Using WebAssembly to run, extend, and secure your [.NET | Java | Python | Go | Rust | …] application

Niels Tanis

WeAreDevelopers World Congress

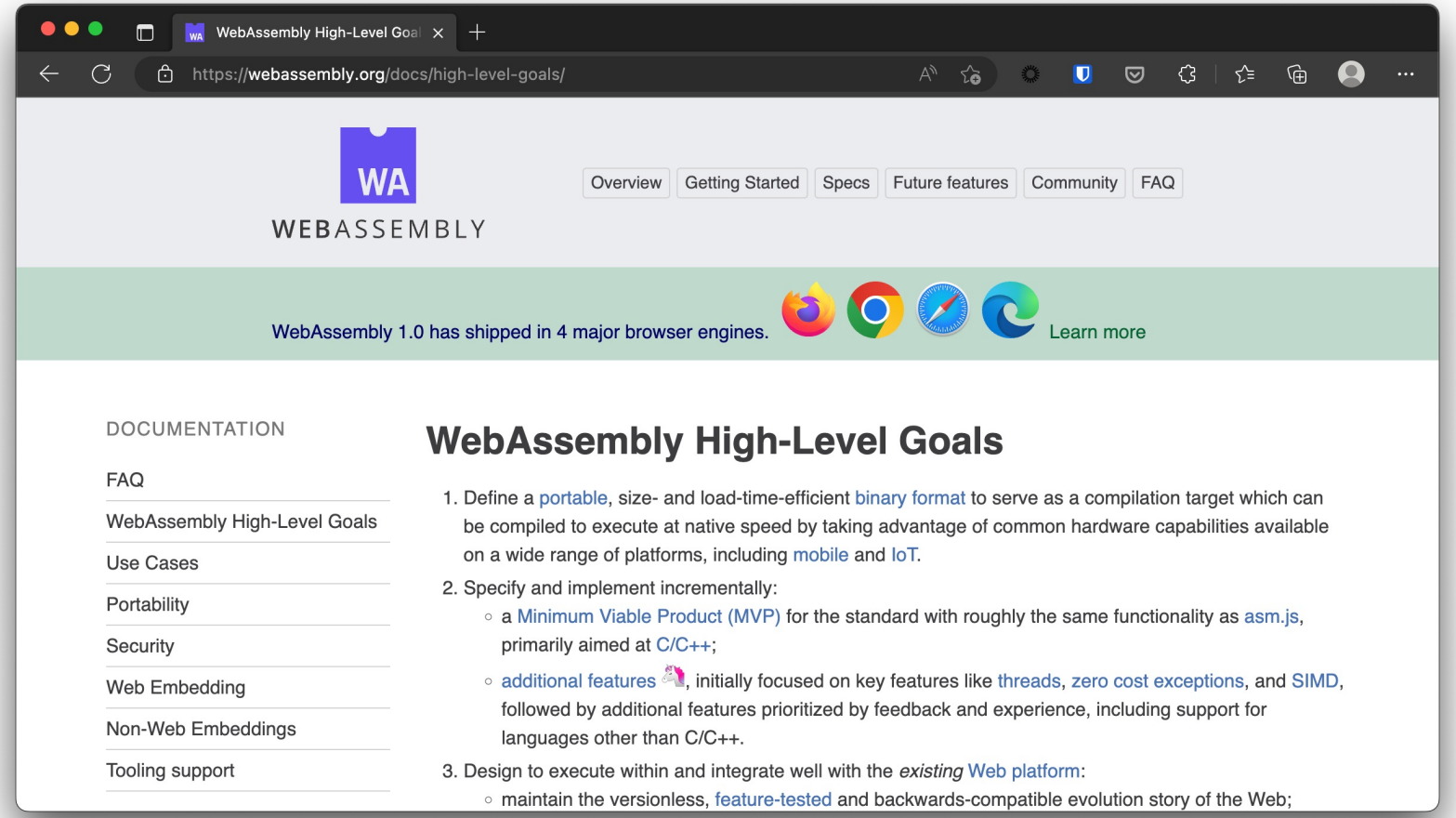
VERACODE

Who am I?

- Niels Tanis
- Sr. Principal Security Researcher
  - Background .NET Development, Pentesting/ethical hacking, and software security consultancy
  - Research on static analysis for .NET apps
  - Enjoying Rust!
- Microsoft MVP – Developer Technologies

