SWETUGG
STOCKHOLM

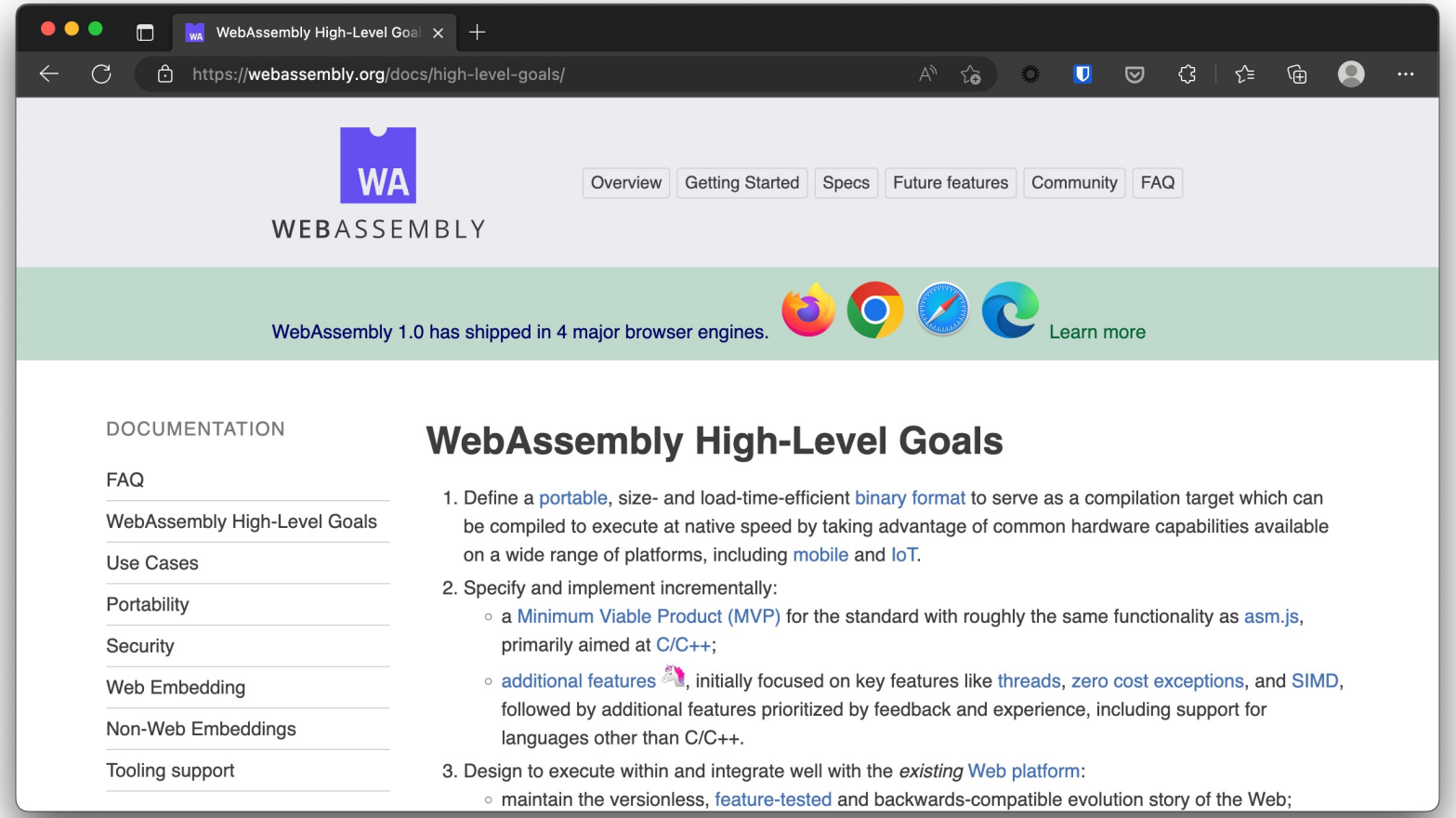
Using WebAssembly to run, extend, and secure your .NET application

