That's how important it is. Webassembly on the server is the future of computing. A standardized system interface was the missing link. Let's hope WASI is up to the task! [twitter.com/linclark/status...](https://twitter.com/linclark/status/110881110000000000)".

At the bottom of the tweet card, there is a link "Deze collectie weergeven".

Below the tweet card, the timestamp is 4:50 a.m. · 28 mrt. 2019 · Twitter Web App. At the very bottom of the card, there are engagement metrics: 56 Retweets, 5 Geciteerde Tweets, and 165 Vind-ik-leuks.

# Docker & WASM

0101  
0101

The screenshot shows a web browser window with the title "Introducing the Docker+Wasm Technical Preview". The page features a purple header bar with the text "Wasm is a fast, light alternative to Linux containers — try it out today in the Docker+Wasm Technical Preview". Below this is the Docker+Wasm logo. The main content area has a dark blue background with the heading "Introducing the Docker+Wasm Technical Preview" in large white font. At the bottom left is a circular profile picture of Michael Irwin, followed by his name "MICHAEL IRWIN" and the date "Oct 24 2022". A text block at the bottom states: "The Technical Preview of Docker+Wasm is now available! Wasm has been producing a lot of buzz recently, and this feature will make it easier for you to quickly build applications targeting Wasm runtimes."

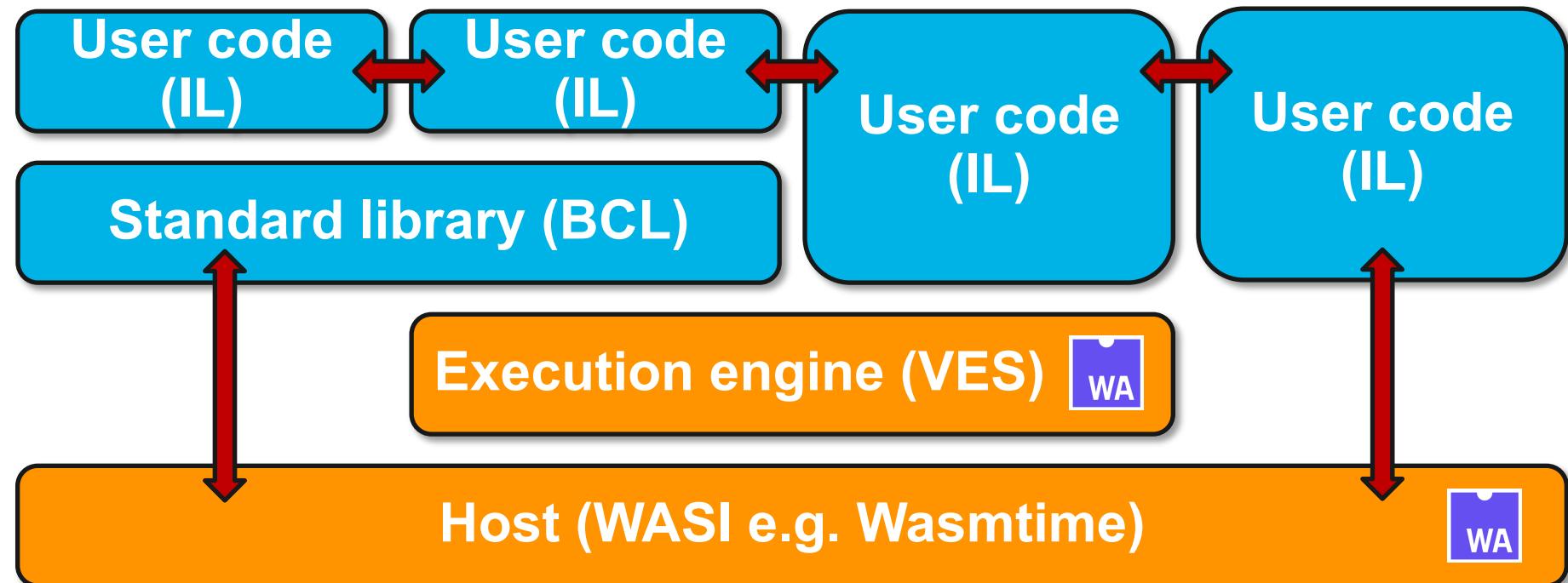
The screenshot shows a diagram of the Docker Engine architecture. At the top is the "Docker Engine" box, which points down to the "containerd" box. The "containerd" box then branches into three separate containers, each labeled "containerd-shim". Each "containerd-shim" box contains a "runc" box, which in turn contains a "Container process" box. The middle "containerd-shim" box also points down to a "wasmedge" box, which contains a "Wasm Module" box.

**Let's look at an example!**

After installing the preview, we can run the following command to start an example Wasm application:

```
docker run -dp 8080:8080 --name=wasm-example --runtime=io.containerd.wasmedge.v1 --platform=wasi/wasm32 michaelirwin244/wasm-example
```