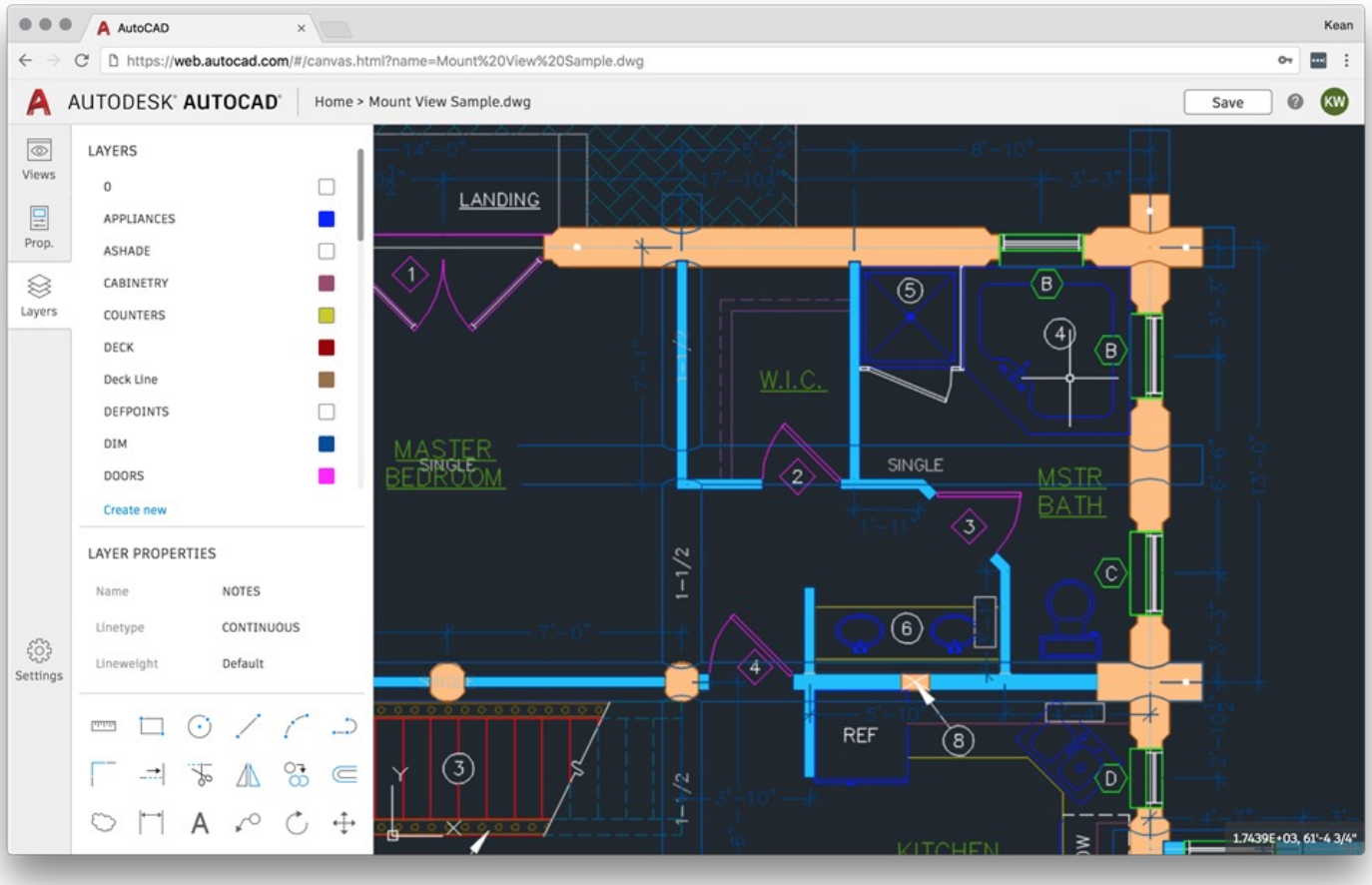
@nielstanis@infosec.exchange

WebAssembly



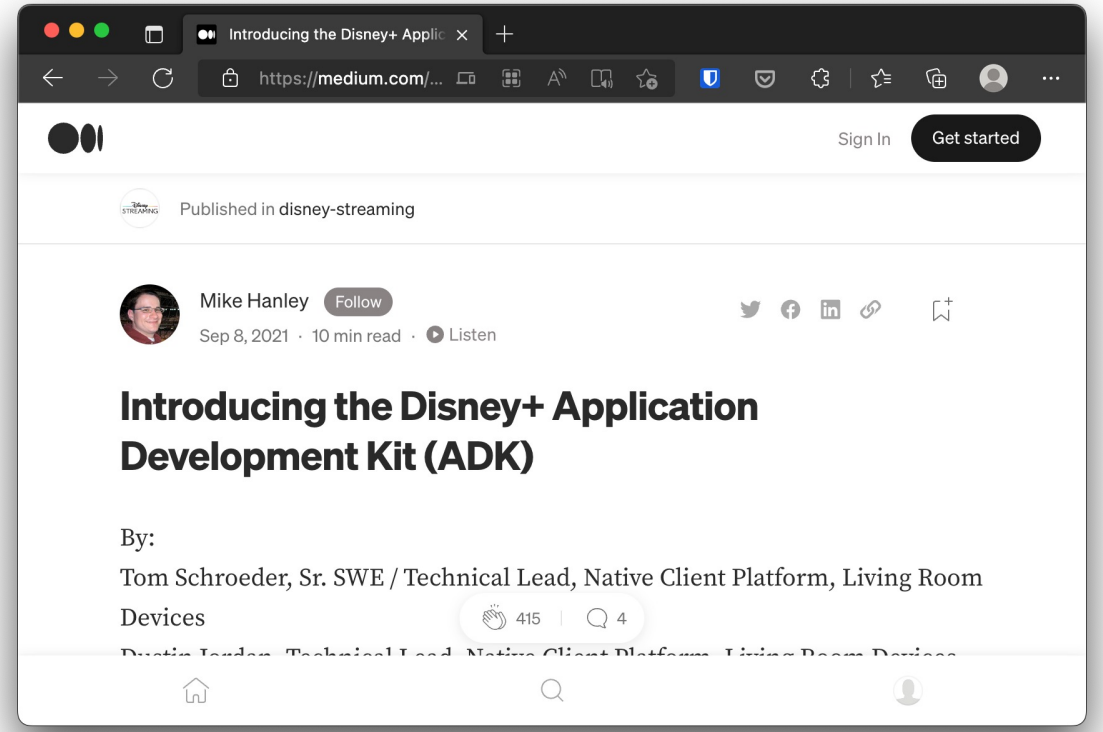
@nielstanis@infosec.exchange

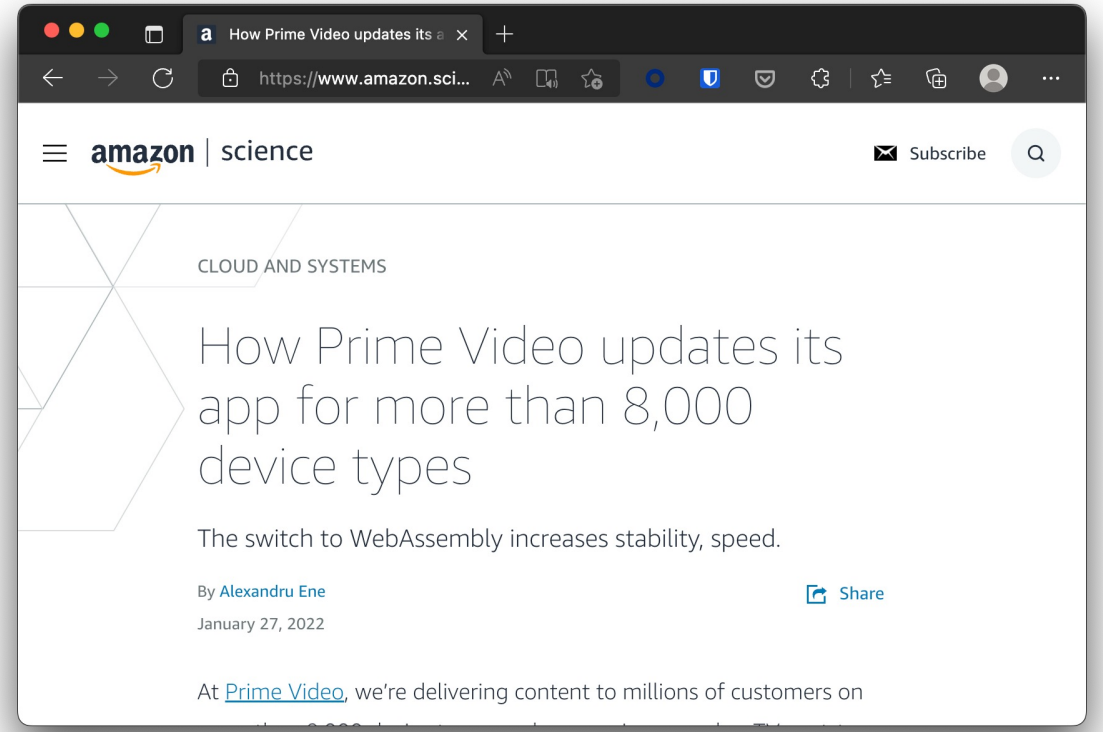
WebAssembly - AutoCAD



@nielstanis@infosec.exchange

WebAssembly - SDK's





@nielstanis@infosec.exchange

Agenda

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

WebAssembly Design

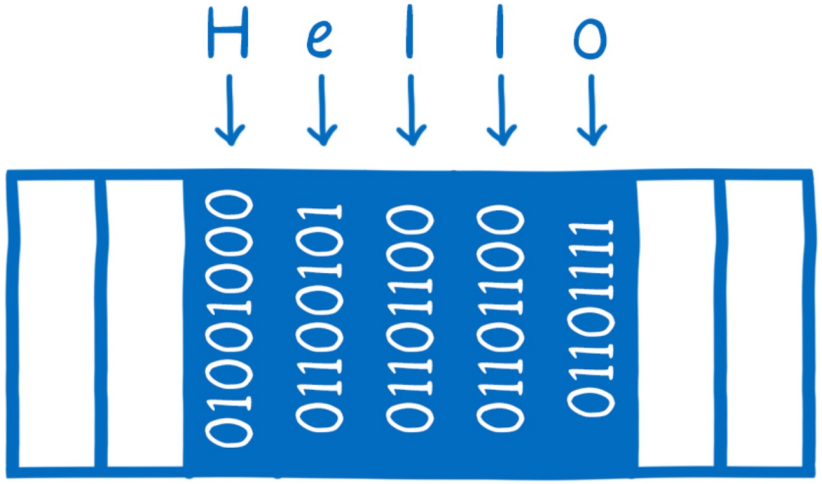
- **Be fast, efficient, and portable**
  - Executed in near-native speed across different platforms
- **Be readable and debuggable**
  - In low-level bytecode but also human readable
- **Keep secure**
  - Run on sandboxed execution environment
- **Don't break the web**
  - Ensure backwards compatibility

WebAssembly

- Binary instruction format for stack-based virtual machine similar to Java running bytecode and .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

WebAssembly Memory

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

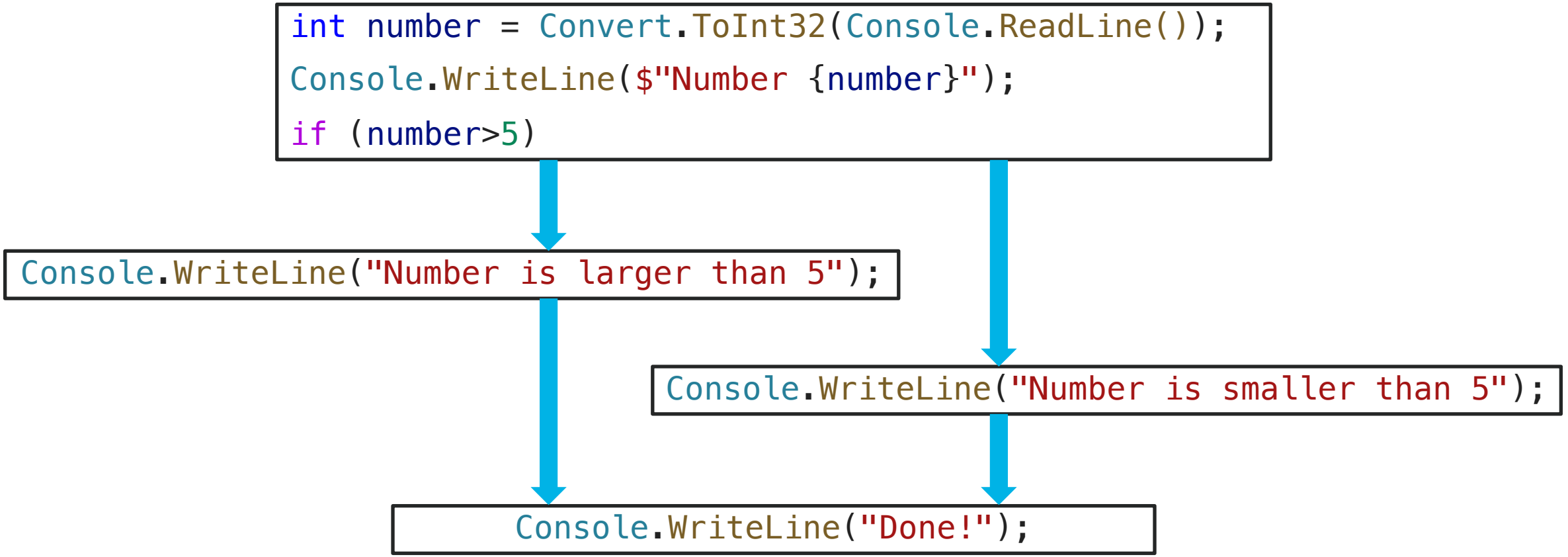


WebAssembly Control-Flow Integrity

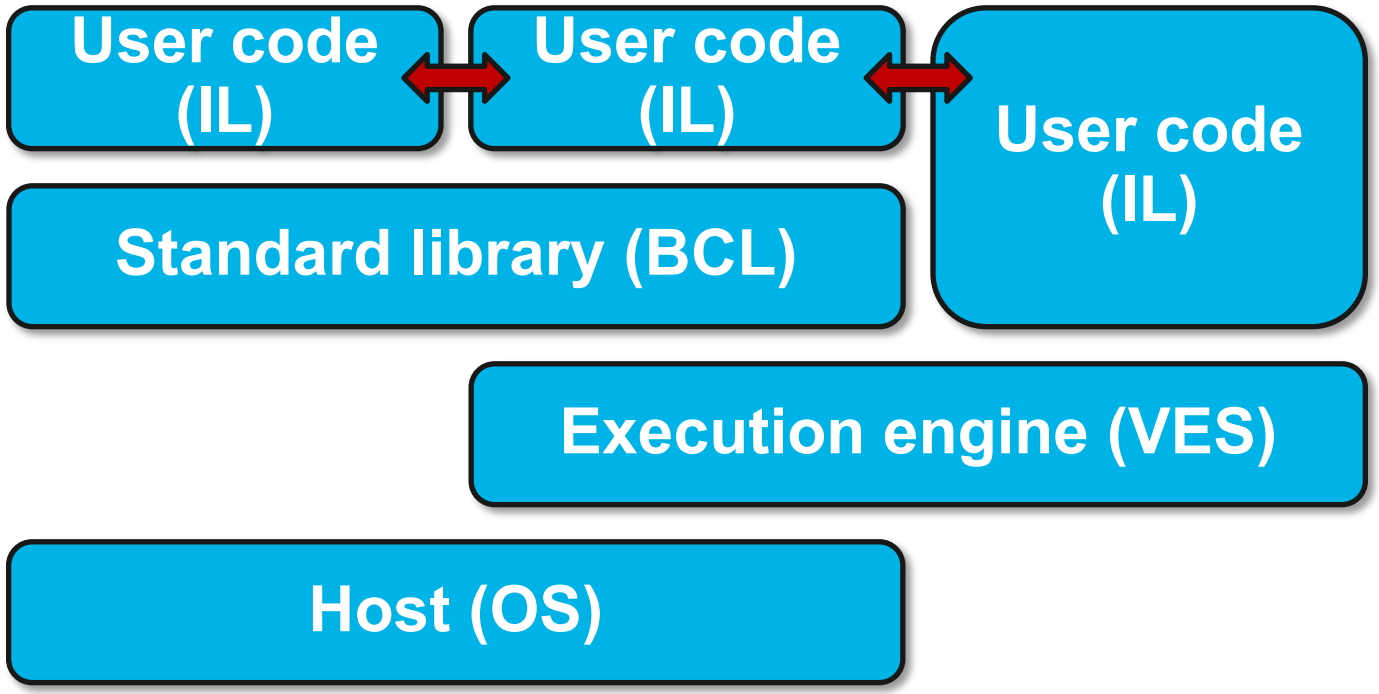
```
int number = Convert.ToInt32(Console.ReadLine());
Console.WriteLine($"Number {number}");
if (number>5)
{
        Console.WriteLine("Number is larger than 5");
}
else
{
        Console.WriteLine("Number is smaller than 5");
}
Console.WriteLine("Done!");
```