Niels Tanis

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

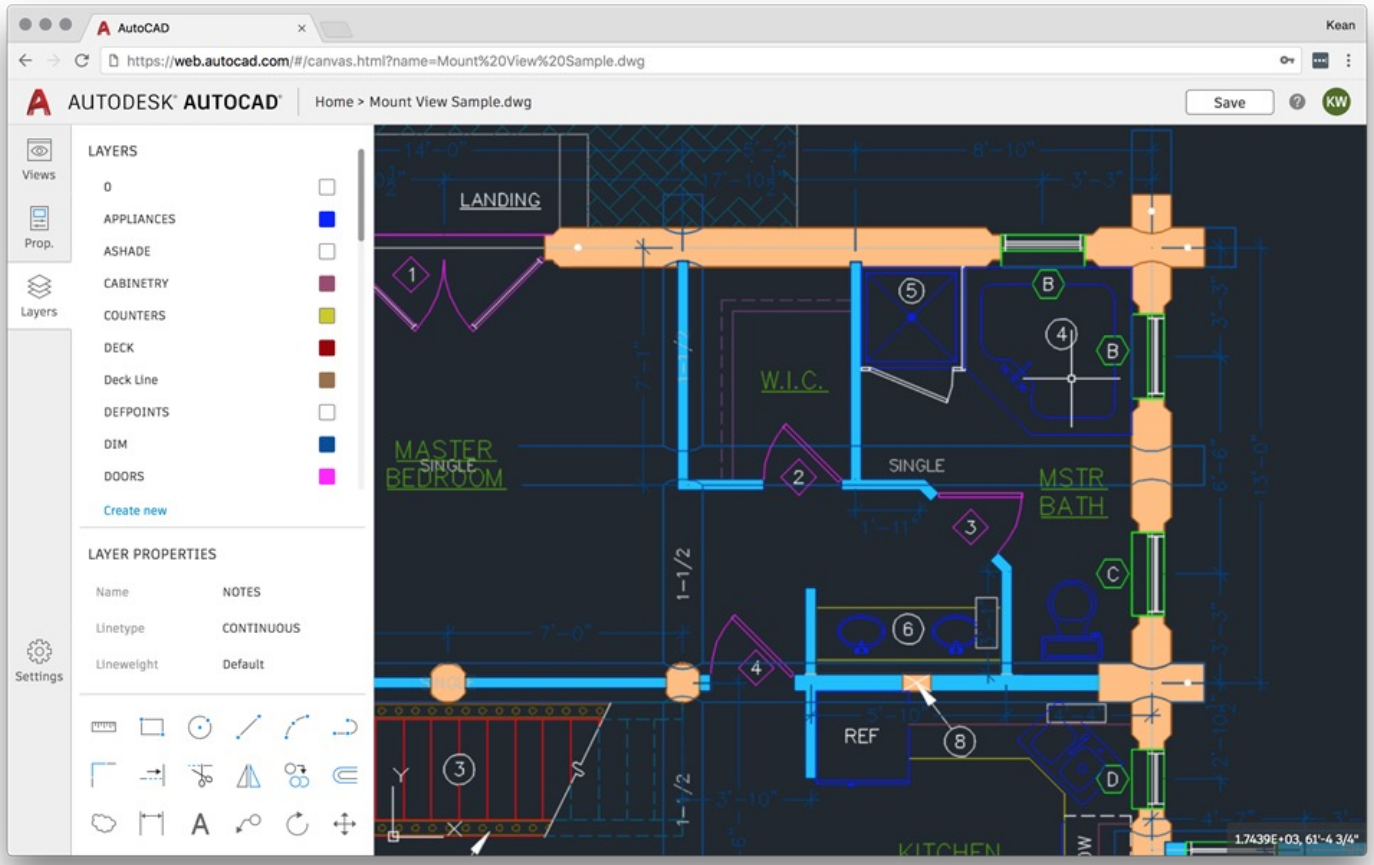
WebAssembly



SWETUGG

@nielstanis@infosec.exchange

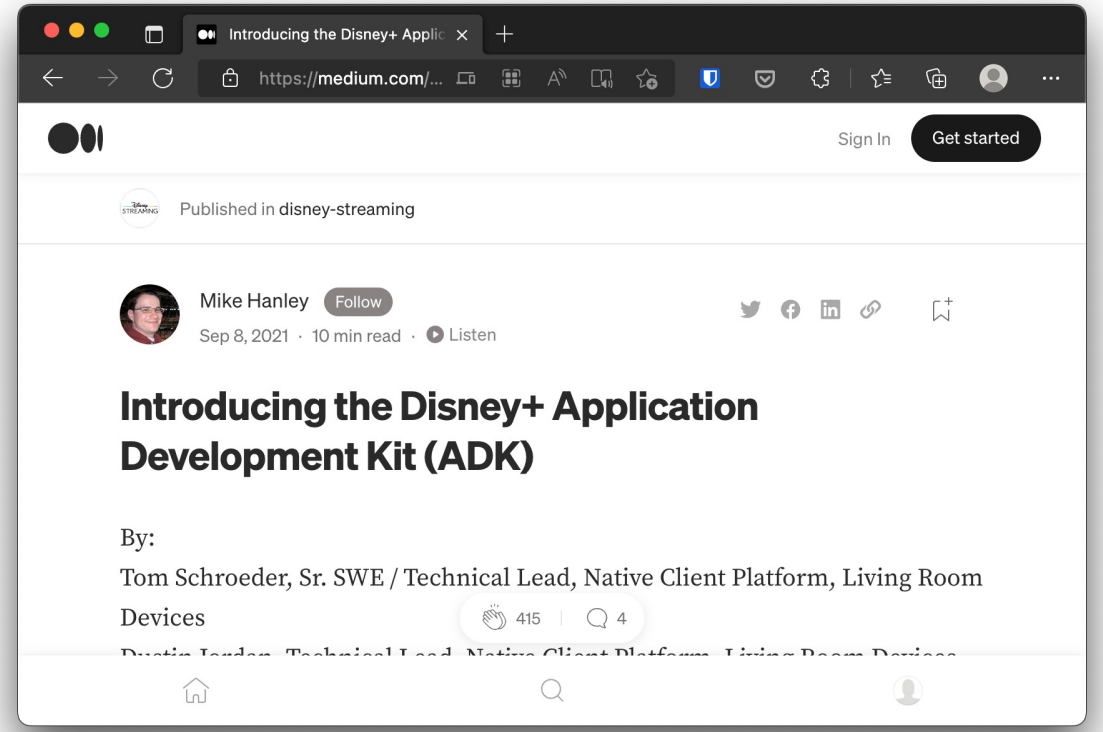
WebAssembly - AutoCAD

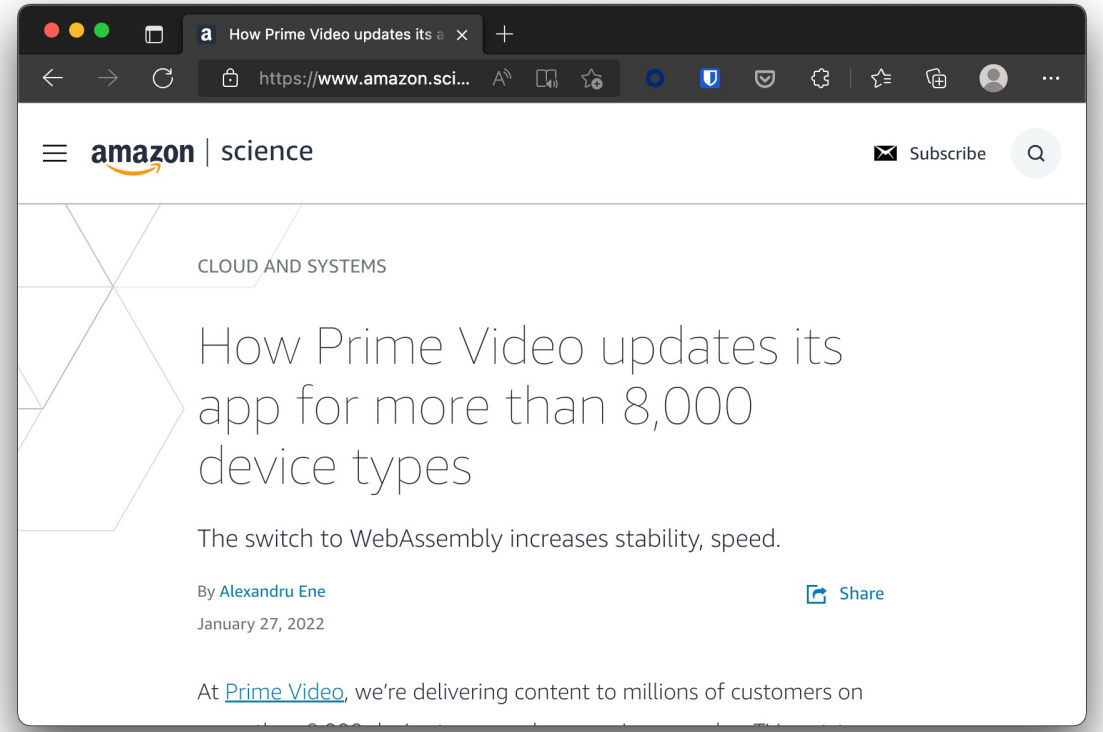


SWETUGG

@nielstanis@infosec.exchange

WebAssembly - SDK's





Agenda

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

WebAssembly Design

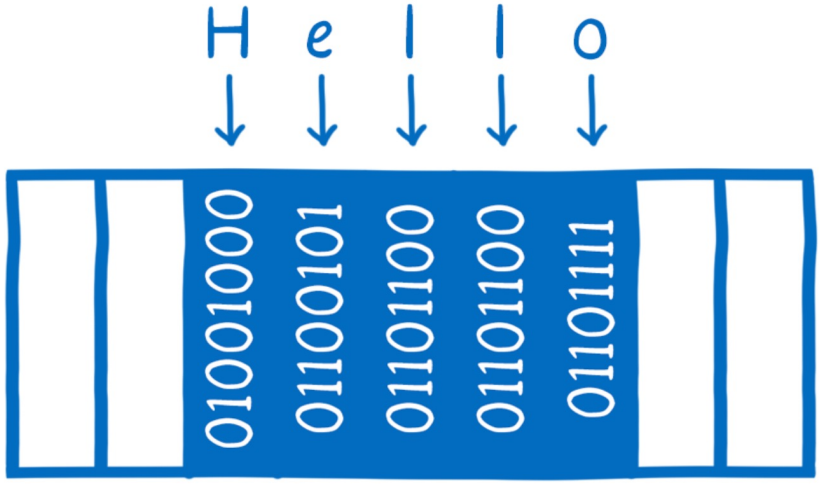
- **Be fast, efficient, and portable**
  - Executed in near-native speed across different platforms
- **Be readable and debuggable**
  - In low-level bytecode but also human readable
- **Keep secure**
  - Run on sandboxed execution environment
- **Don't break the web**
  - Ensure backwards compatibility

WebAssembly

- Binary instruction format for stack-based virtual machine similar to .NET CLR running MSIL or JVM running bytecode
- 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



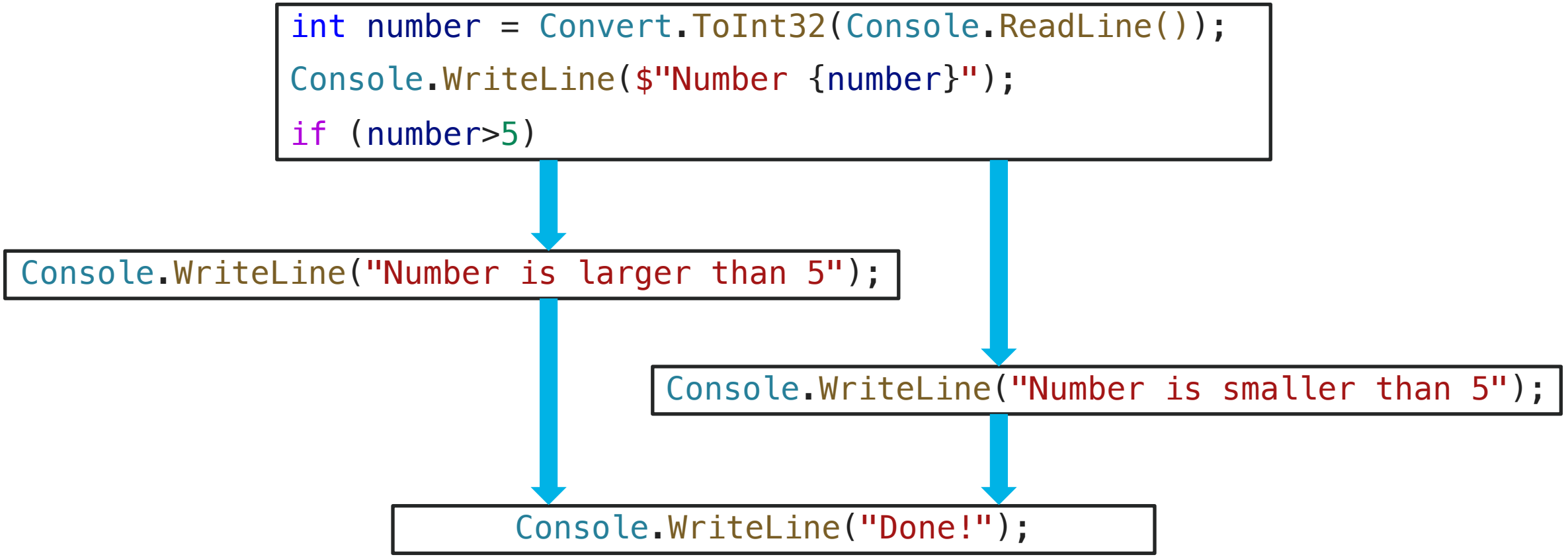
WebAssembly Control-Flow Integrity

```
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!");
```