# WebAssembly System Interface WASI



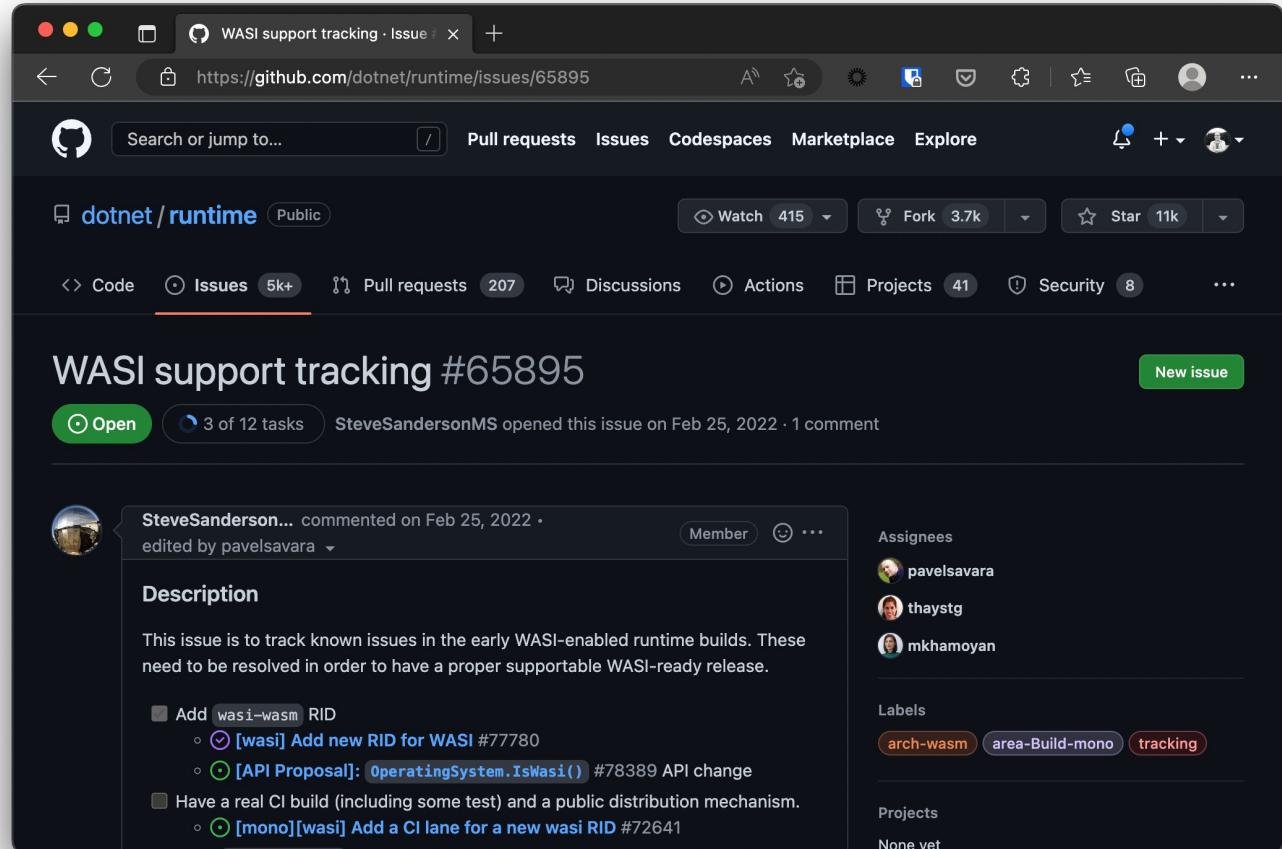


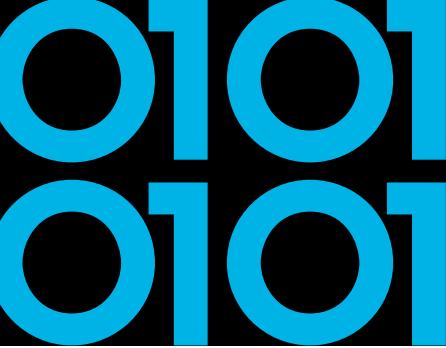
# Experimental WASI SDK for .NET



0101  
0101

# Experimental WASI SDK for .NET



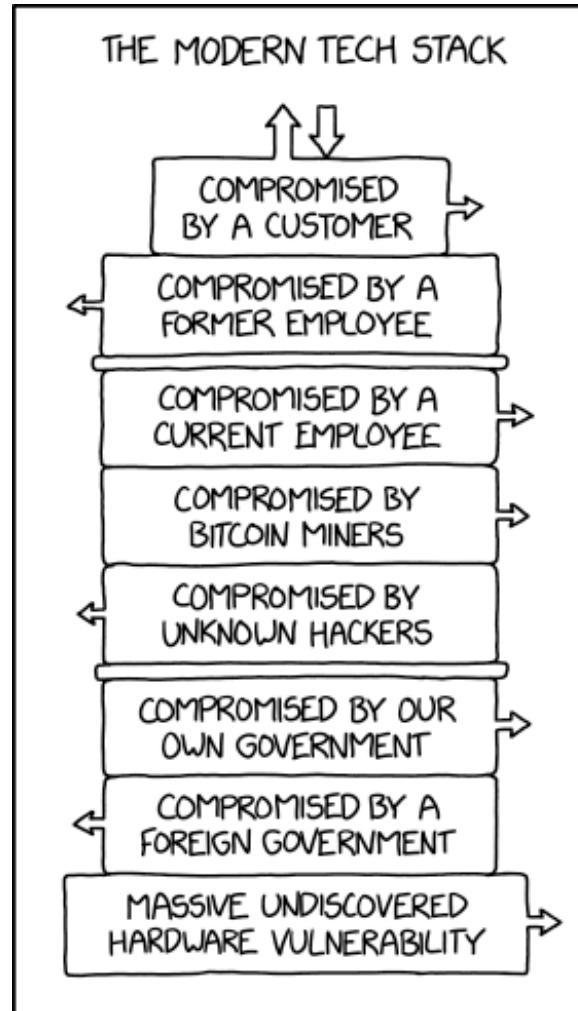


# Extending .NET with WASM

- WasmTime.NET NuGet package
- Can run WASM inside of any .NET application
- Extend with Rust based WASM module
- Demo time!