@nielstanis@infosec.exchange

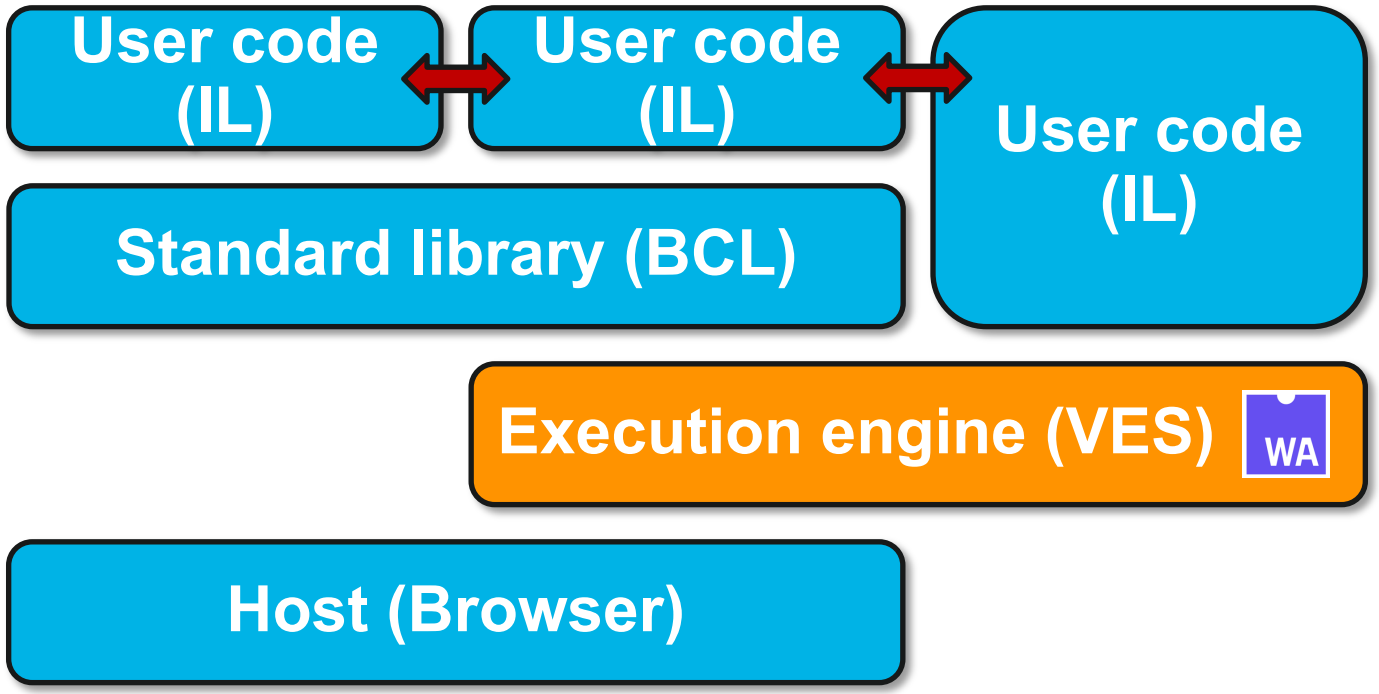
WebAssembly Control-Flow Integrity



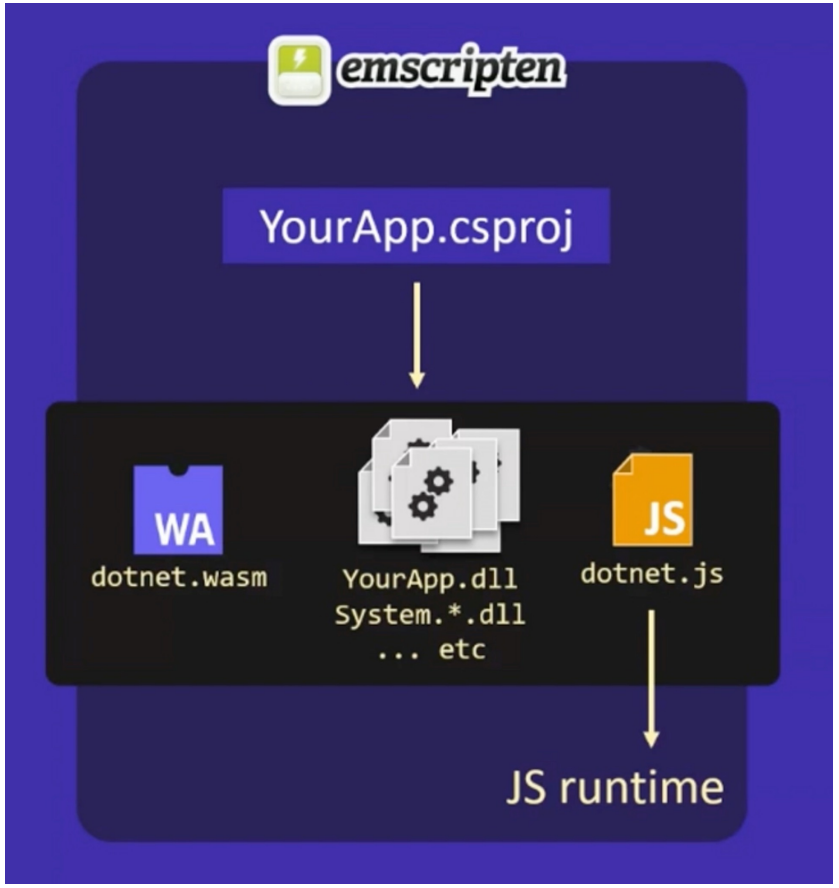
Running .NET on WebAssembly



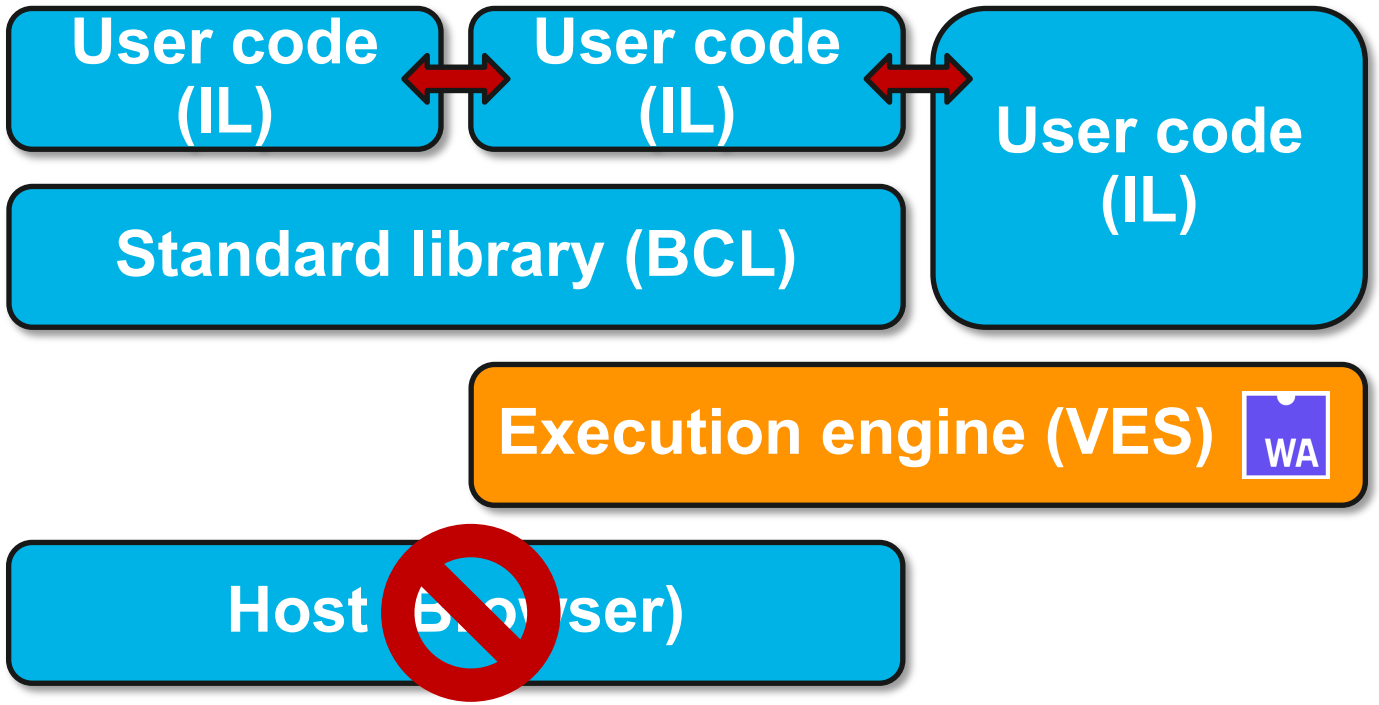
Running .NET on WebAssembly



Blazor WebAssembly



Running .NET on WebAssembly



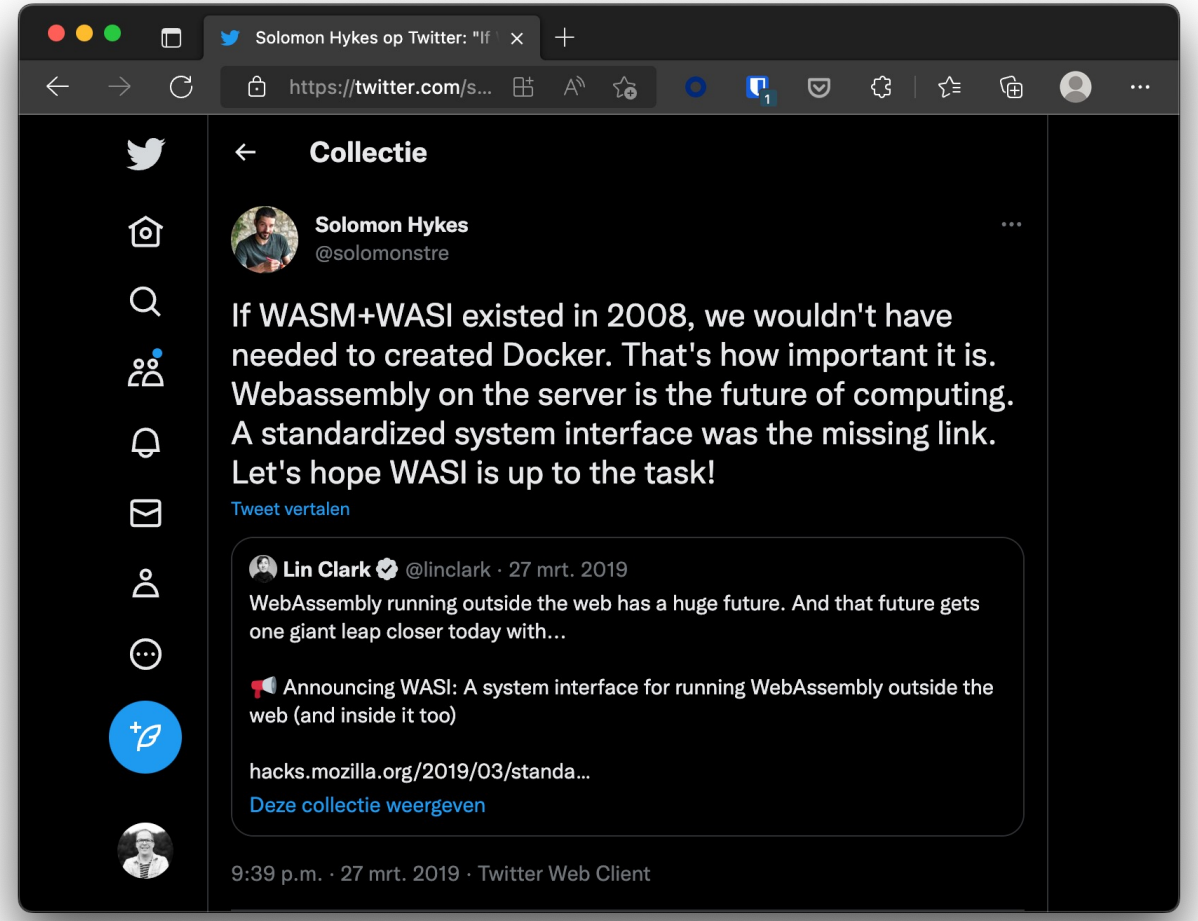
WebAssembly System Interface WASI

- 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? ☺

Docker vs WASM & WASI



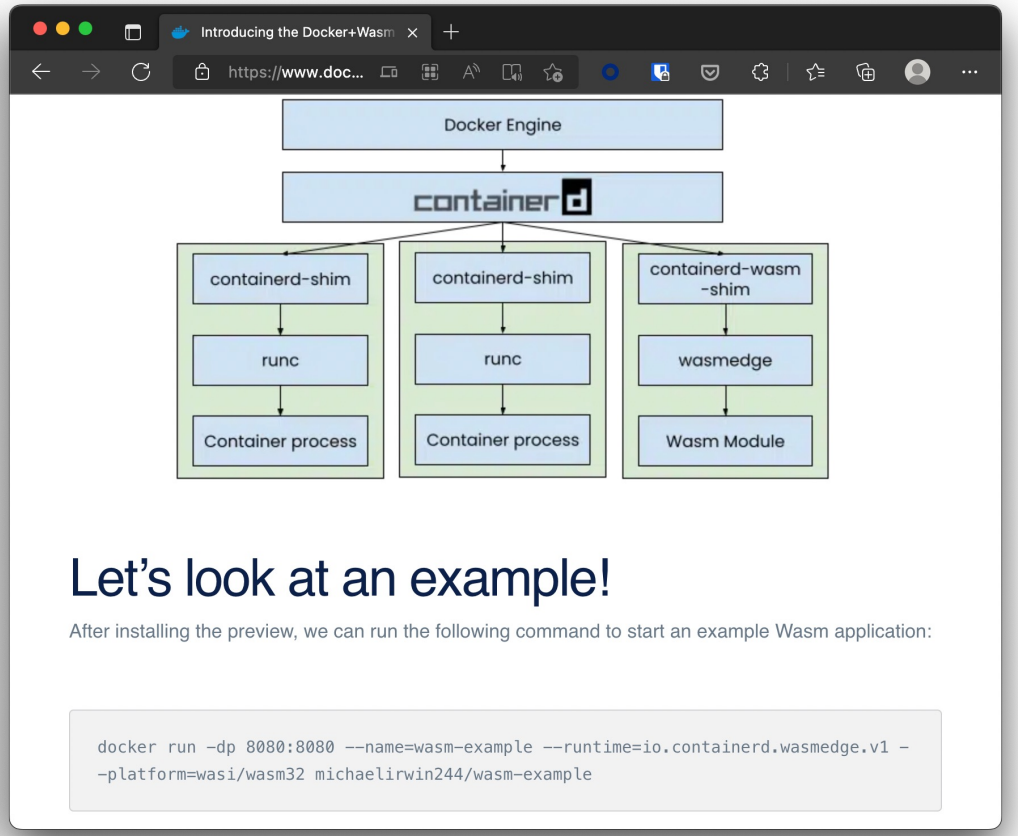
@nielstanis@infosec.exchange

Docker vs WASM & WASI

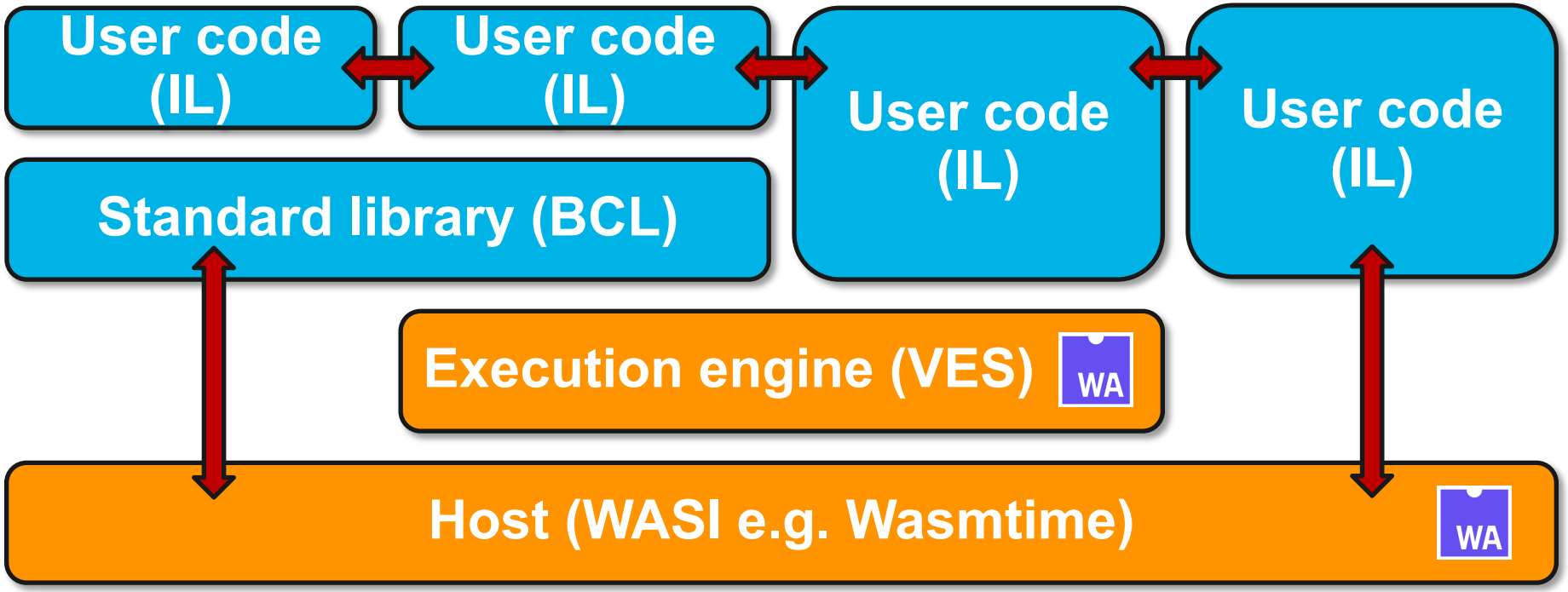
@nielstanis@infosec.exchange

Docker & WASM

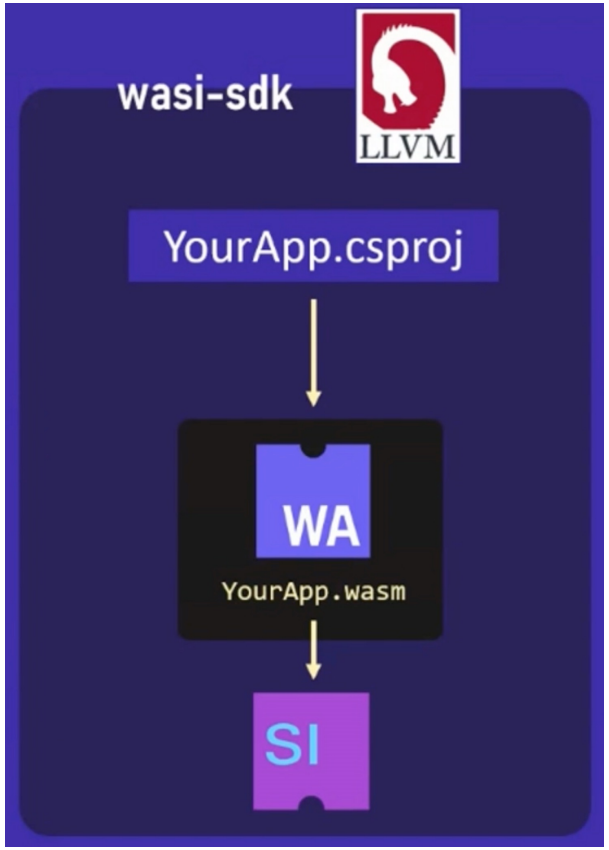

Introducing the Docker+Wasm
https://www.doc...
Wasm is a fast, light alternative to Linux containers — try it out today in the Docker+Wasm Technical Preview