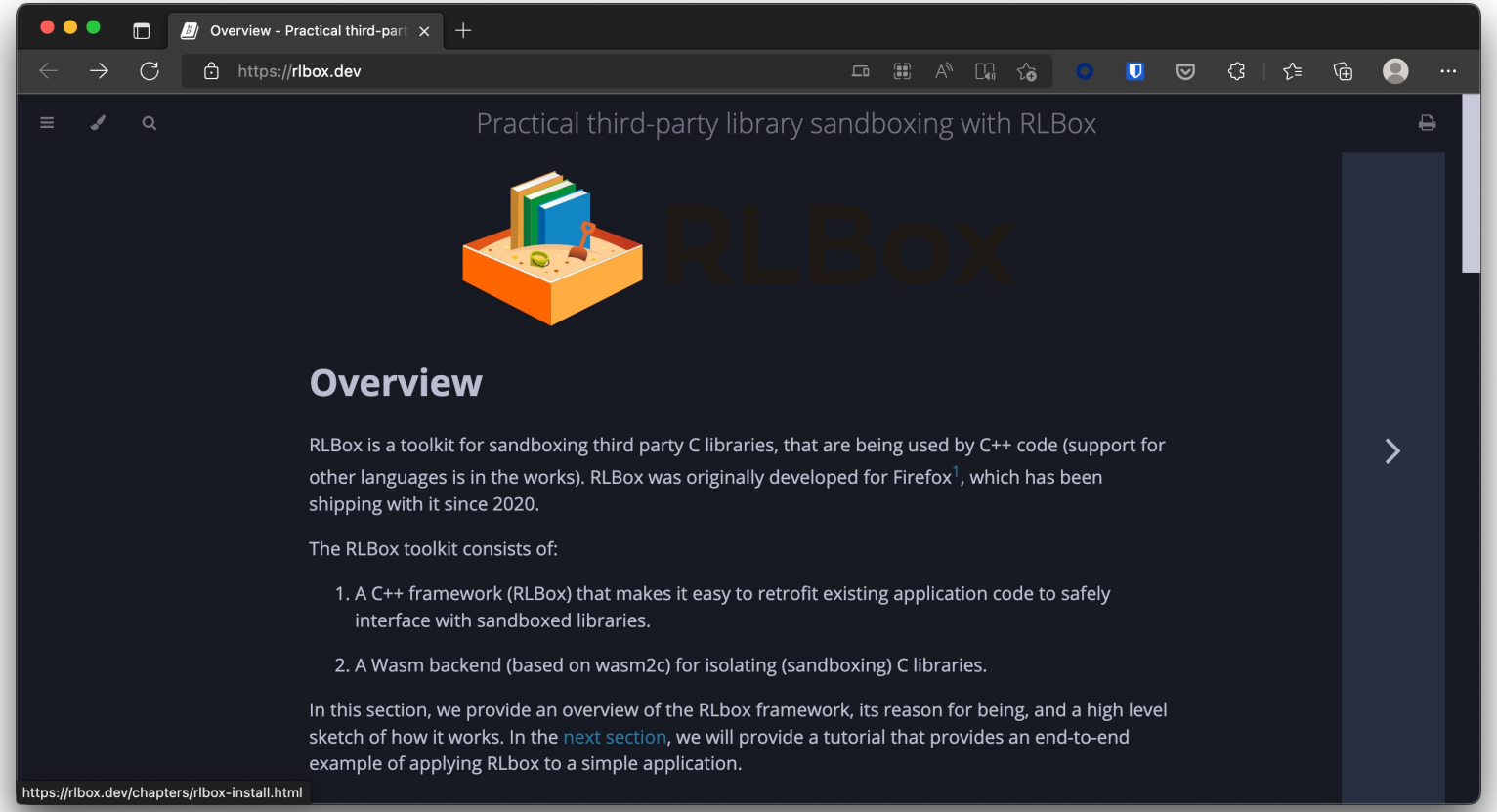
SWETUGG

@nielstanis@infosec.exchange

WebAssembly Control-Flow Integrity



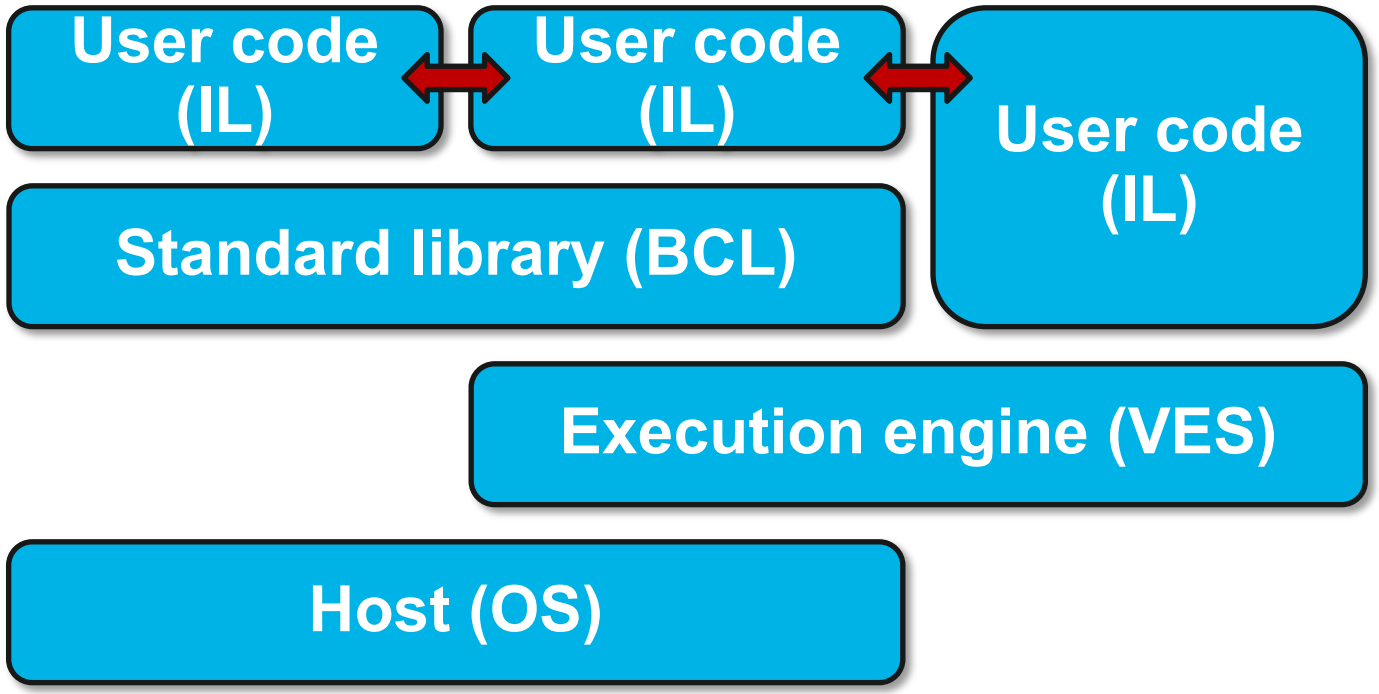
FireFox RLBox



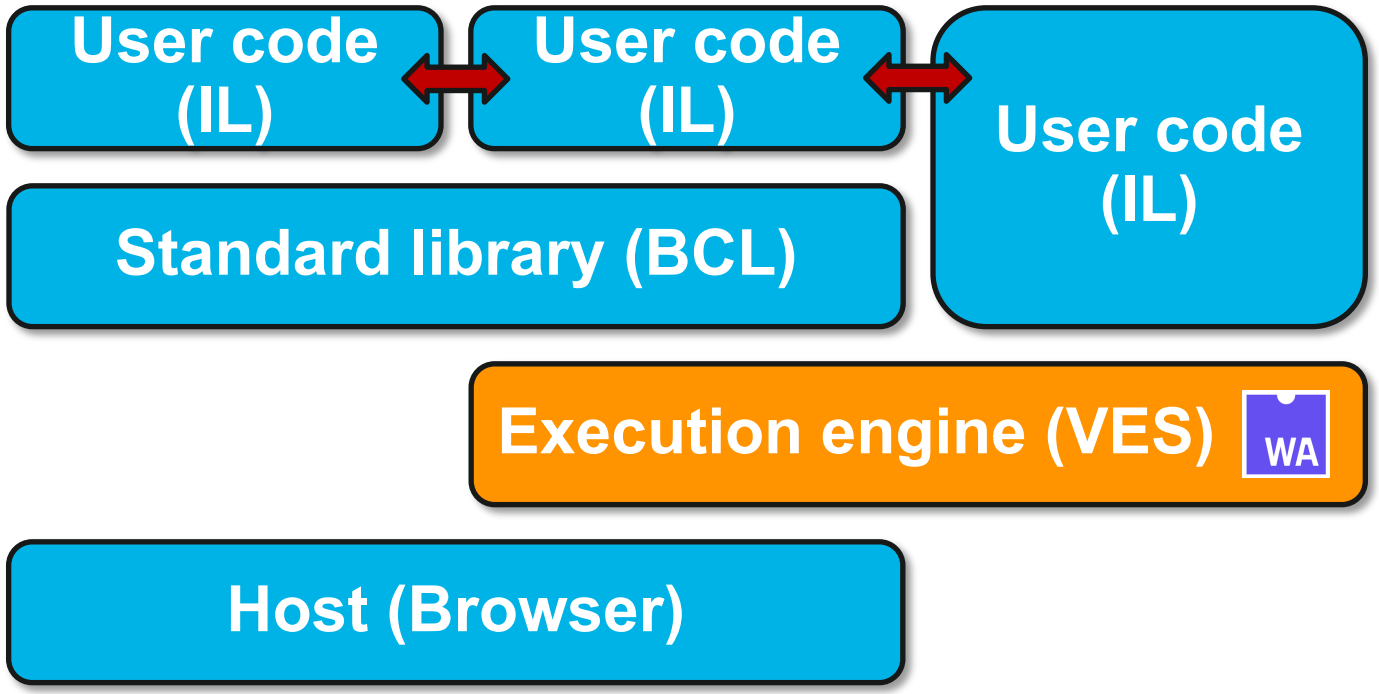
SWETUGG

@nielstanis@infosec.exchange

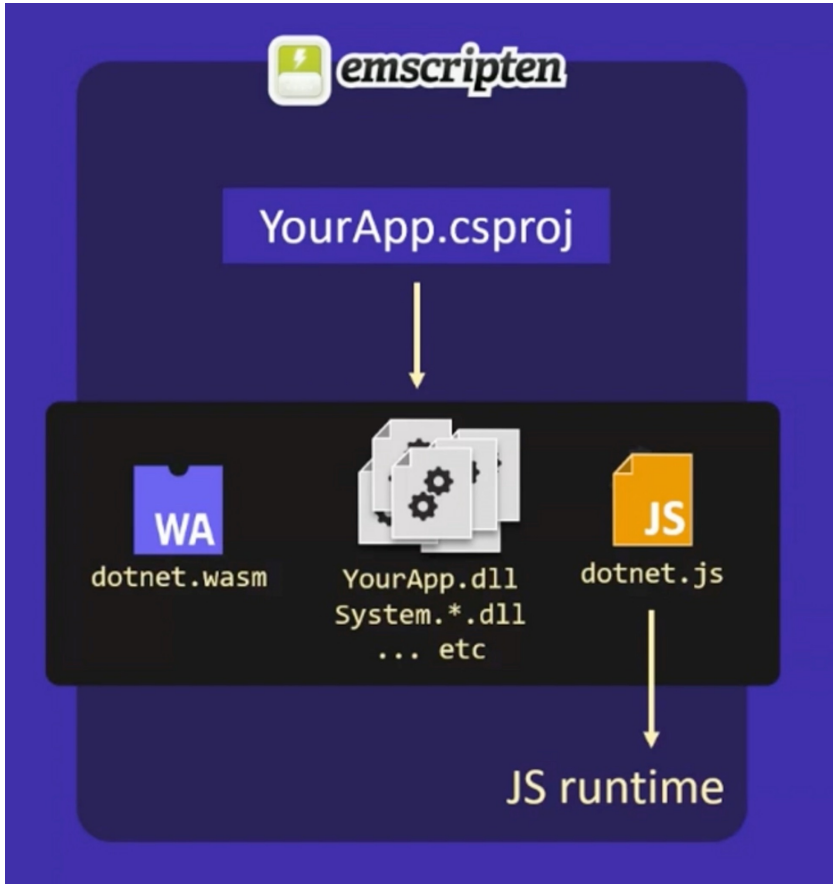
Running .NET on WebAssembly



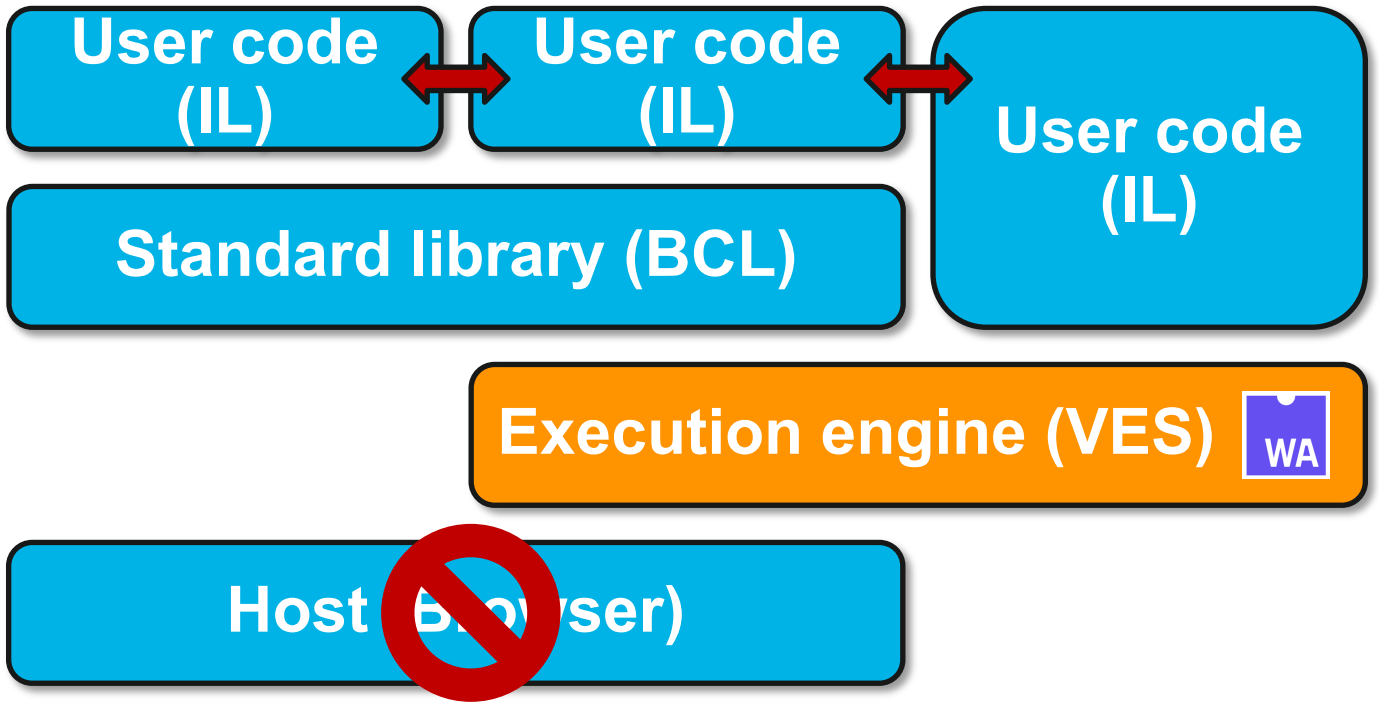
Running .NET on WebAssembly



Blazor WebAssembly