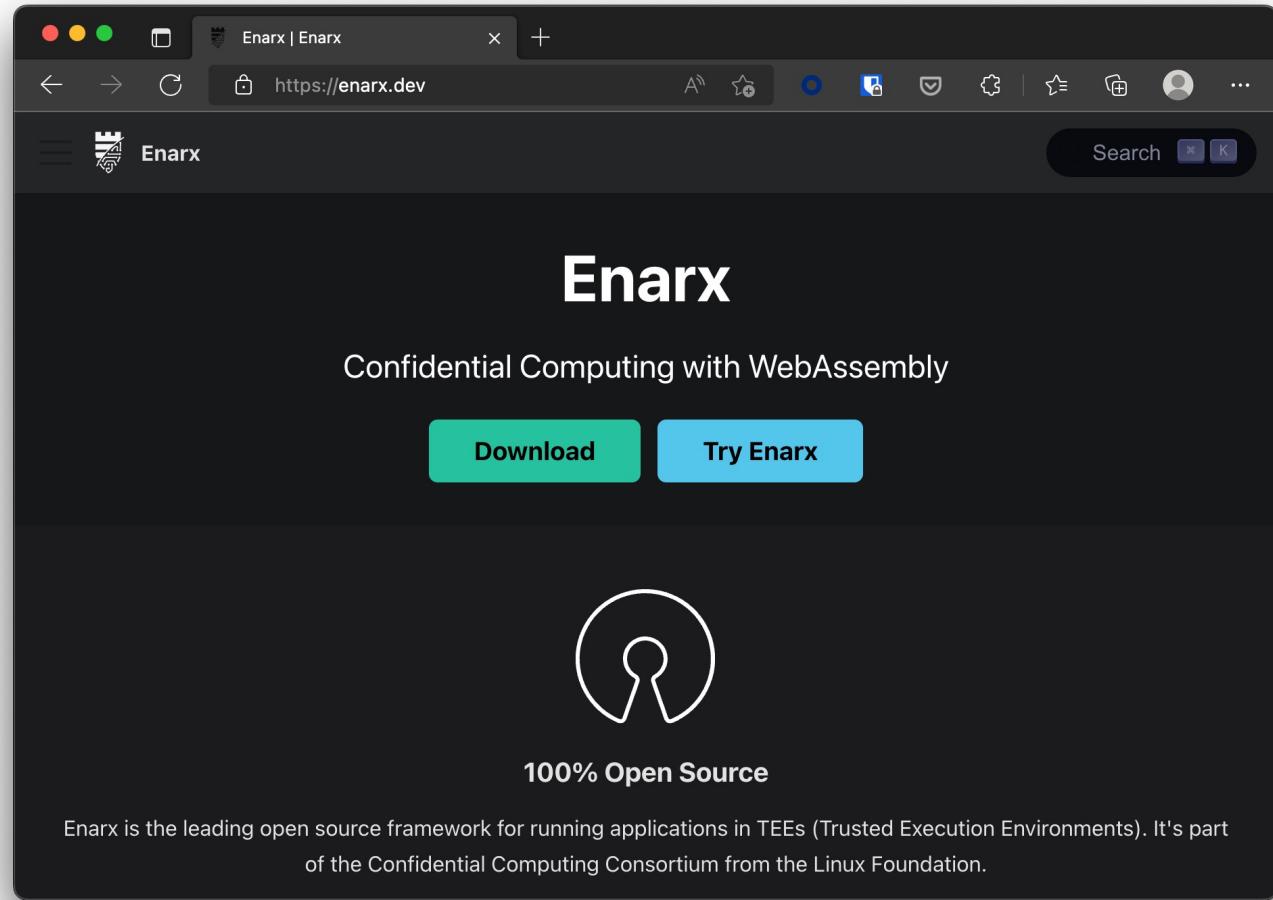
0101  
0101

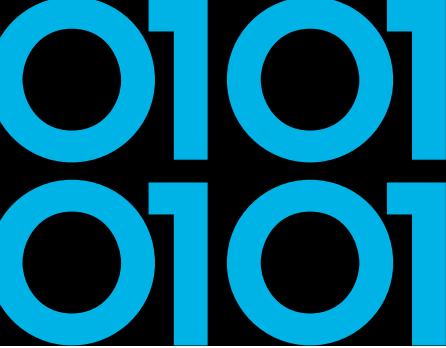
# Trusted Computing - XKCD 2166



# Enarx

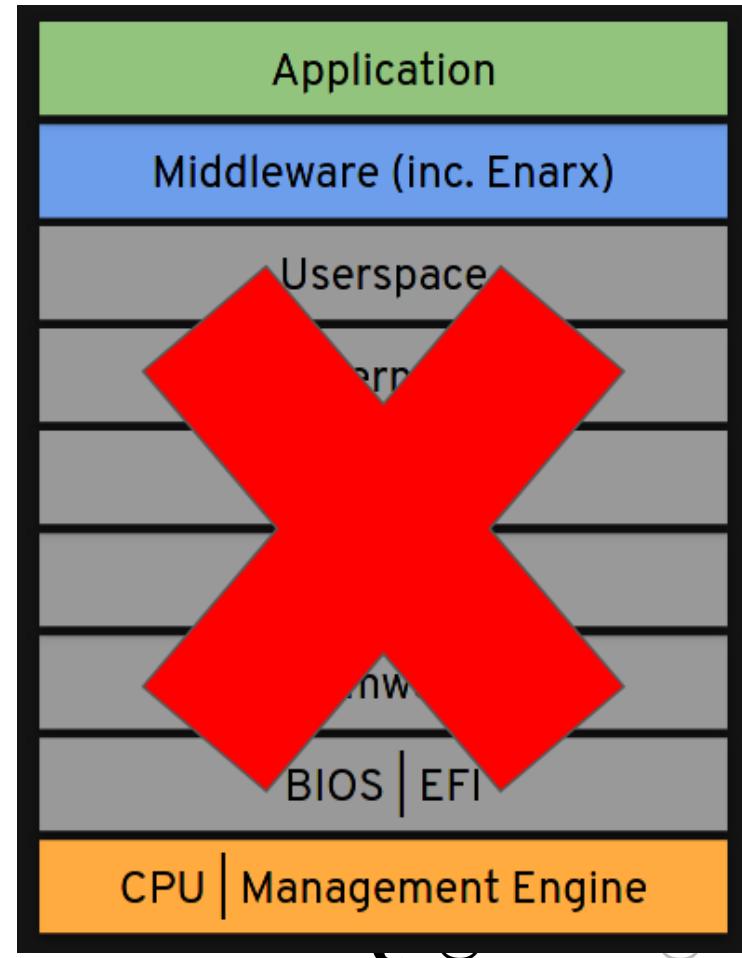
0101  
0101





# Enarx Threat Model

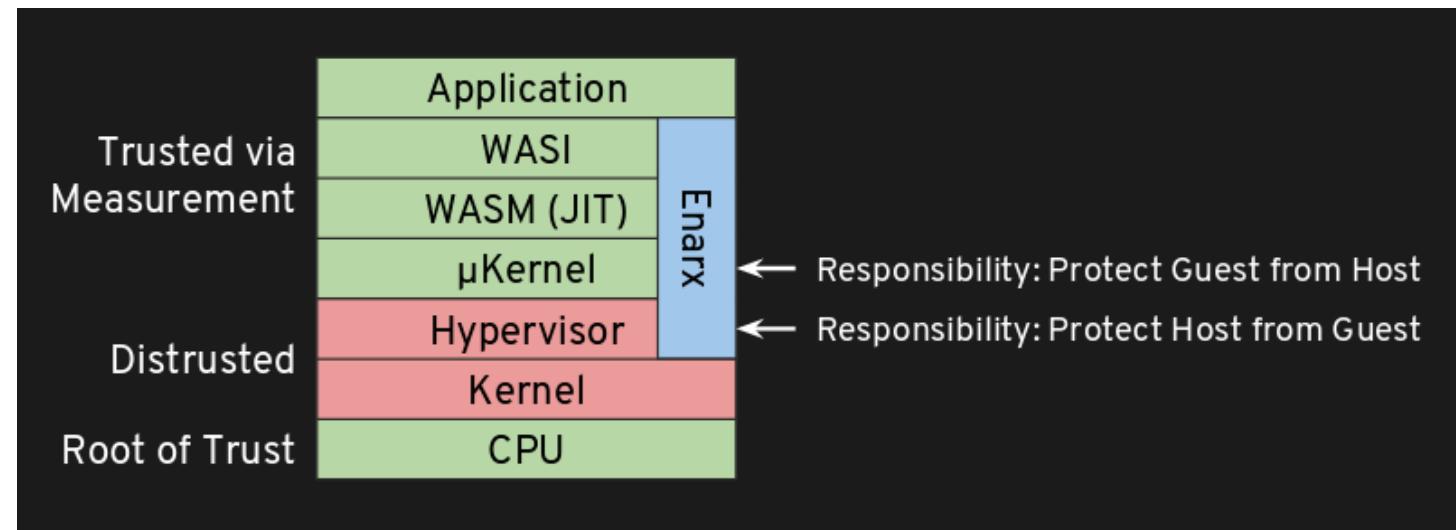
- Don't trust the host
- Don't trust the host owner
- Don't trust the host operator
- Hardware cryptographically verified
- Software audited and cryptographically verified





# Enarx

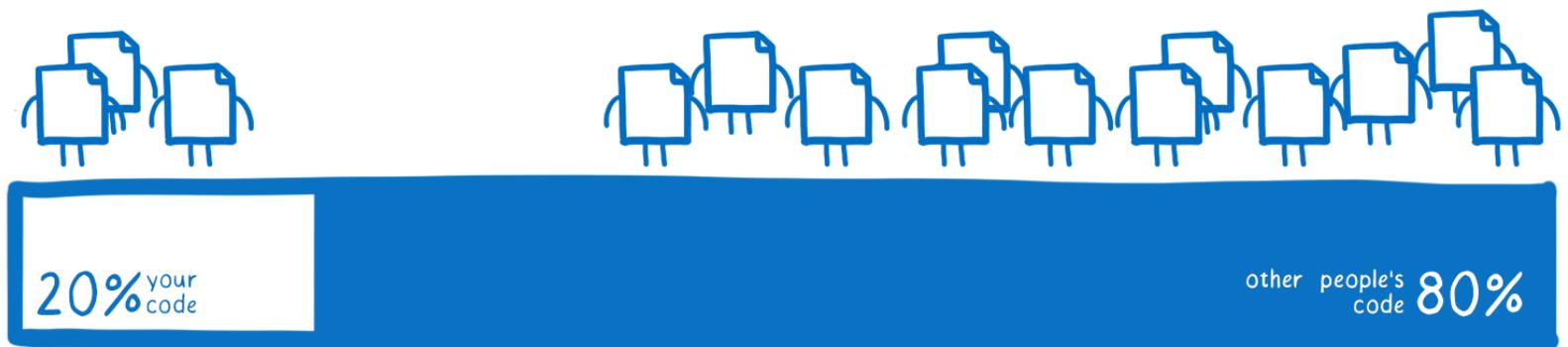
- Leverages Trusted Execution Environment (TEE) direct on processor
  - AMD's SEV, Intel's SGX and IBM's PEF
- Attestation of hardware and Enarx runtime