docker +WASM
Introducing the Docker+Wasm Technical Preview
MICHAEL IRWIN
Oct 24 2022
The Technical Preview of Docker+Wasm is now available! Wasm has been producing a lot of buzz recently, and this feature will make it easier for you to quickly build applications targeting Wasm runtimes.


Introducing the Docker+Wasm
https://www.doc...
Docker Engine
containerd
containerd-shim
runc
Container process
containerd-shim
runc
Container process
containerd-wasm -shim
wasmedge
Wasm Module
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



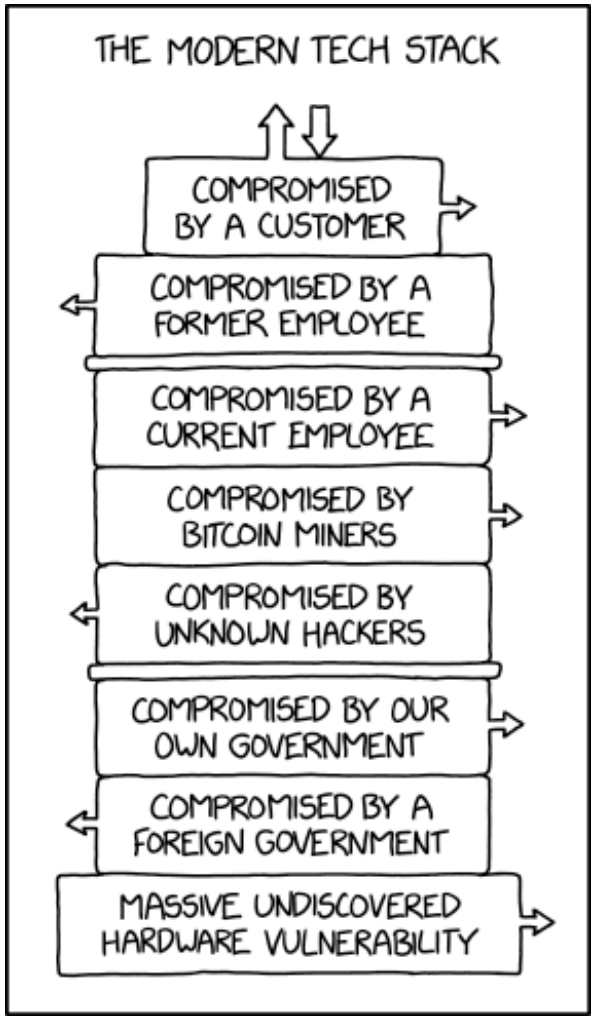
Experimental WASI SDK for .NET



Extending .NET with WASM

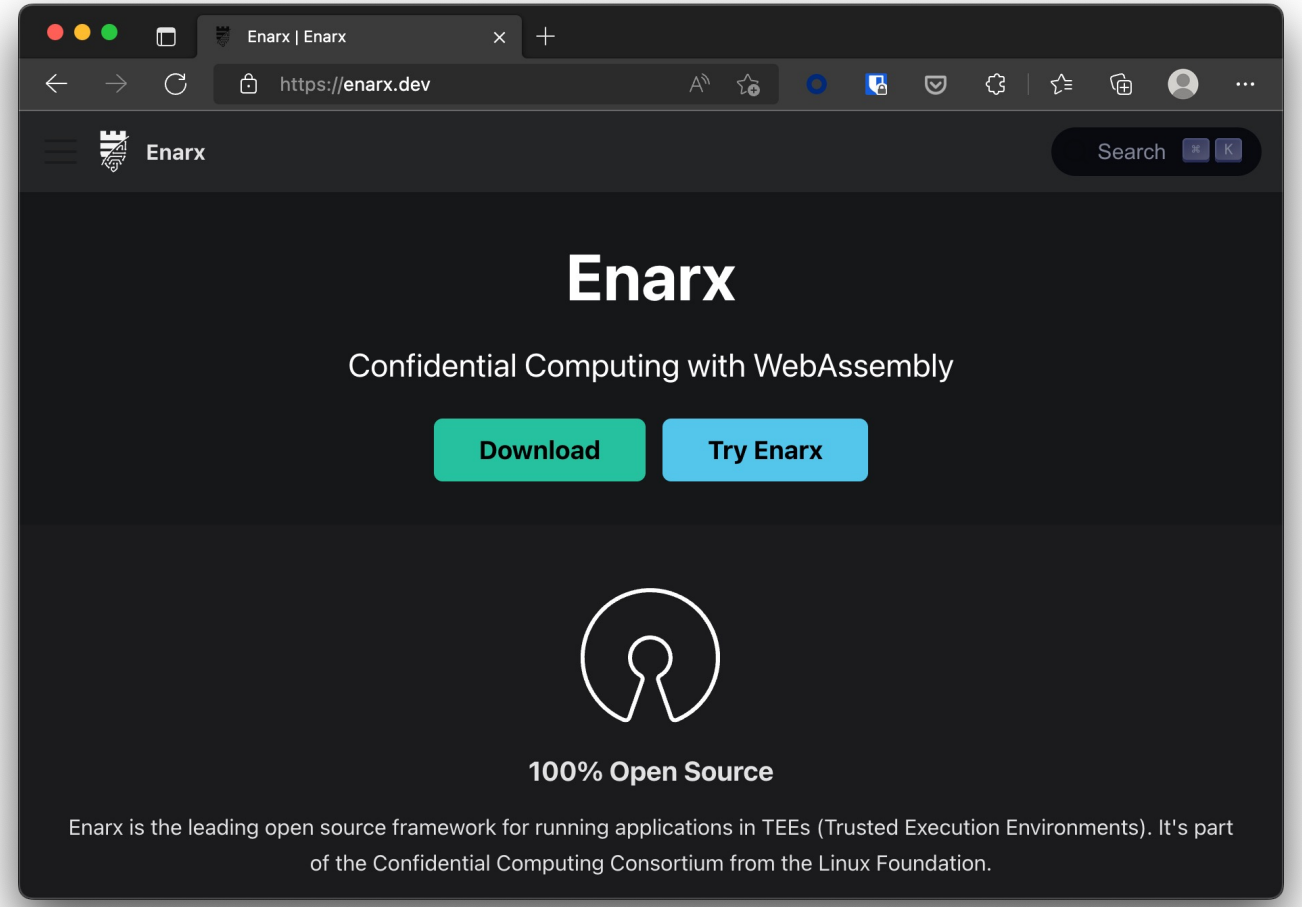
- WasmTime.NET NuGet package
- Can run WASM inside of any .NET application
- Extend with Rust based WASM module
- Demo time!