Running .NET on WebAssembly



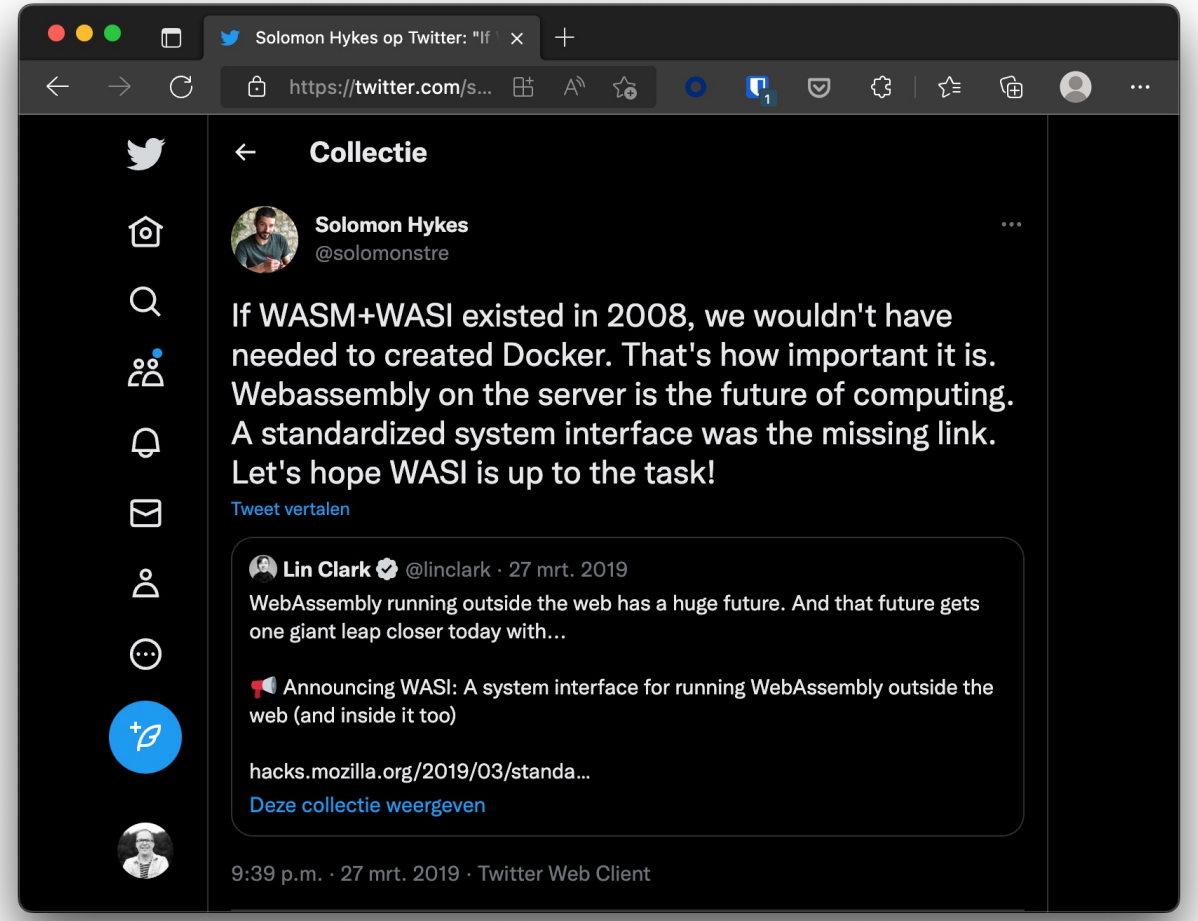
WebAssembly System Interface WASI

- 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



SWETUGG

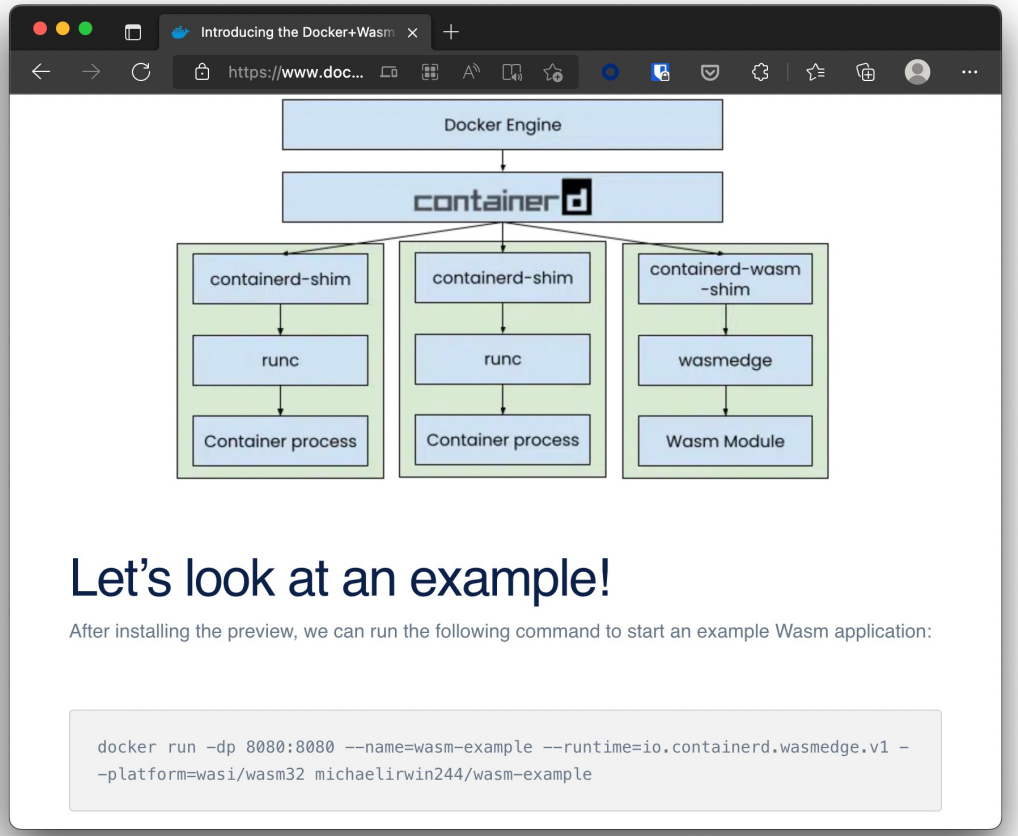
@nielstanis@infosec.exchange

Docker vs WASM & WASI

SWETUGG

@nielstanis@infosec.exchange

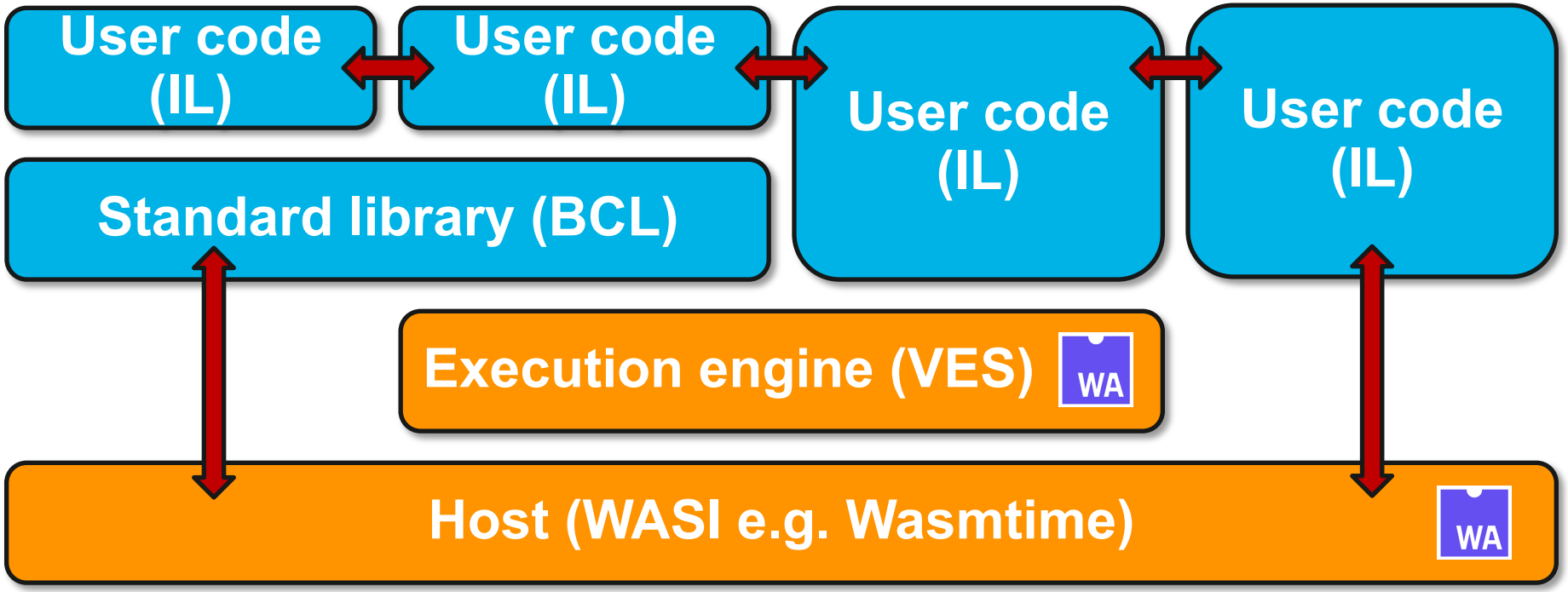
Docker & WASM



SWETUGG

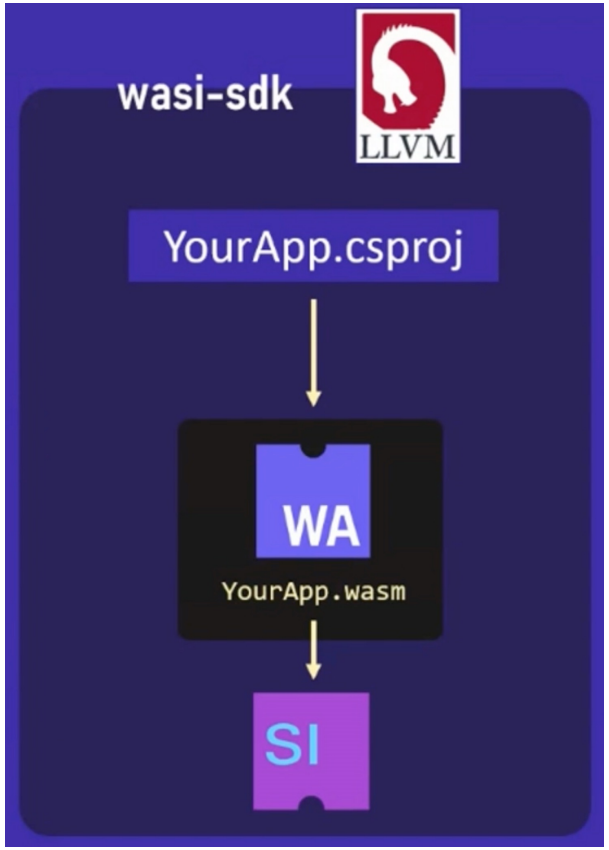
@nielstanis@infosec.exchange