0101  
0101

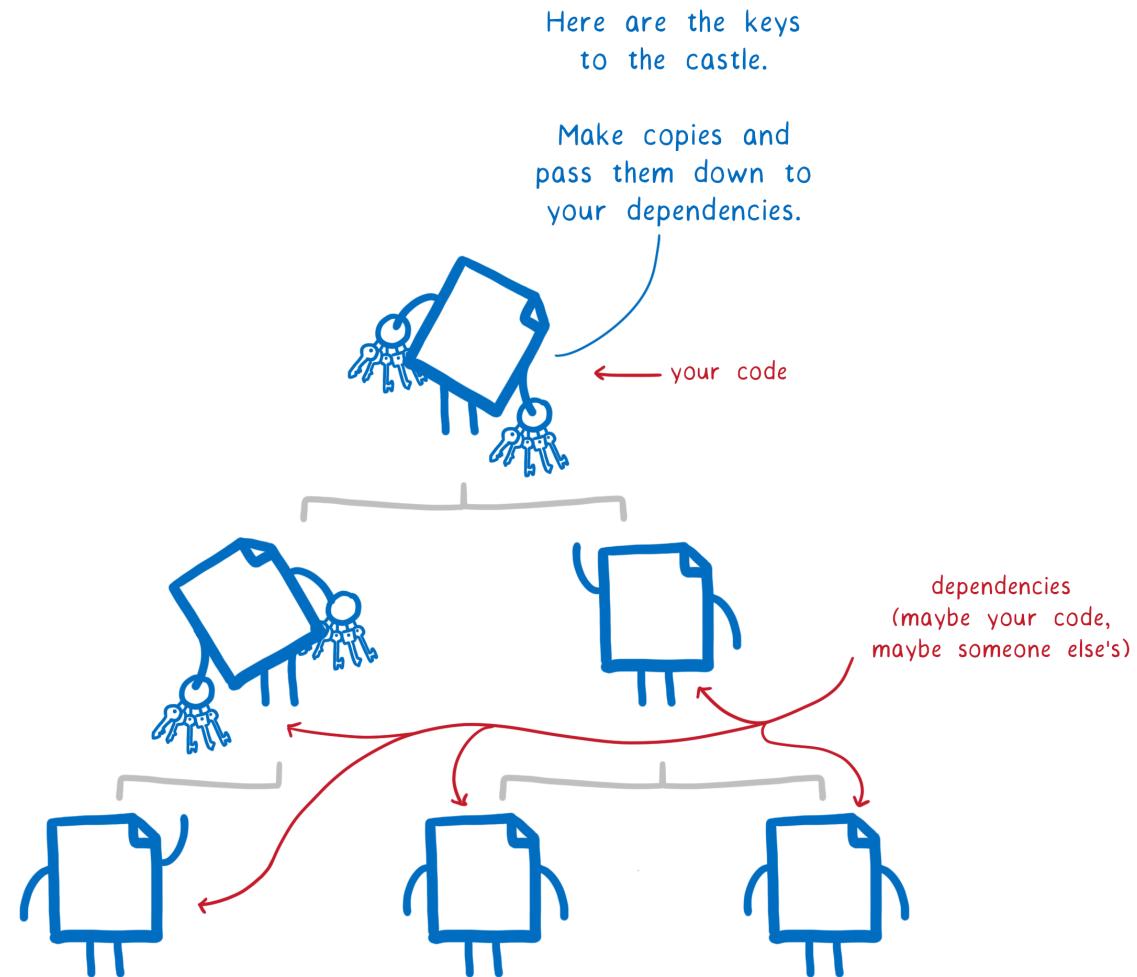
# WASM - What's next?