Trusted Computing - XKCD 2166



@nielstanis@infosec.exchange

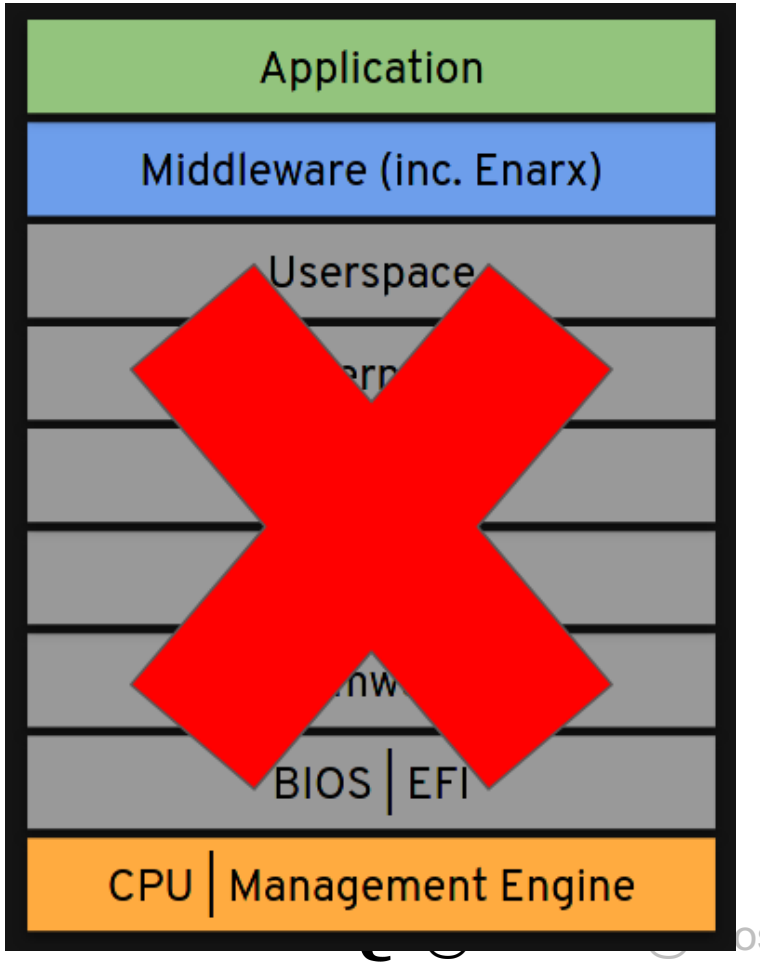
Enarx



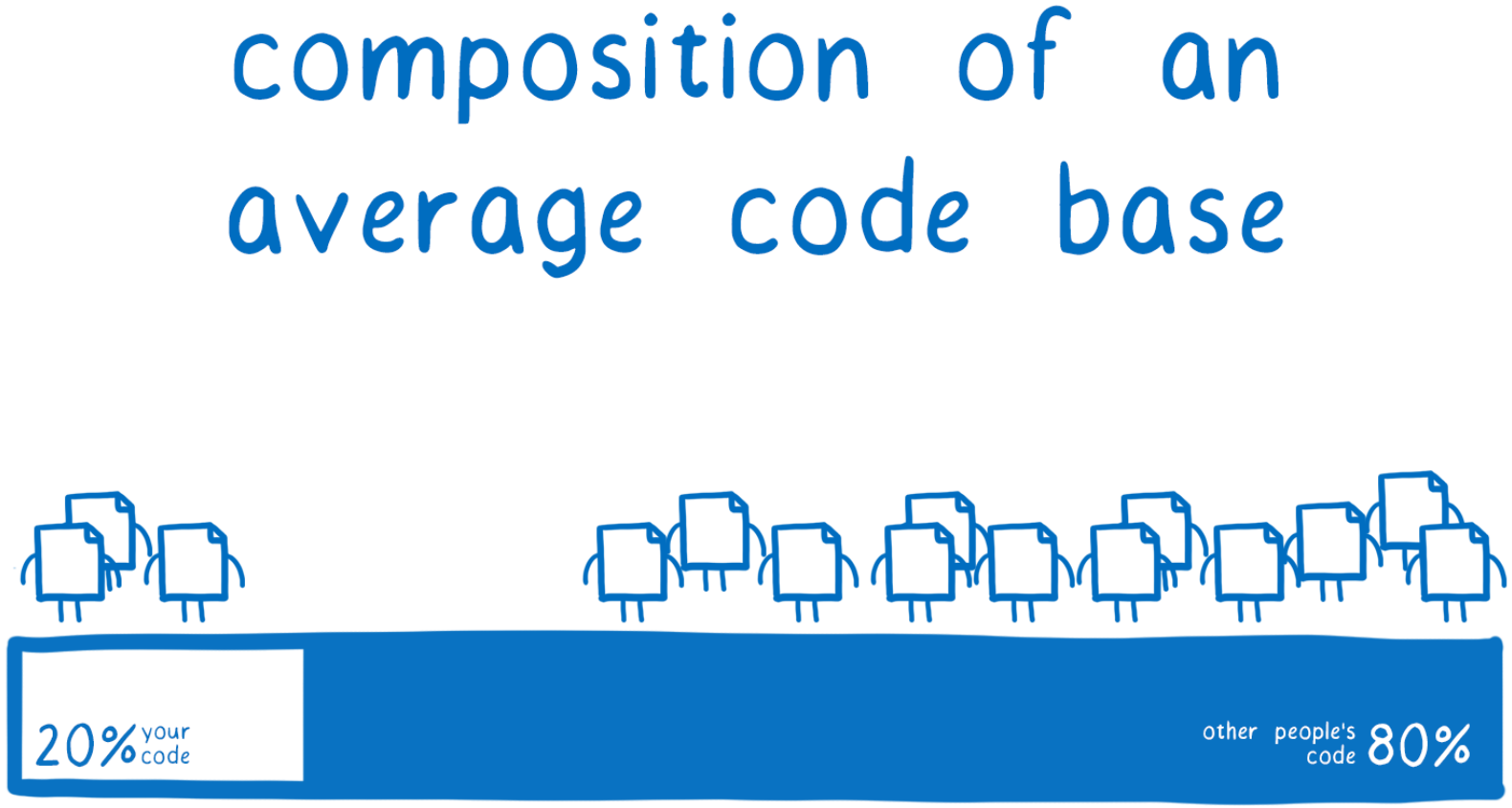
@nielstanis@infosec.exchange

Enarx Threat Model

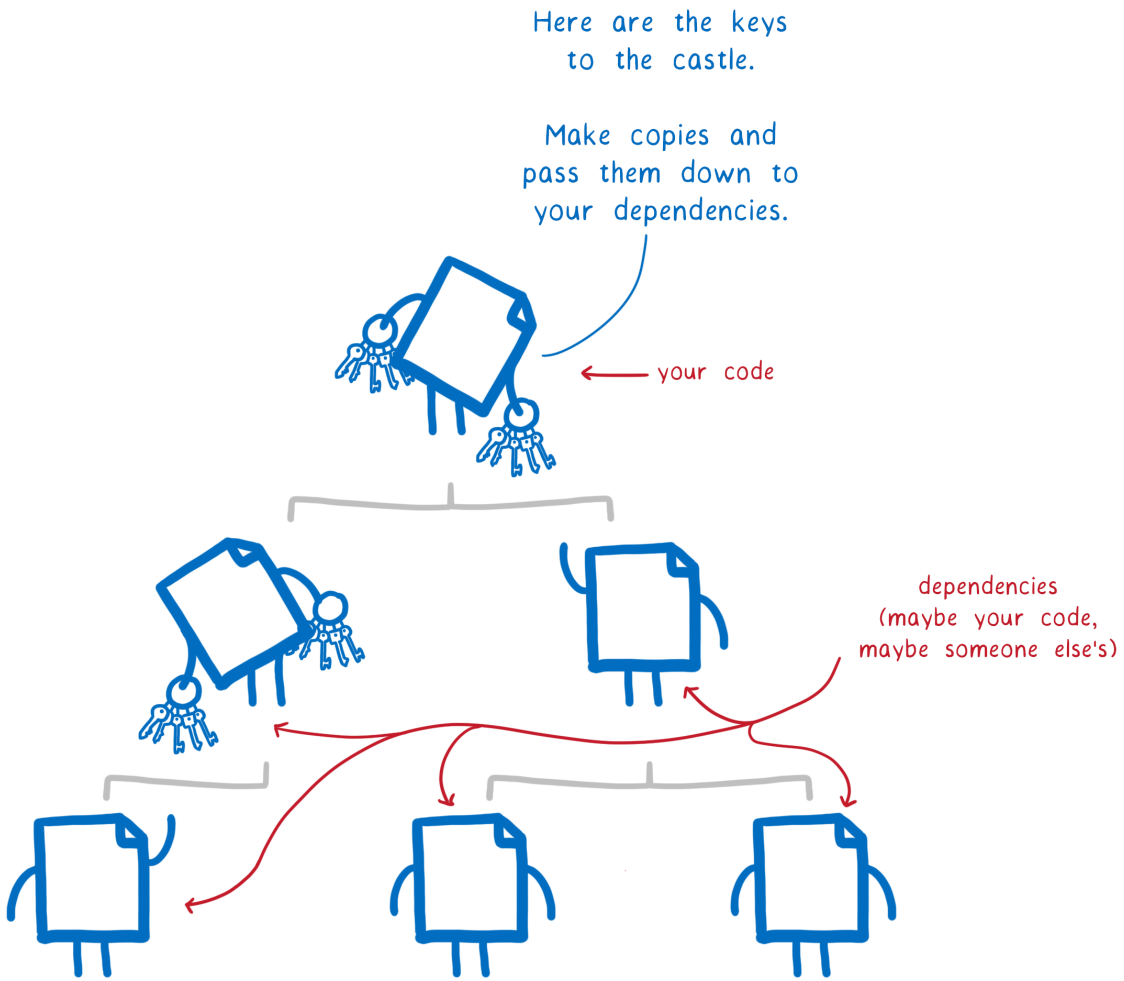
- Don’t trust the host
- Don’t trust the host owner
- Don’t trust the host operator
- Hardware cryptographically verified
- Software audited and cryptographically verified