WebAssembly System Interface WASI

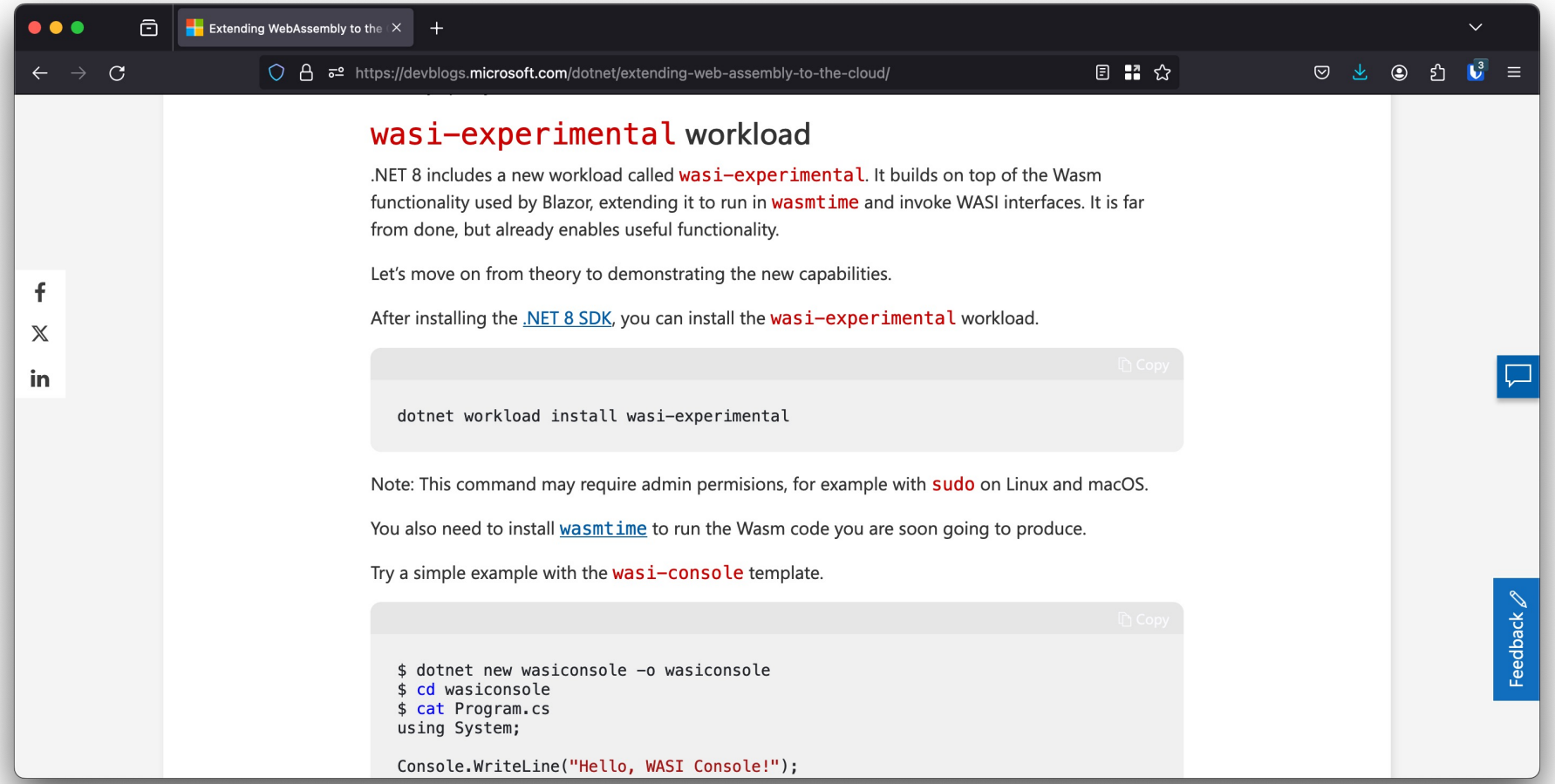


@nielstanis@infosec.exchange

Experimental WASI SDK for .NET



.NET 8 WASI-Experimental



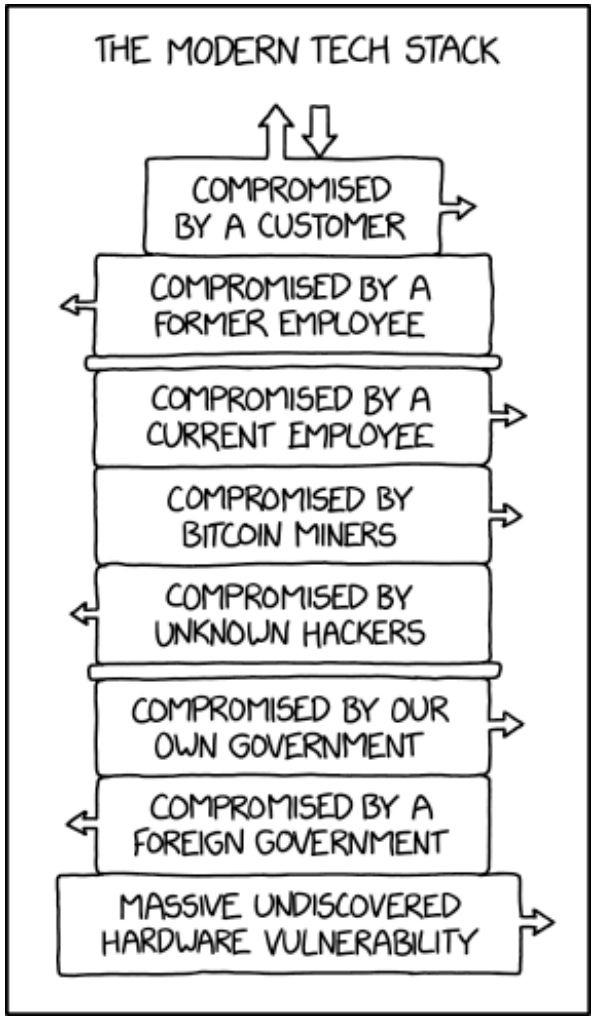
SWETUGG

@nielstanis@infosec.exchange

Extending .NET with WASM

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

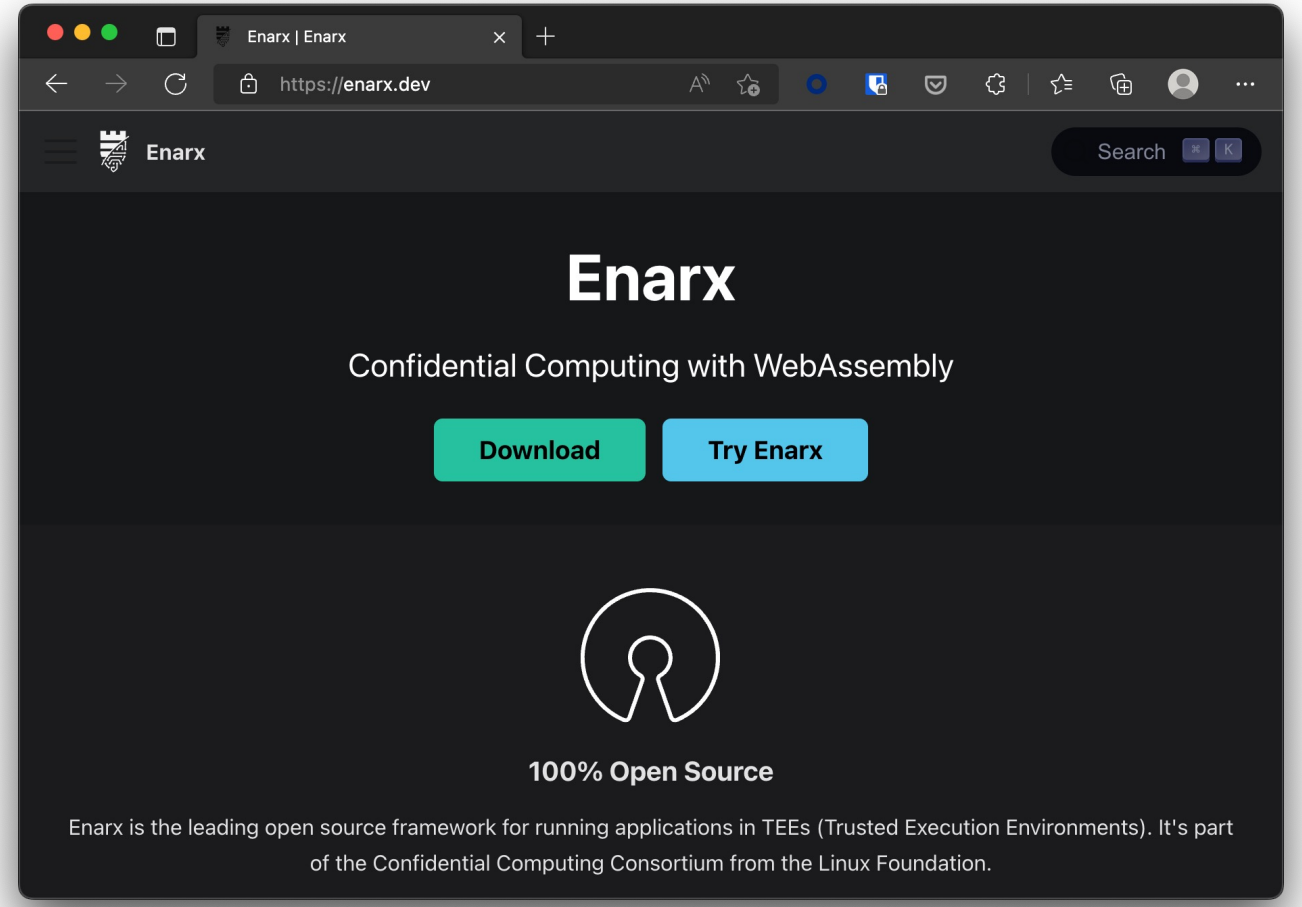
Trusted Computing - XKCD 2166



SWETUGG

@nielstanis@infosec.exchange