composition of an  
average code base



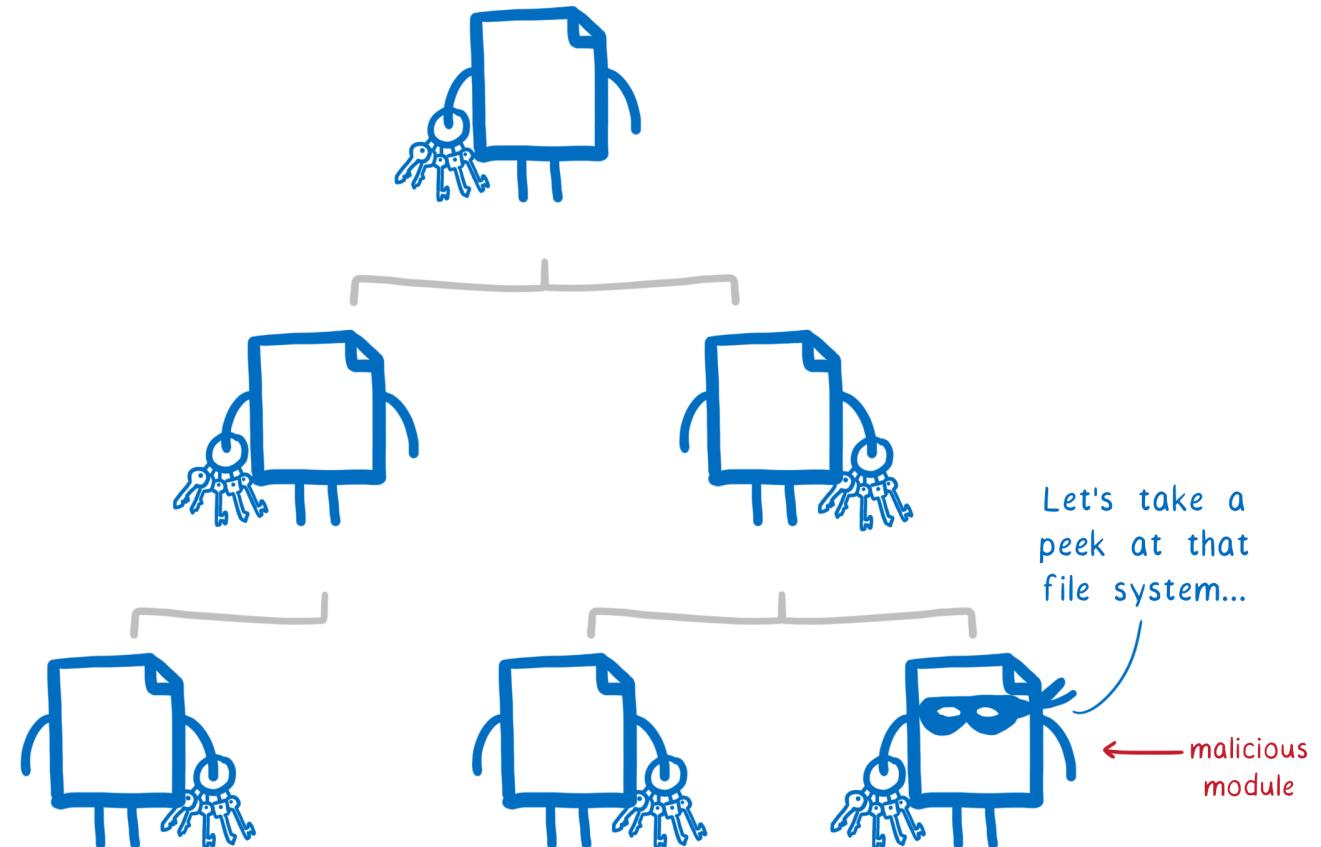
0101  
0101

# Dependencies



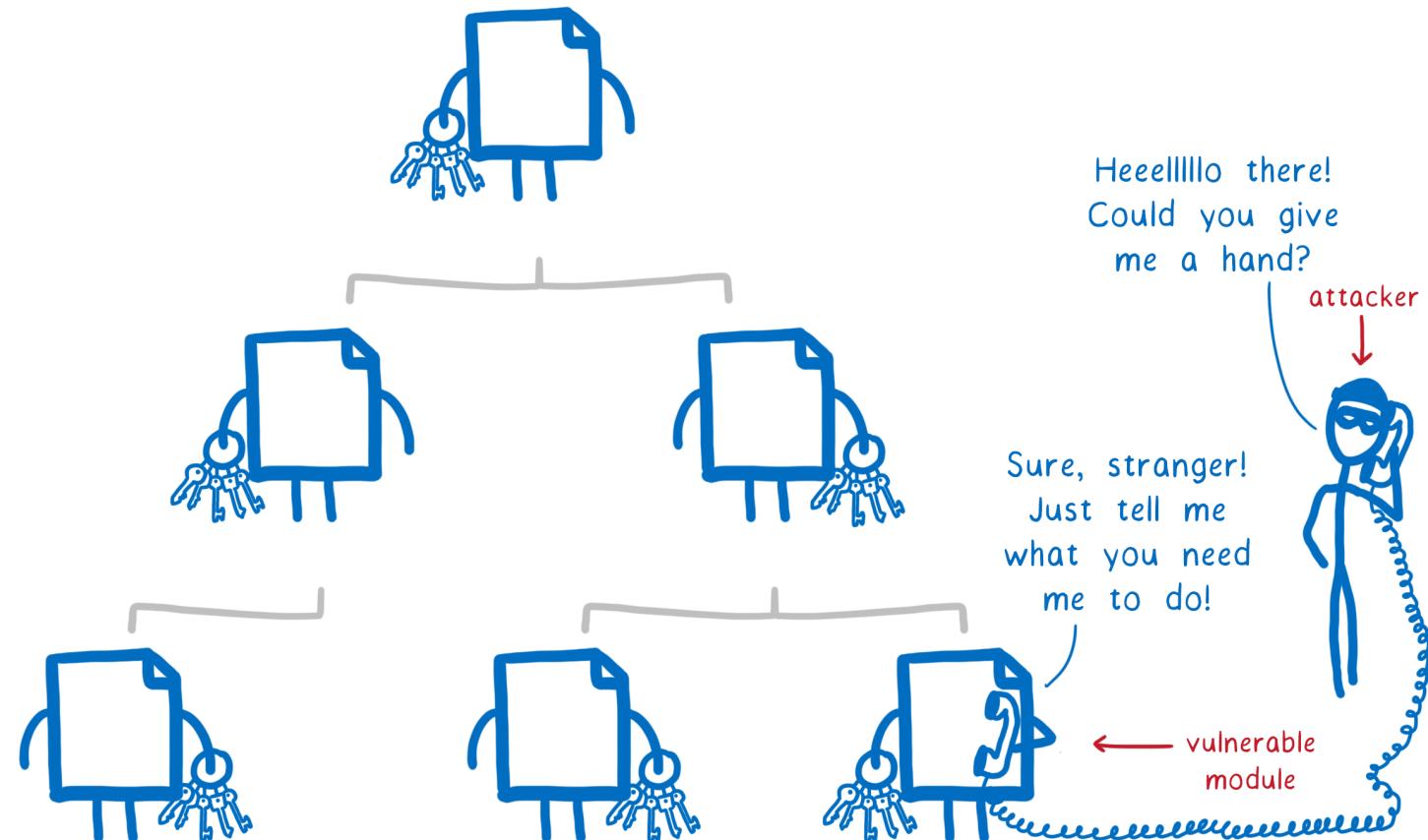
0101  
0101

# Malicious module



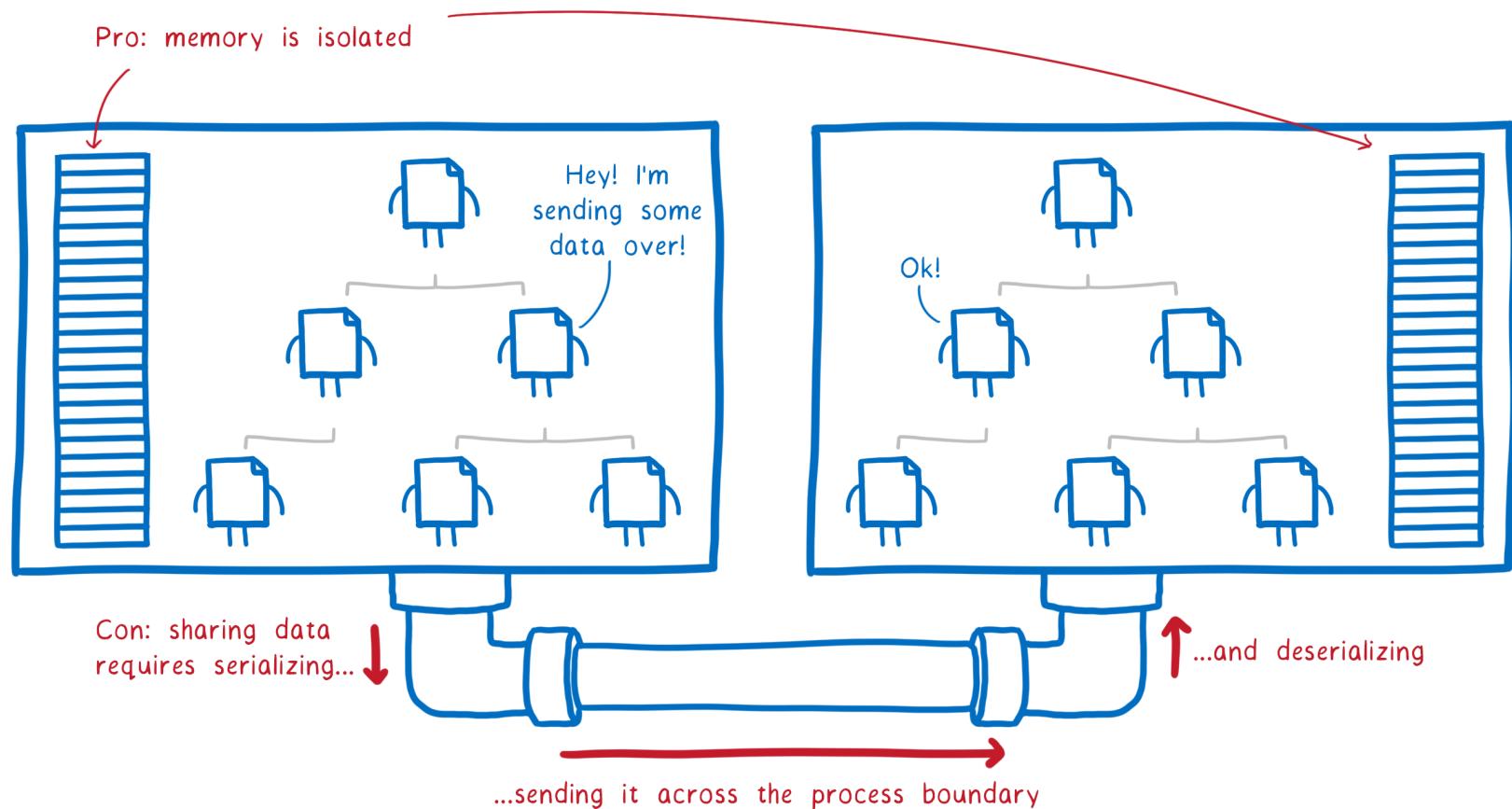
0101  
0101