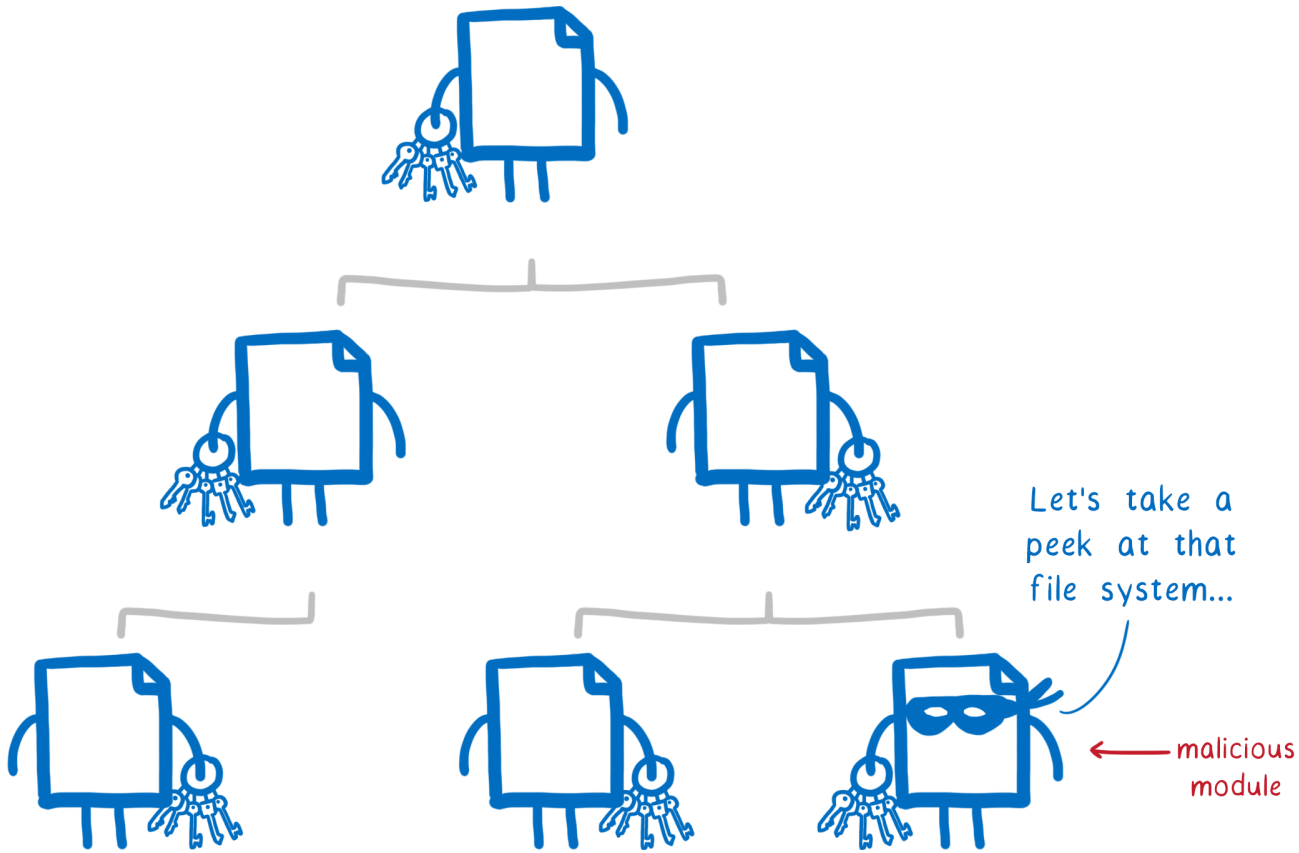
WASM – What's next?