Enarx

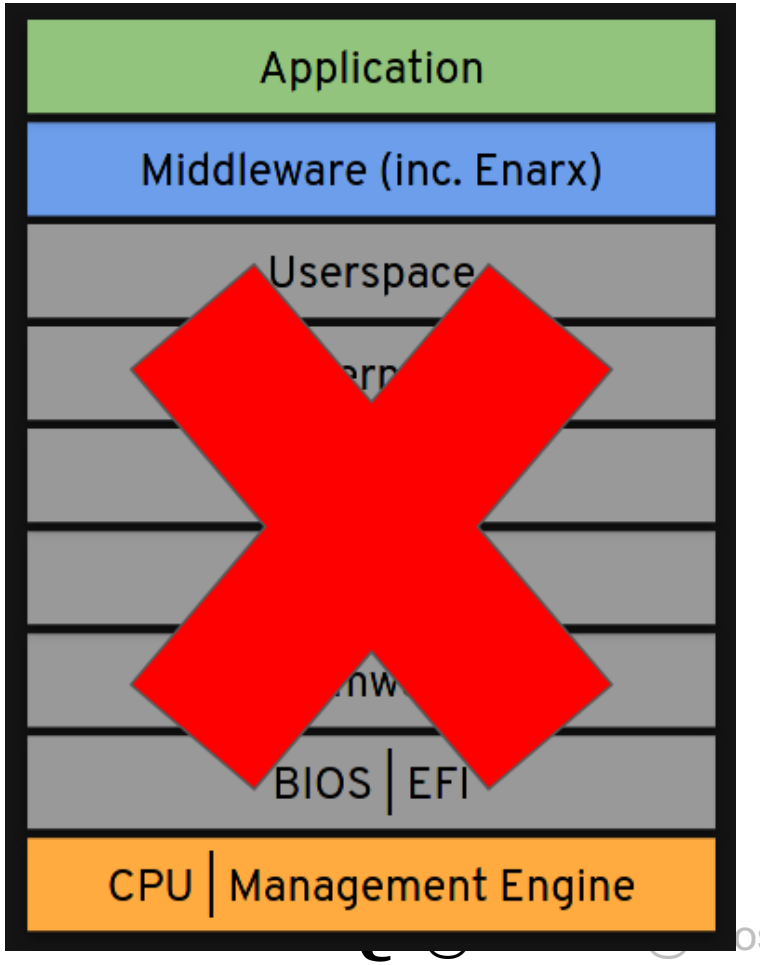


SWETUGG

@nielstanis@infosec.exchange

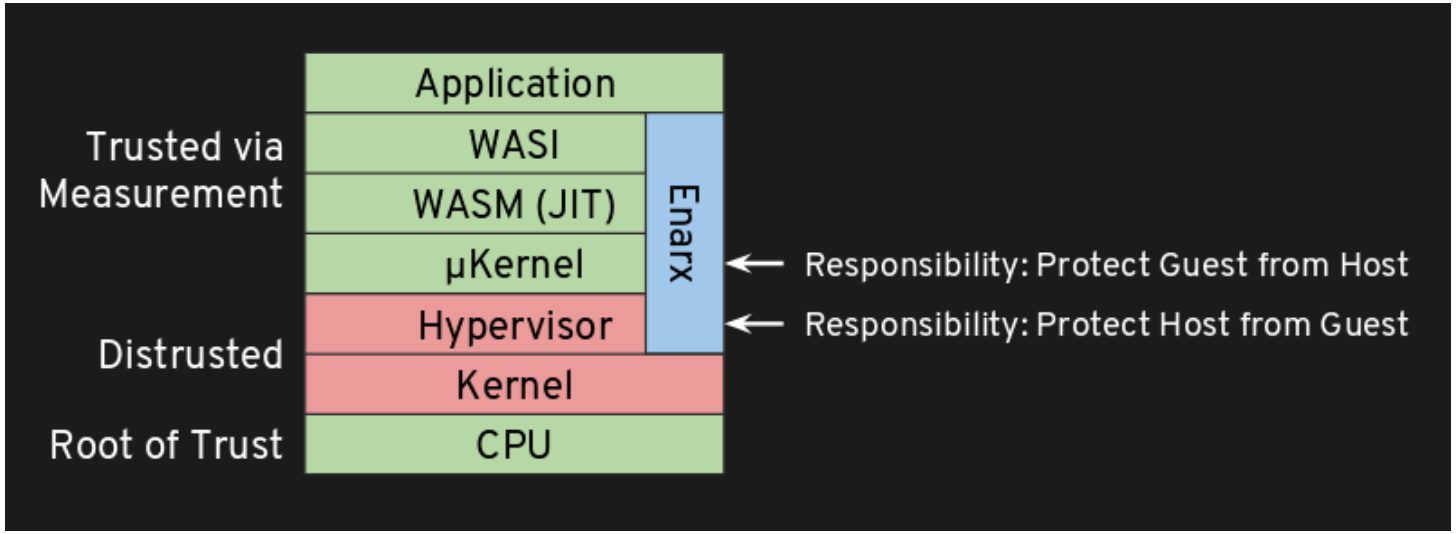
Enarx Threat Model

- 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



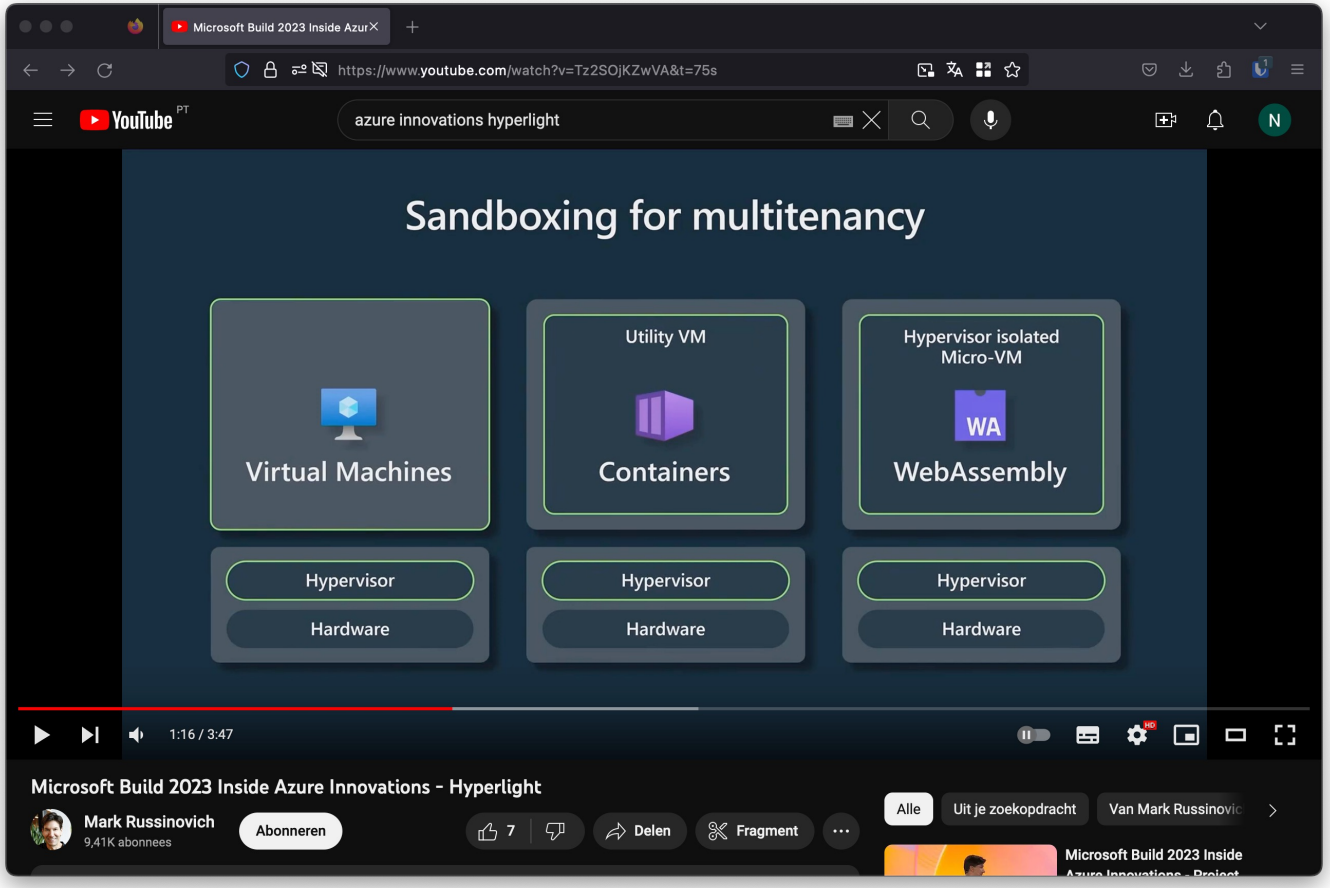
Enarx

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



@nielstanis@infosec.exchange

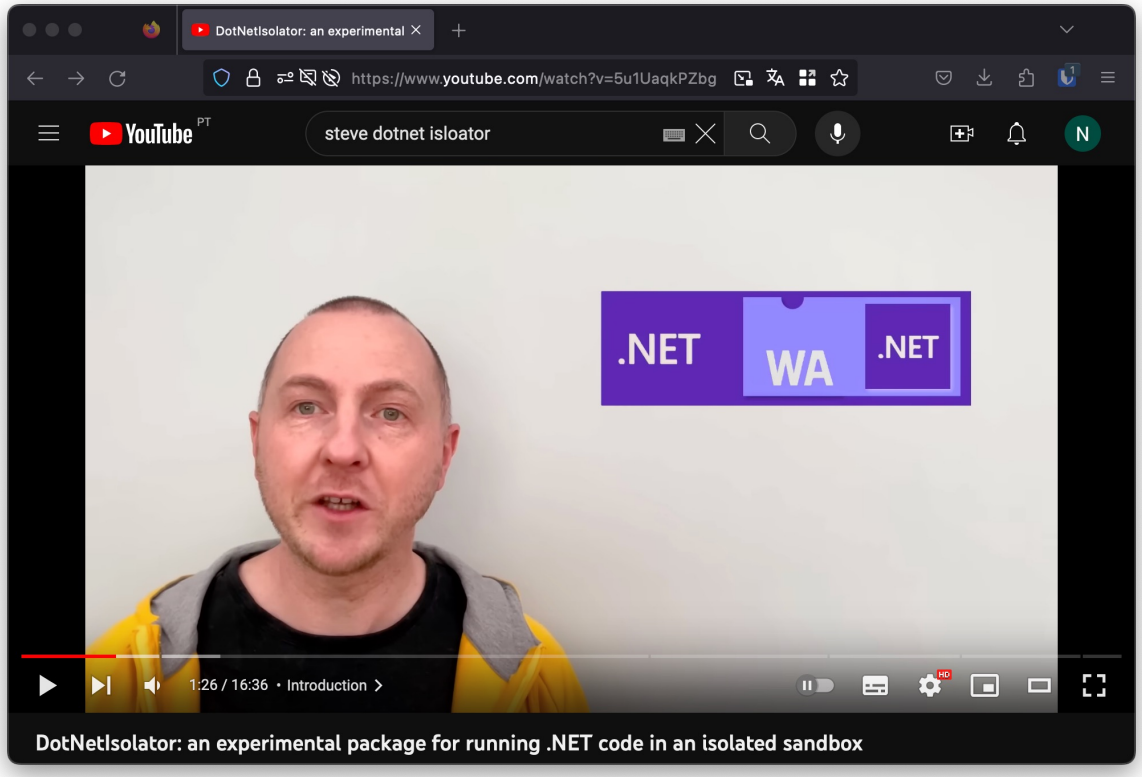
Project Hyperlight