# Vulnerable module



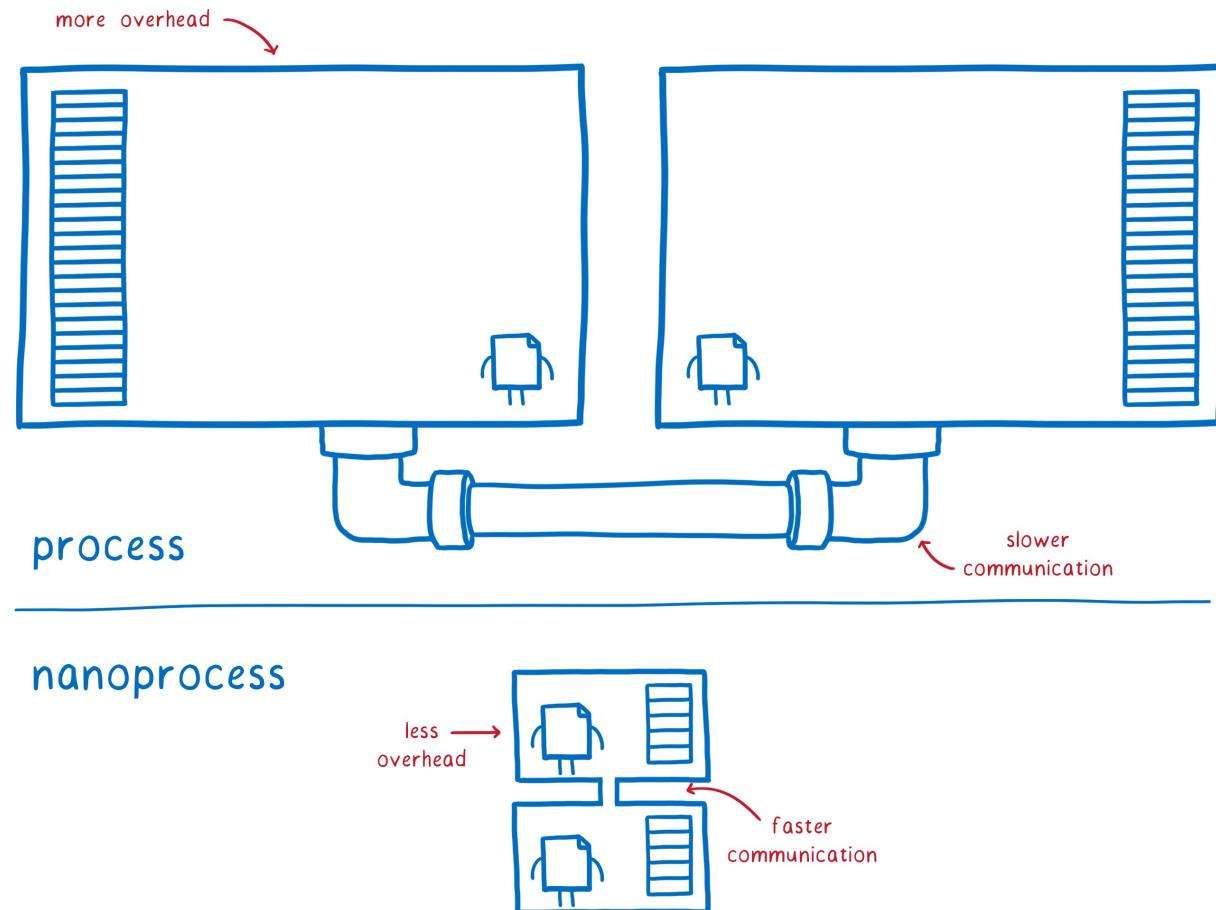
0101  
0101

# Process Isolation



0101  
0101

# WebAssembly Nano-Process

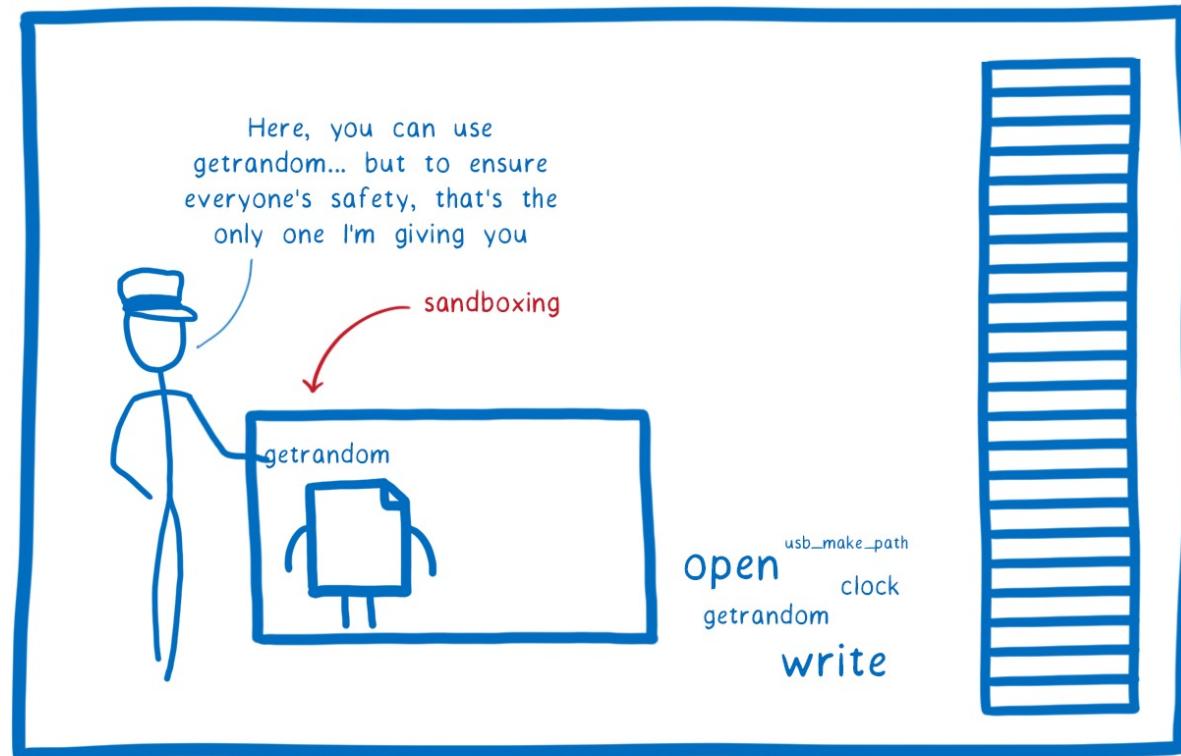


\* not drawn to scale



# WebAssembly Nano-Process

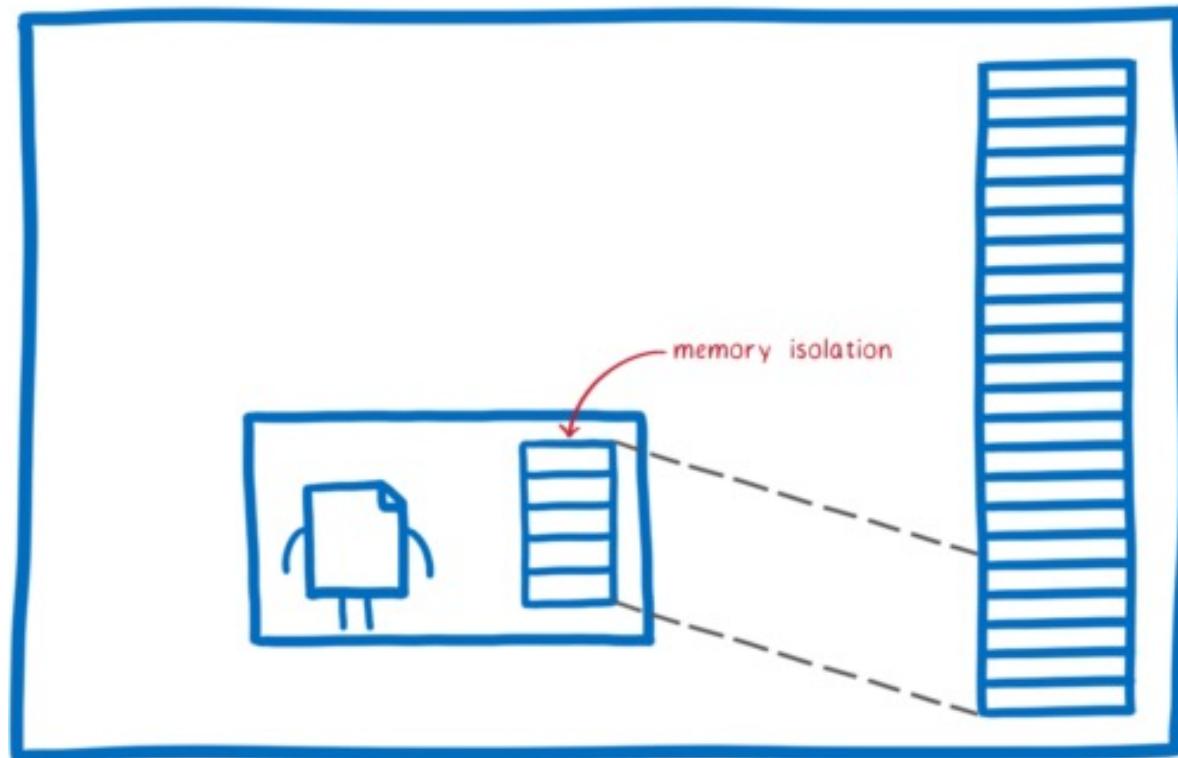