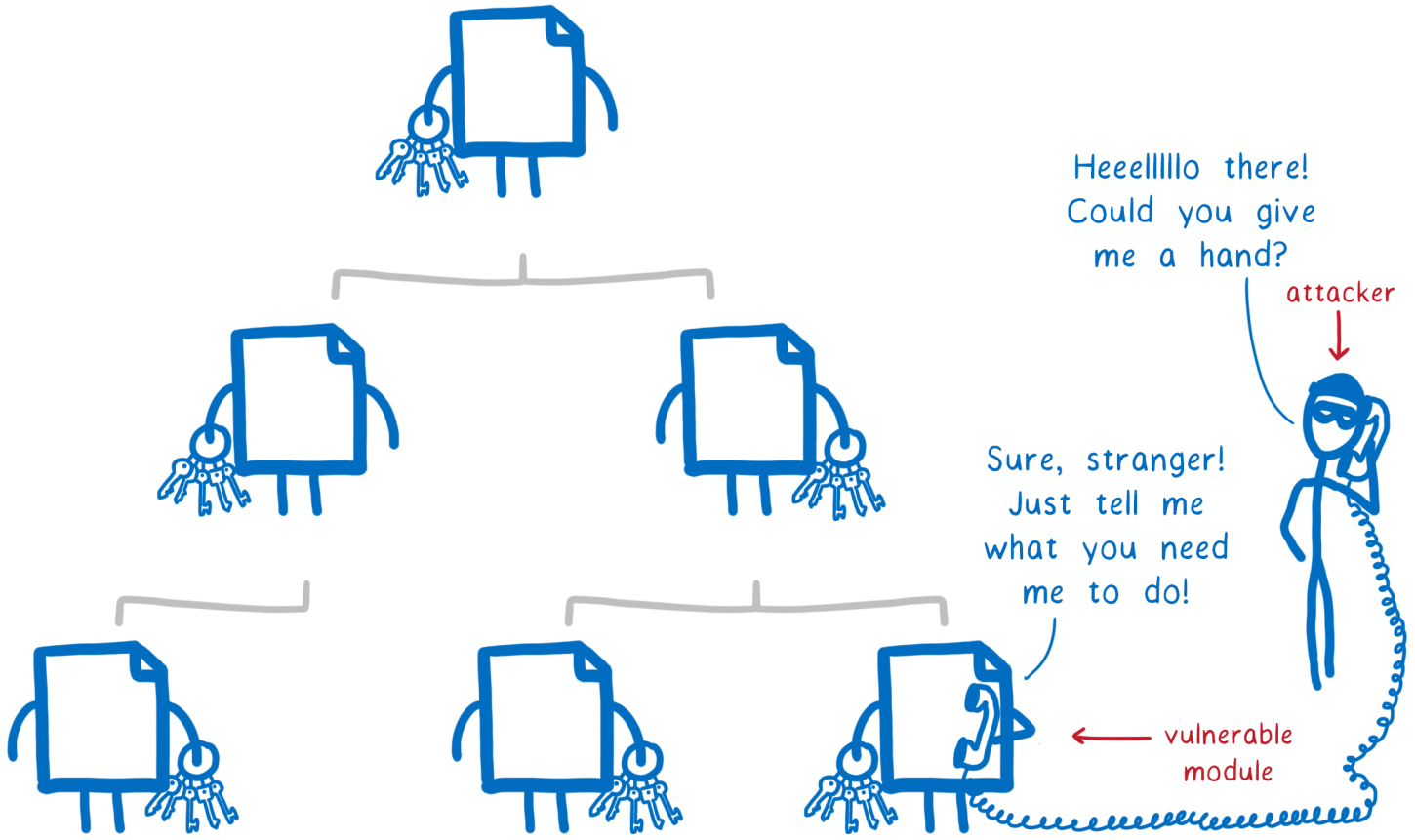
Dependencies



Malicious module

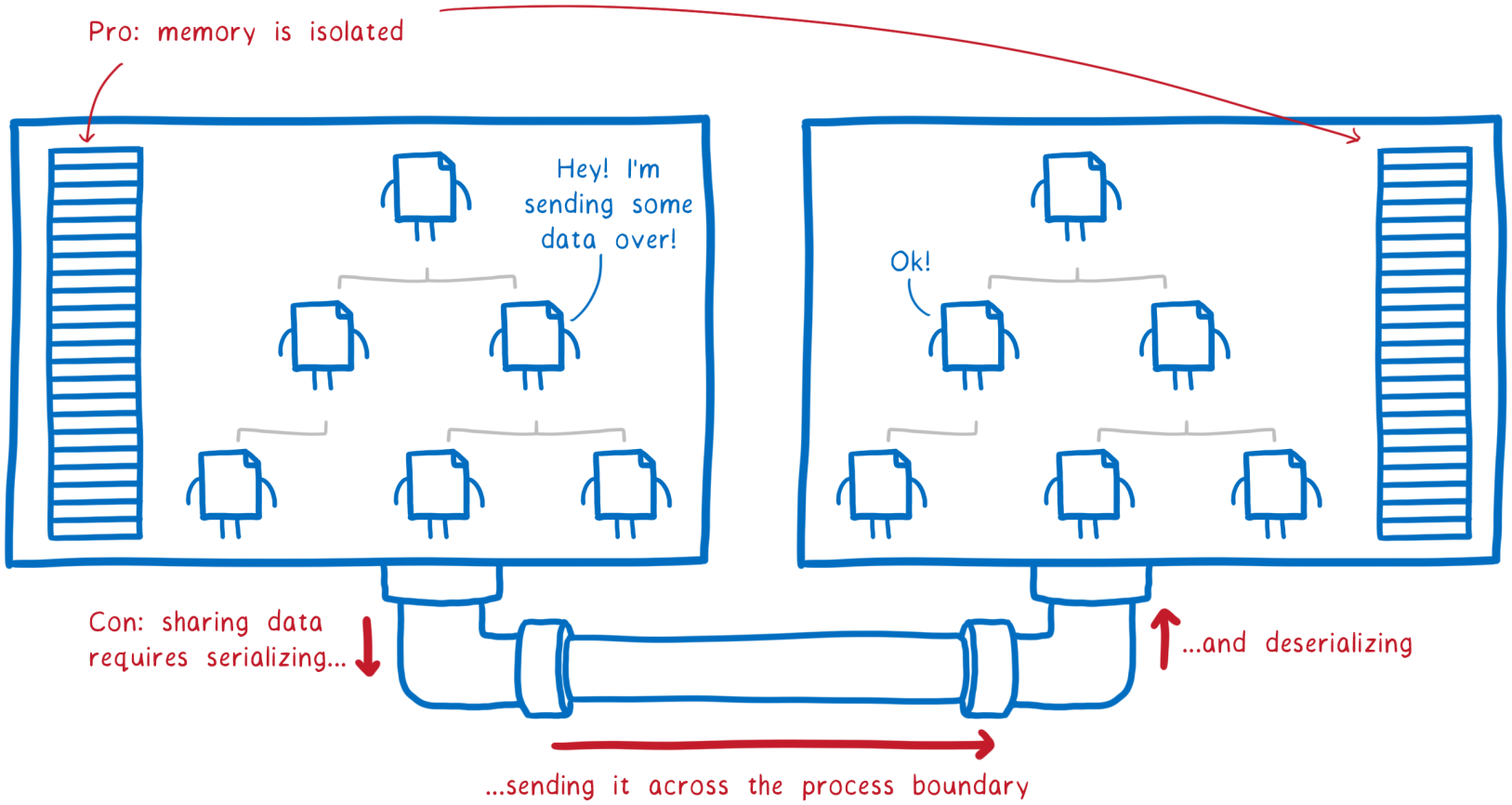


Vulnerable module

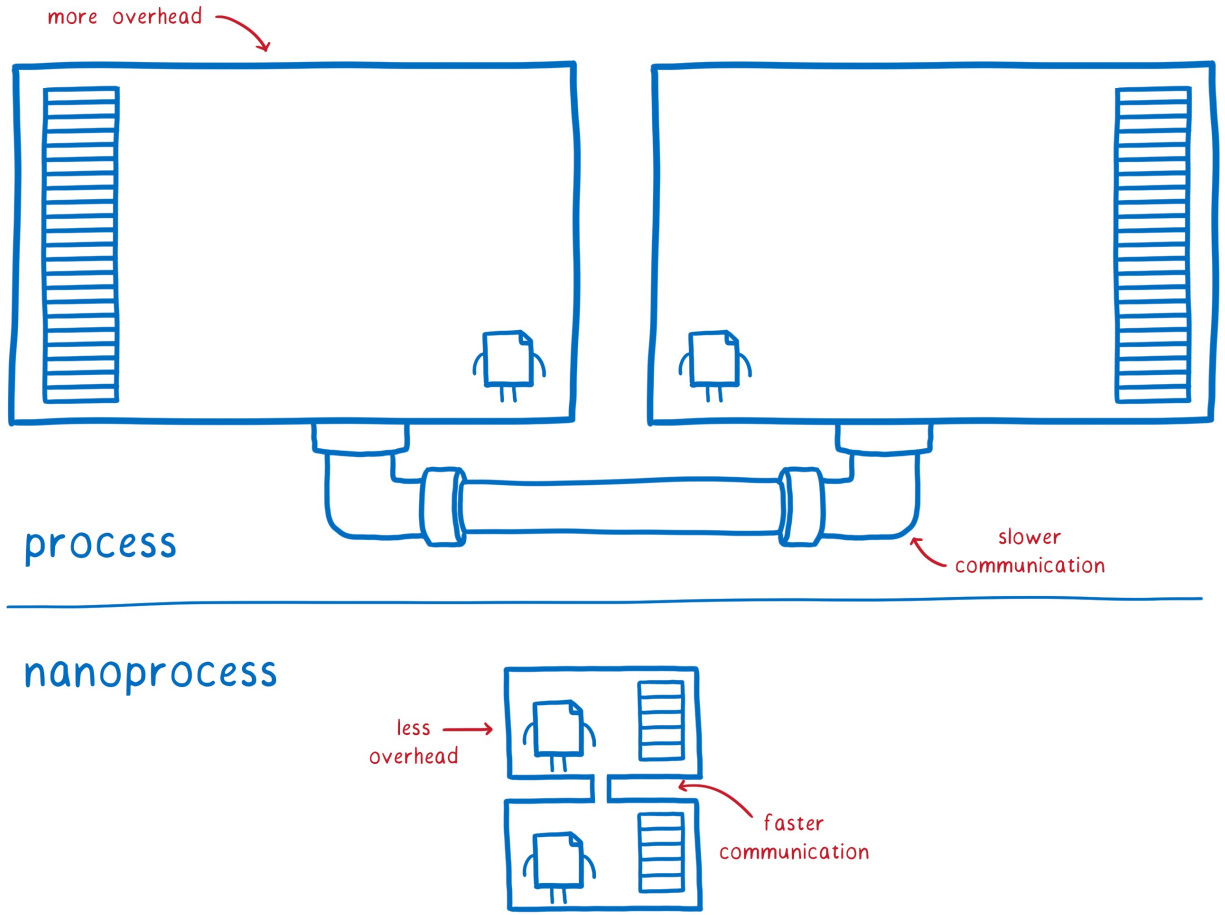


@nielstanis@infosec.exchange

Process Isolation

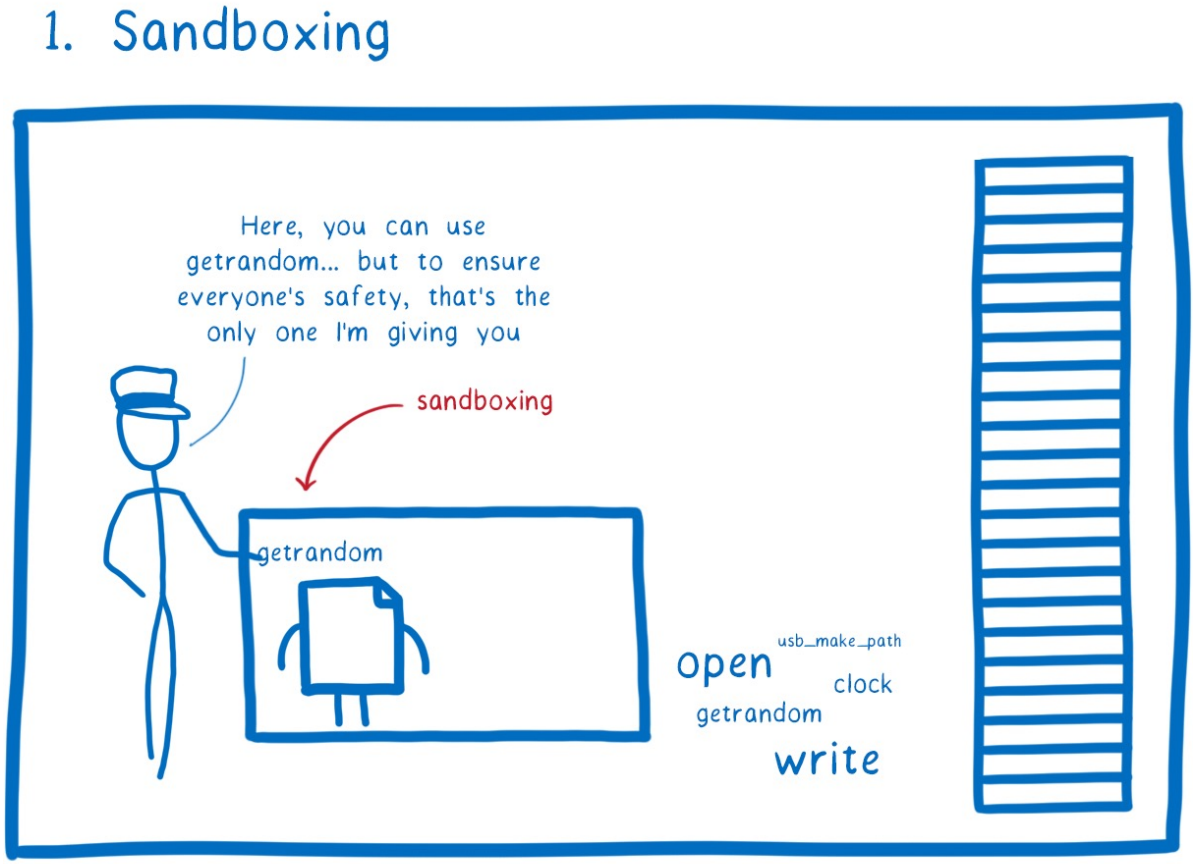


WebAssembly Nano-Process

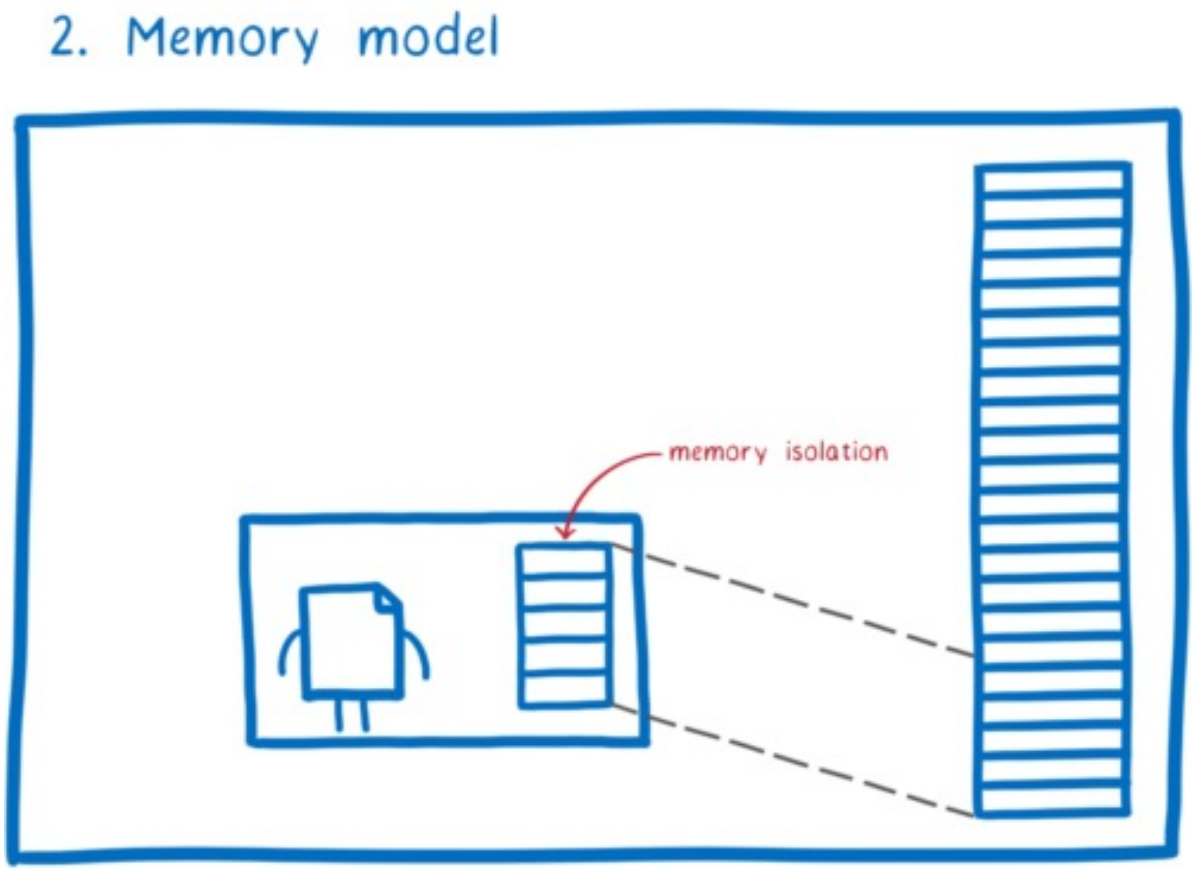


@nielstanis@infosec.exchange

WebAssembly Nano-Process

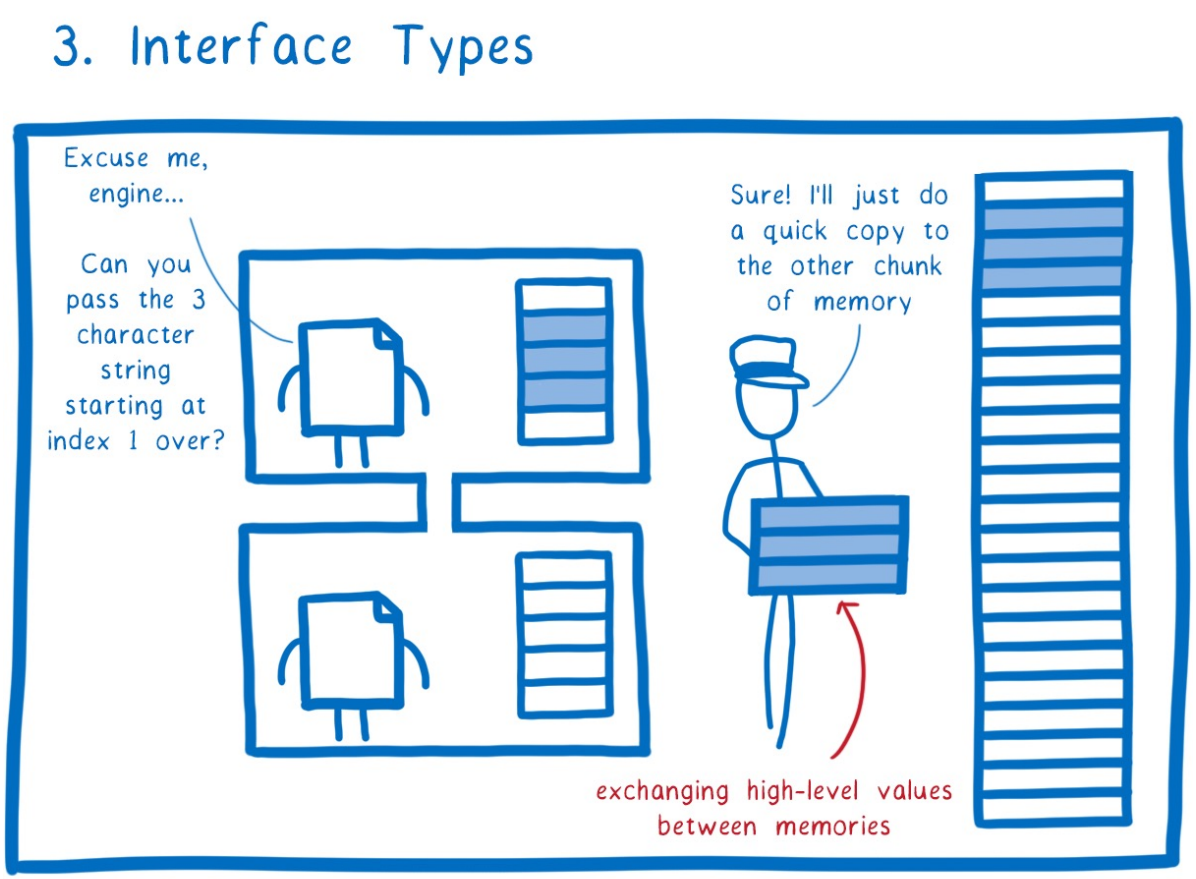


WebAssembly Nano-Process



@nielstanis@infosec.exchange

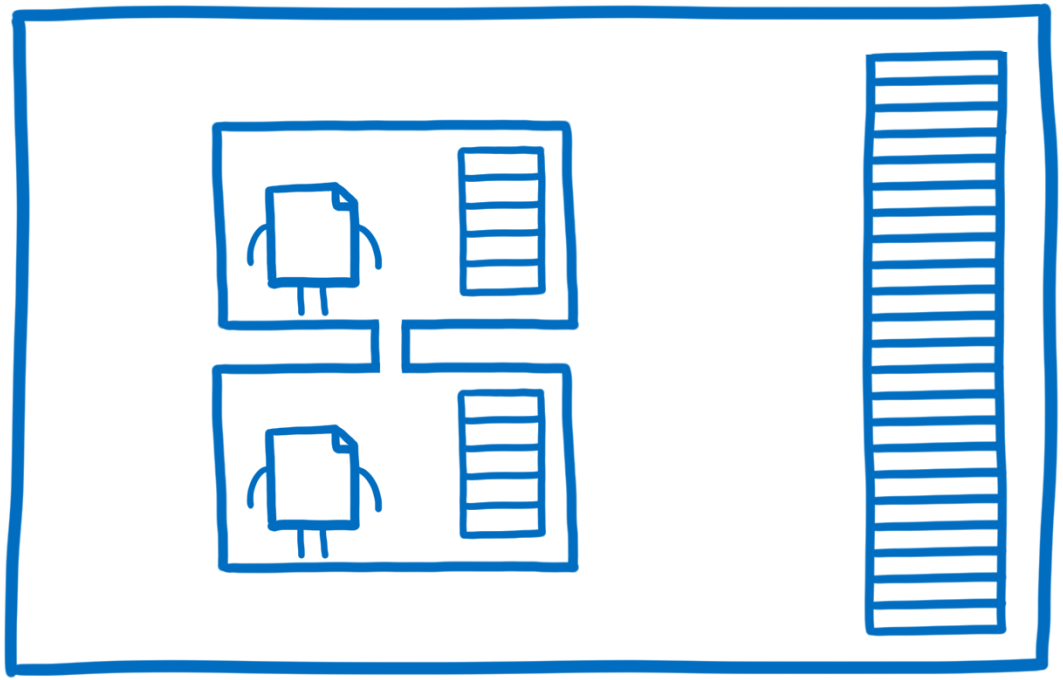
WebAssembly Nano-Process

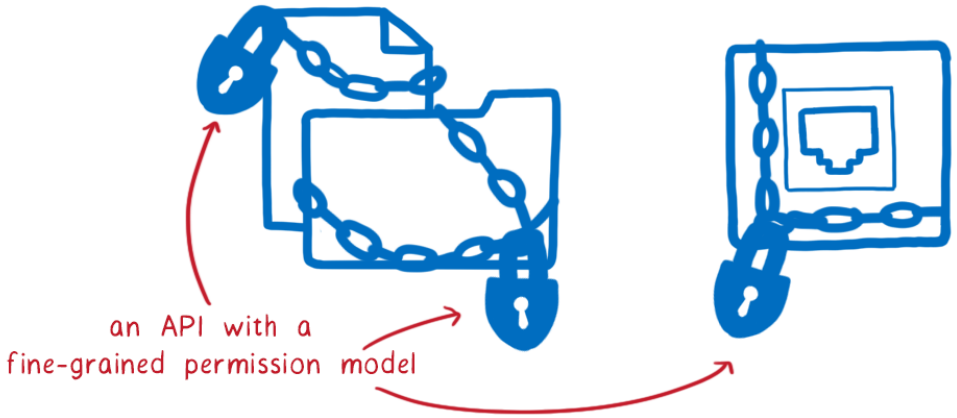