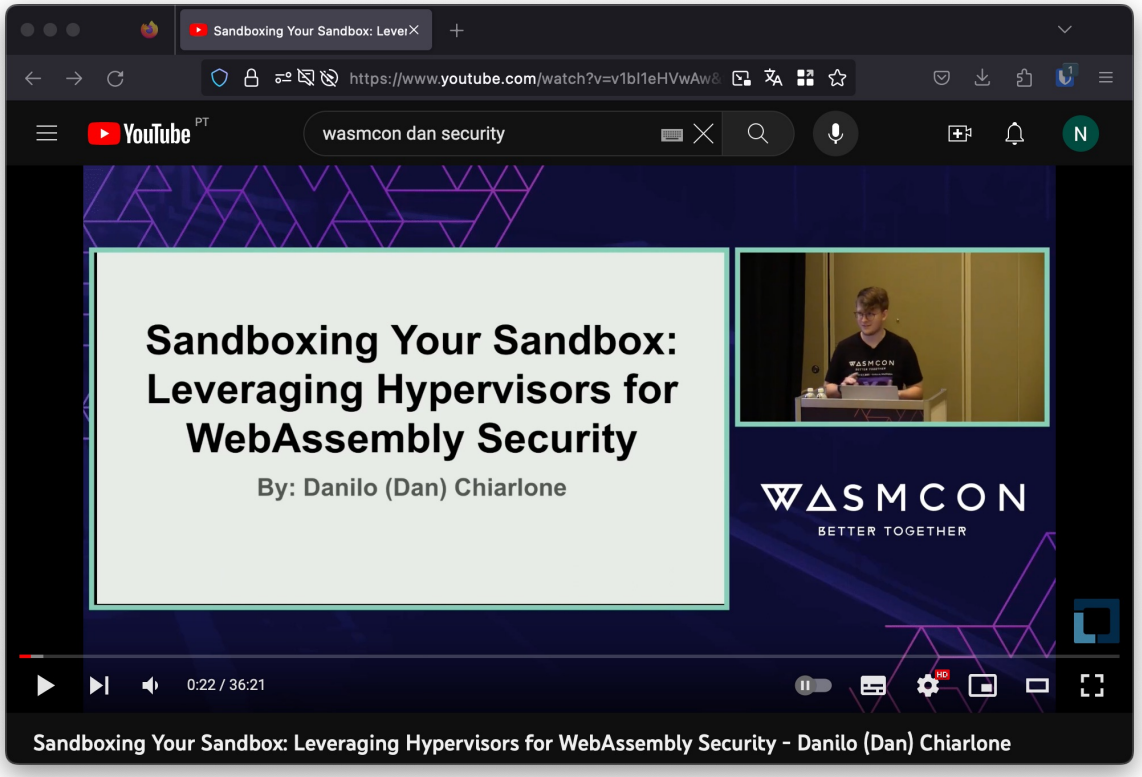
SWETUGG

@nielstanis@infosec.exchange

DotNetIsolator & Project Hyperlight



DotNetIsolator: an experimental package for running .NET code in an isolated sandbox



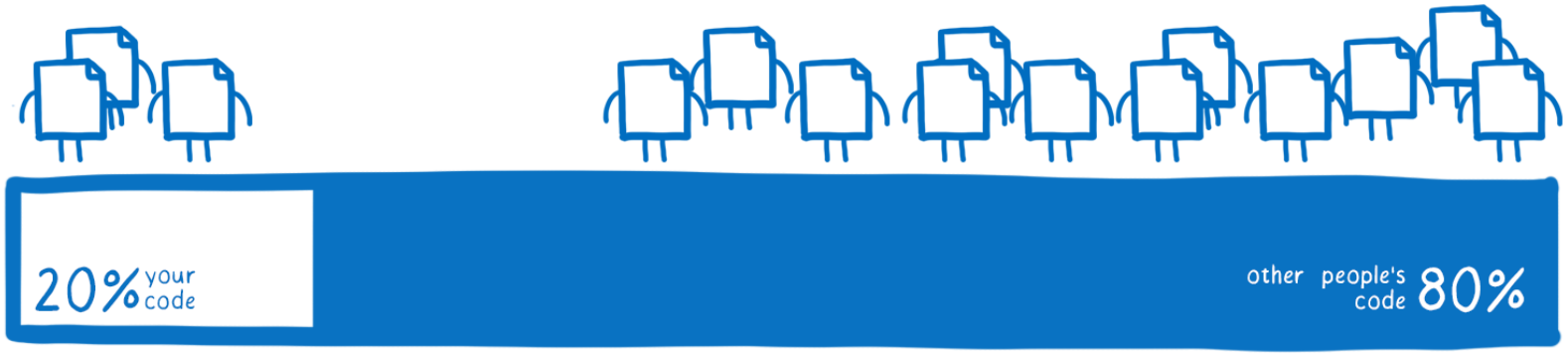
Sandboxing Your Sandbox: Leveraging Hypervisors for WebAssembly Security - Danilo (Dan) Chiarlone

SWETUGG

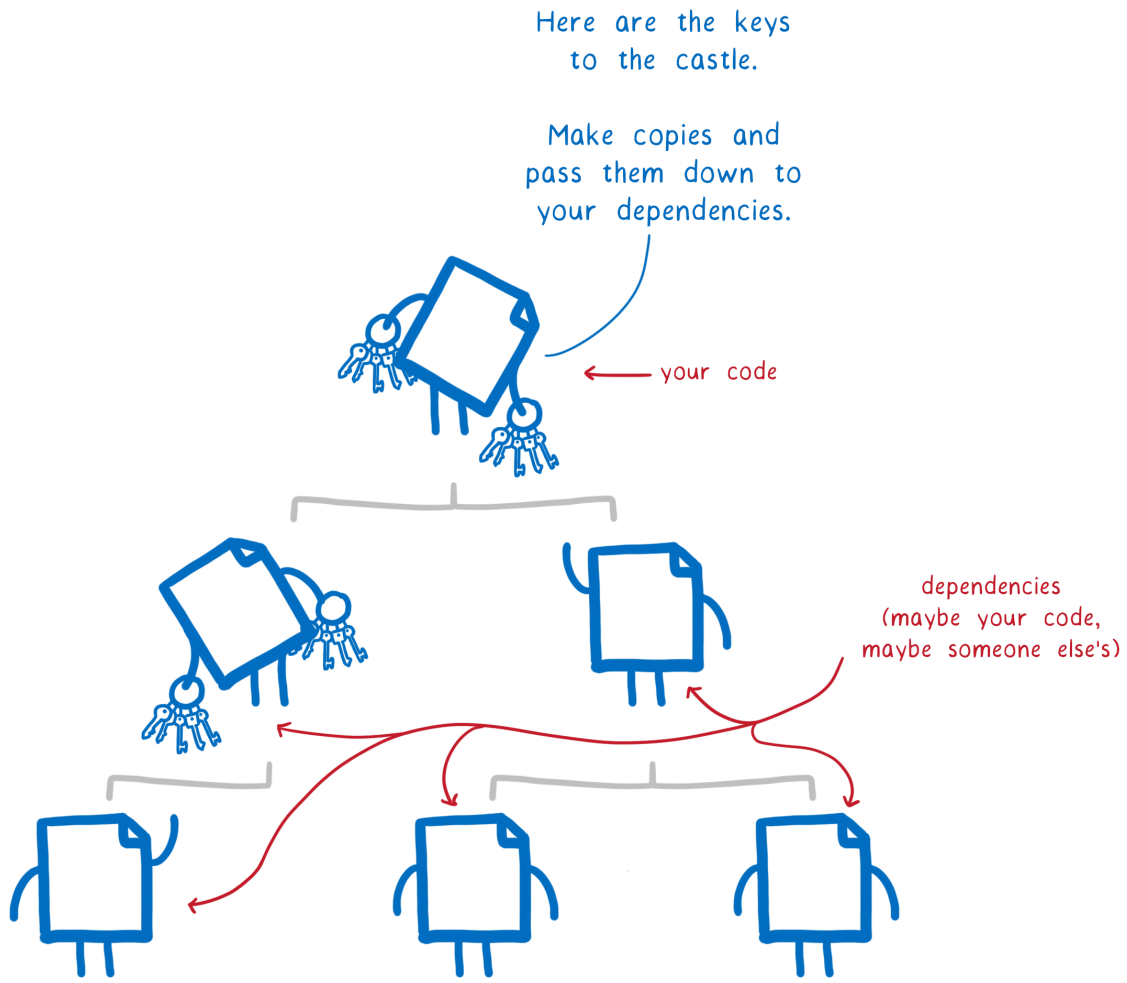
@nielstanis@infosec.exchange

WASM – What's next?