## 1. Sandboxing





# WebAssembly Nano-Process

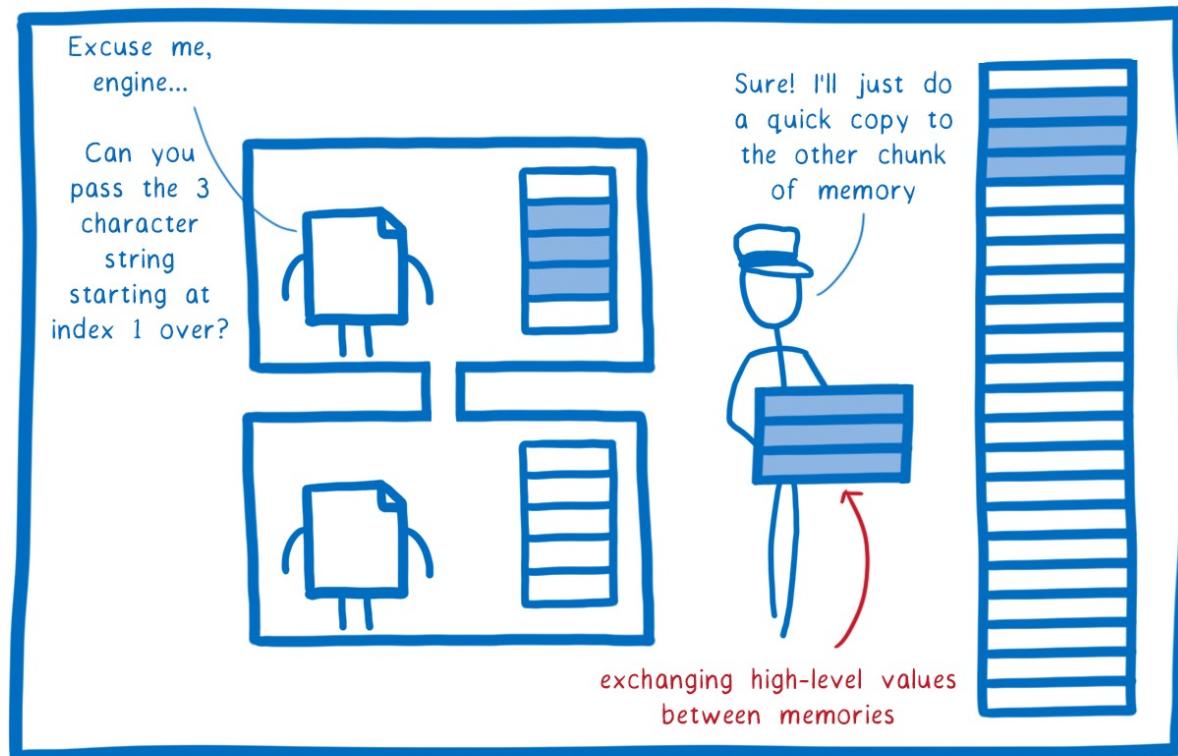
## 2. Memory model





# WebAssembly Nano-Process

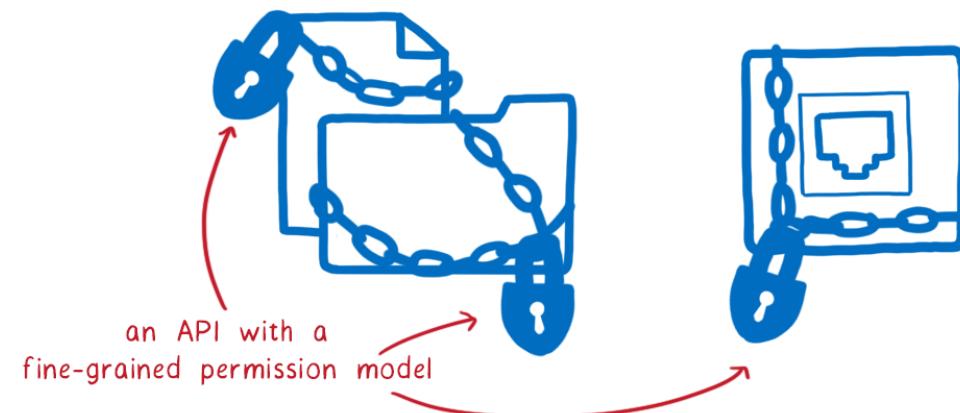
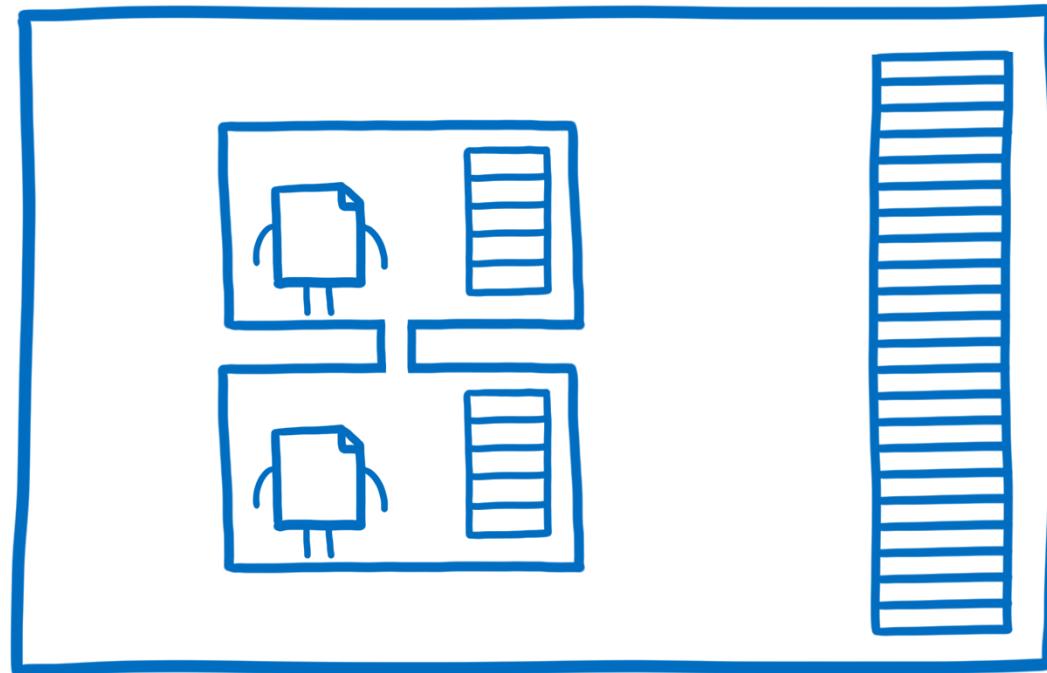
## 3. Interface Types