@nielstanis@infosec.exchange

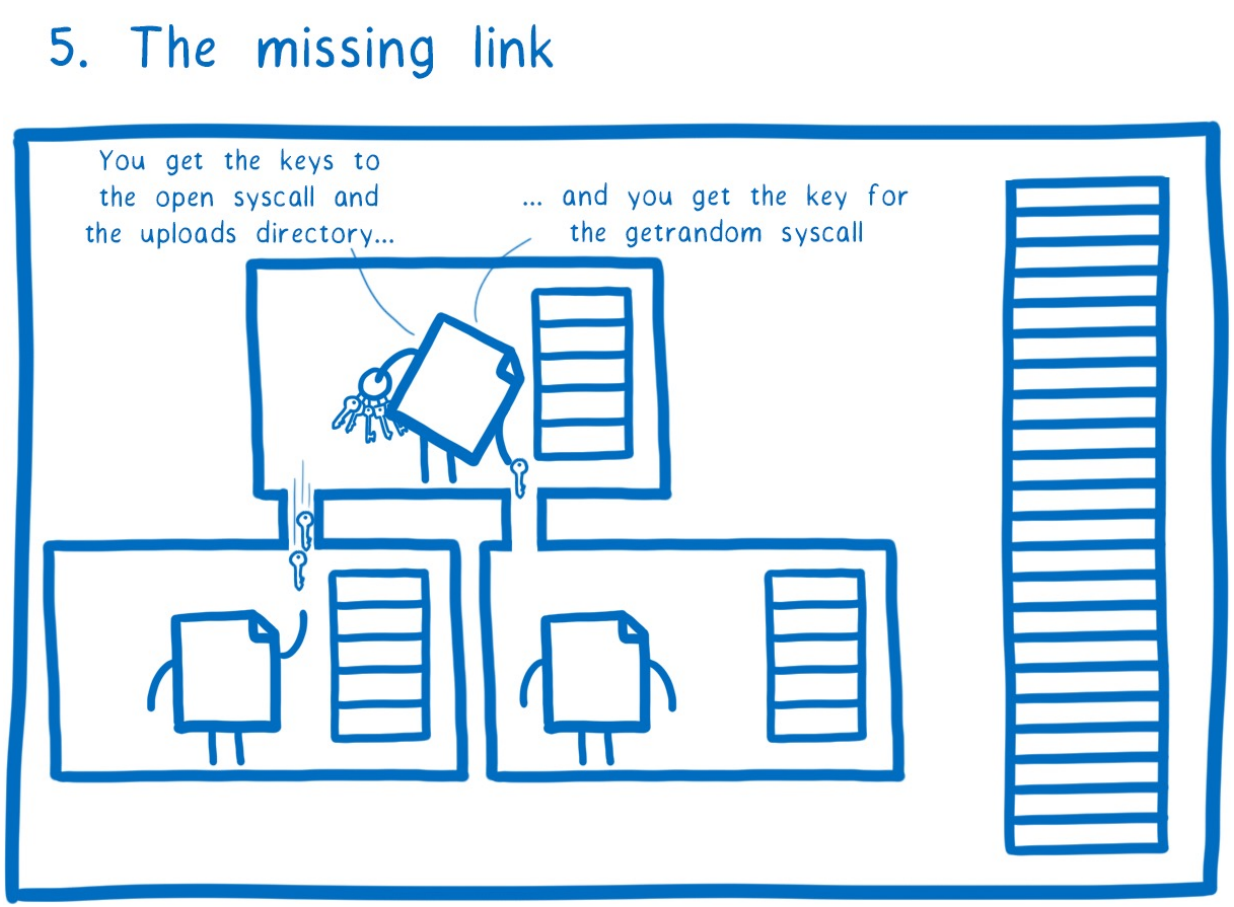
WebAssembly Nano-Process

4. WebAssembly System Interface



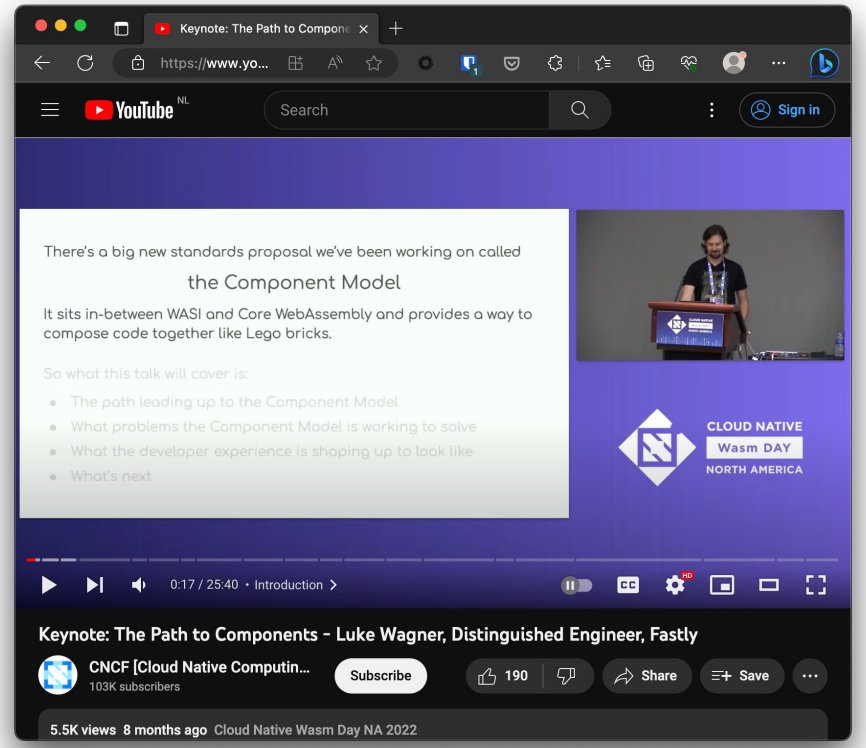


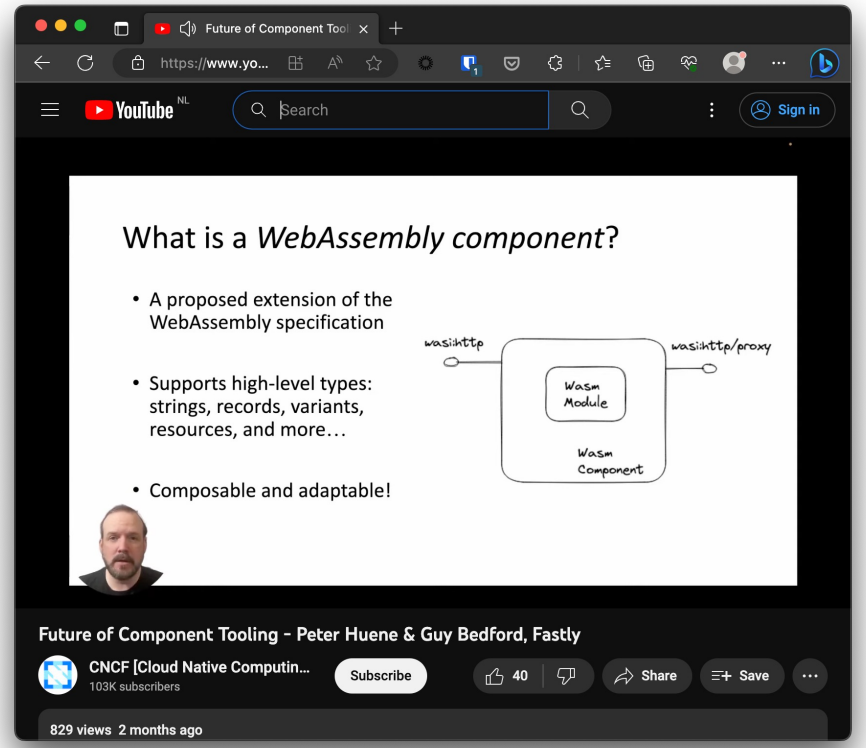
WebAssembly Nano-Process



@nielstanis@infosec.exchange

WebAssembly Component Model





@nielstanis@infosec.exchange

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/influence

Questions?

- https://github.com/nielstanis/wearedevs2023
- ntanis at Veracode.com
- @nielstanis@infosec.exchange
- https://blog.fennec.dev
- Danke! Thank you!
- Want to talk more about Veracode?
  Our booth is at A51!