# WebAssembly Nano-Process

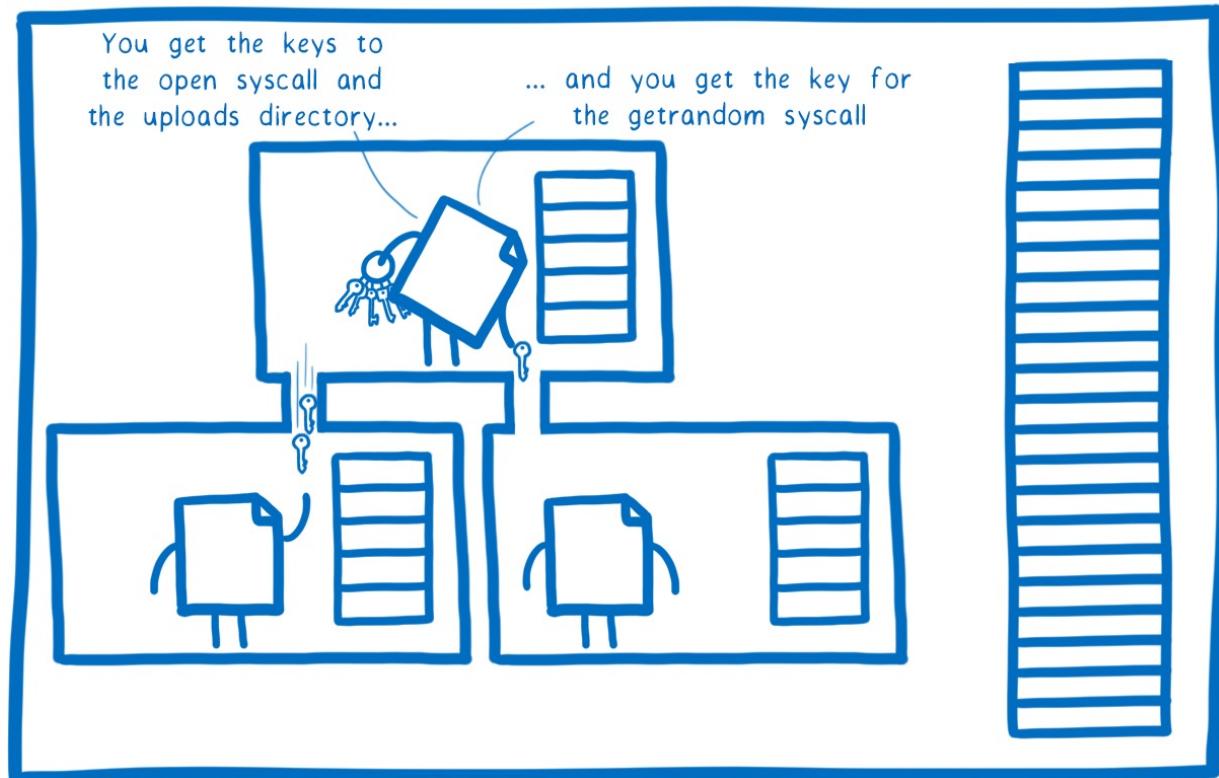
## 4. WebAssembly System Interface





# WebAssembly Nano-Process

## 5. The missing link



0101  
0101

# WebAssembly Component Model

The screenshot shows a YouTube video player with the following content:

There's a big new standards proposal we've been working on called the Component Model

It sits in-between WASI and Core WebAssembly and provides a way to compose code together like Lego bricks.

So what this talk will cover is:

- The path leading up to the Component Model
- What problems the Component Model is working to solve
- What the developer experience is shaping up to look like
- What's next

Keynote: The Path to Components - Luke Wagner, Distinguished Engineer, Fastly

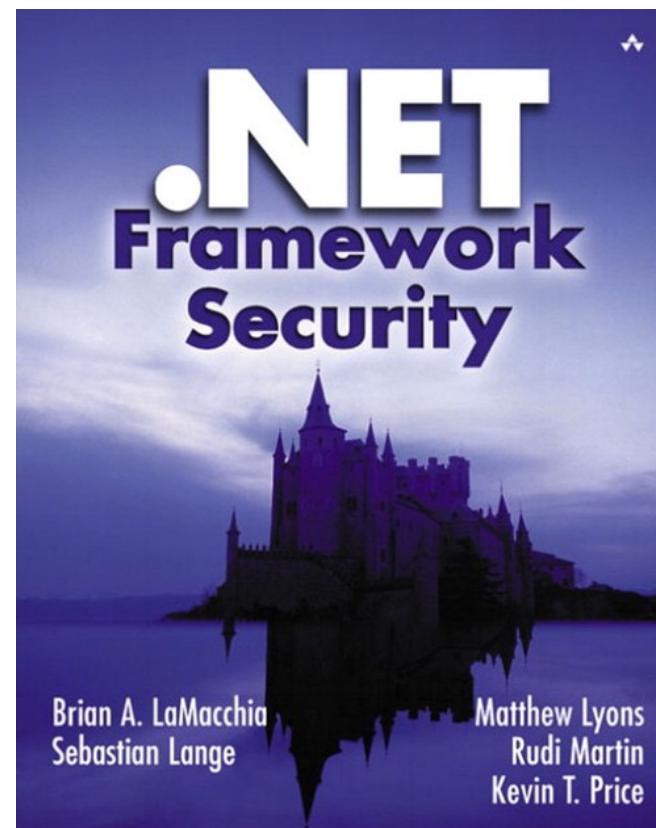
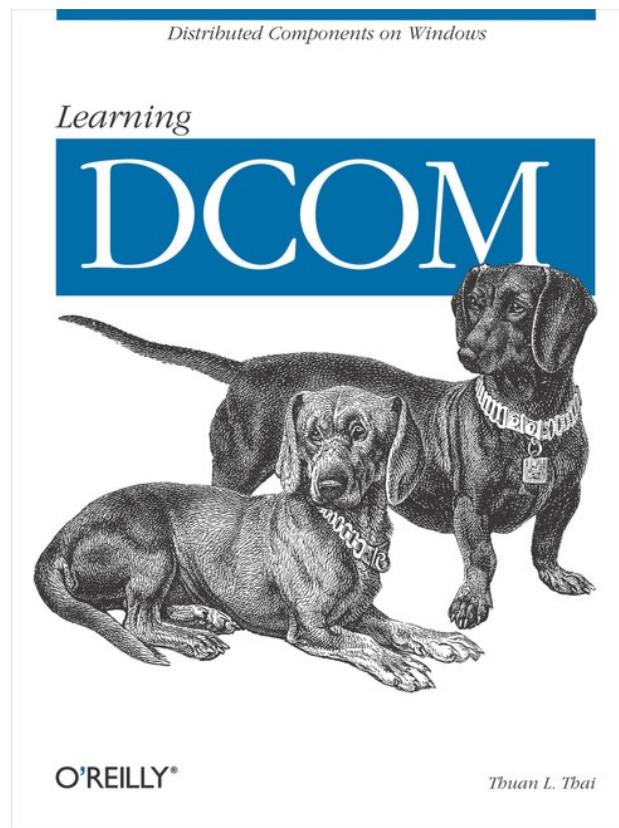
Cloud Native Wasm Day NA 2022

CNCF [Cloud Native Computing Foundation] - 2 / 19

NDC { Security } nielstanis@infosec.exchange

0101  
0101

# Have we seen this before?





# Runtimes and Security

- Most security research published focusses on correctness of WASM runtimes/VM's
- Bytecode Alliance Blogpost September 2022:
  - “Security and Correctness in Wasmtime”
  - Written in Rust → Using all it's LangSec features
  - Continues Fuzzing & formal verification
  - Security process & vulnerability disclosure



# Conclusion

- Cloud Native ❤️ WebAssembly
- WebAssembly has a lot of potential to be used to run, extend, and secure your .NET applications
- Its as secure as the WebAssembly runtime implementation!
- I like top-down approach Bytecode Alliance is taking in moving forward, feedback will change



# Questions?

- <https://github.com/nielstanis/ndcsecurity2023>
- ntanis at Veracode.com
- @nielstanis@infosec
- <https://blog.fennec.dev>
- Takk! Thank you!