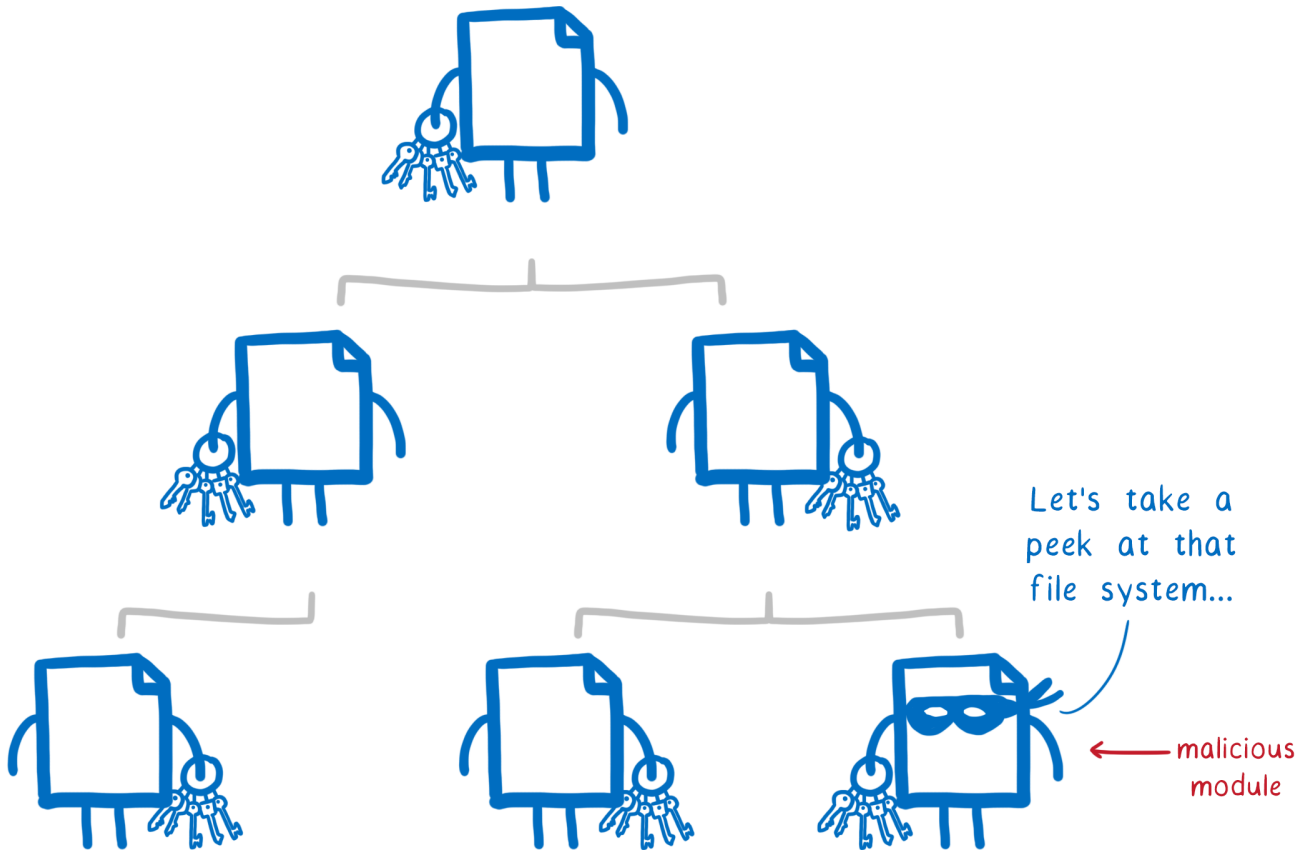
composition of an
average code base



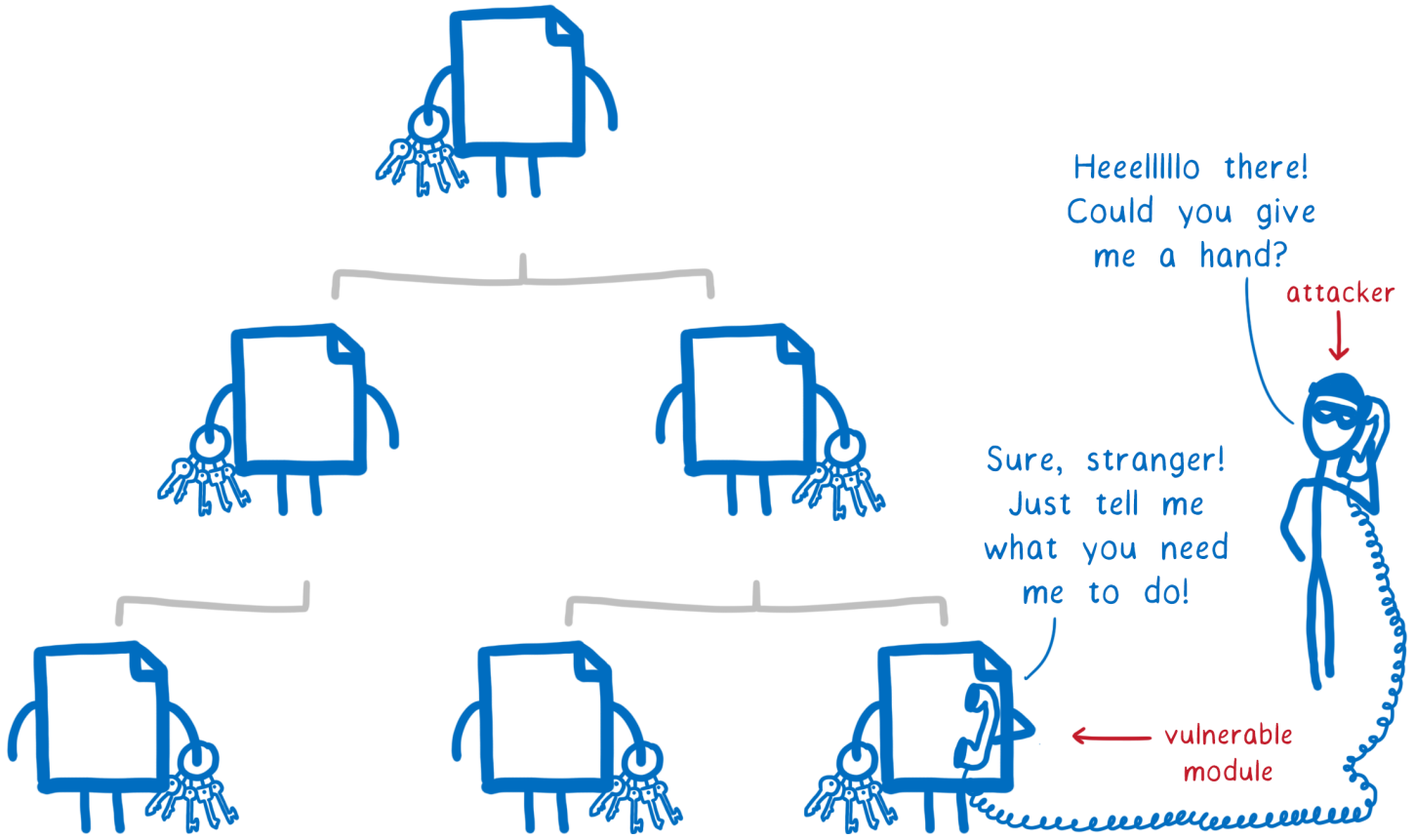
Dependencies



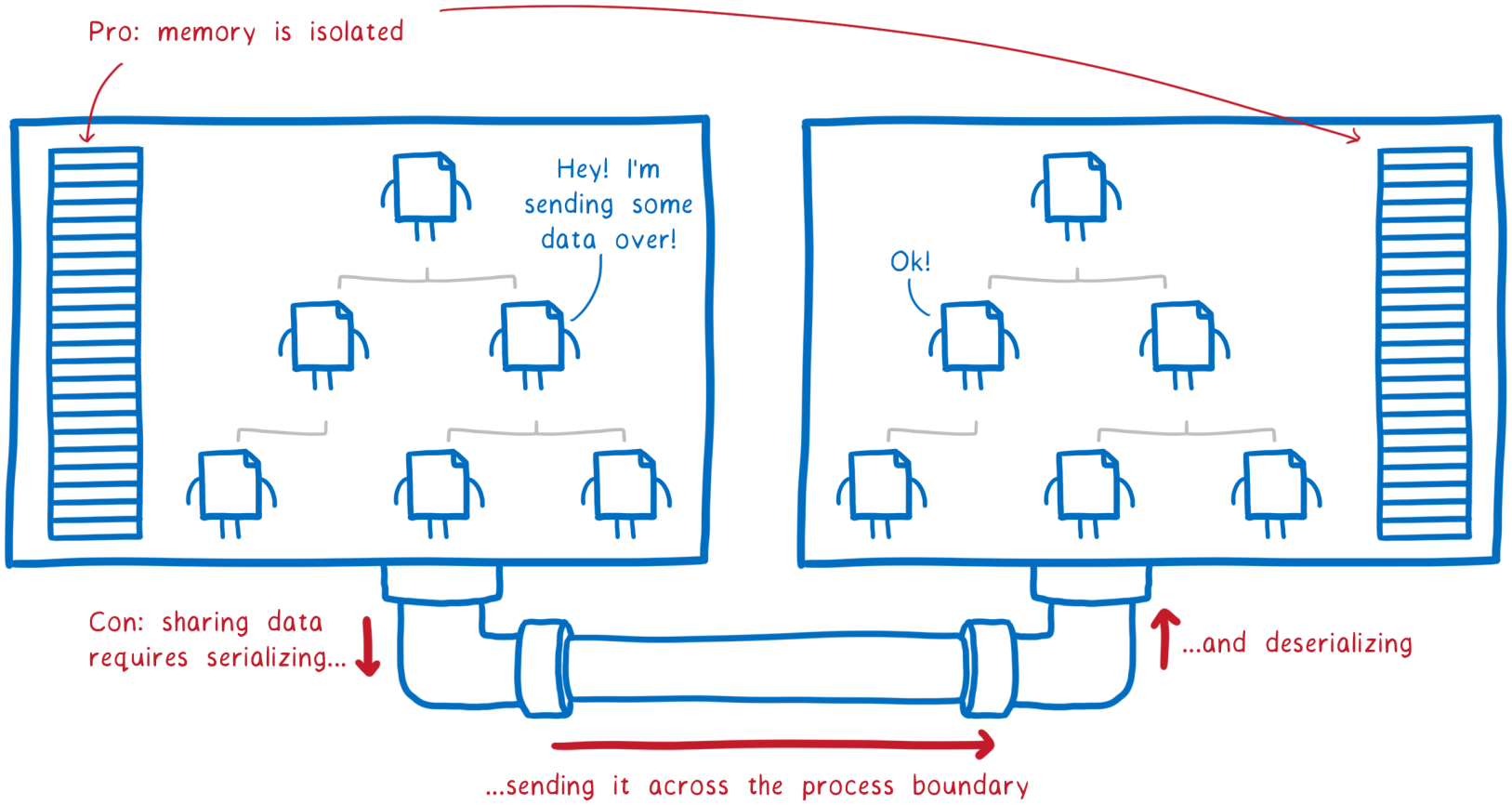
Malicious module



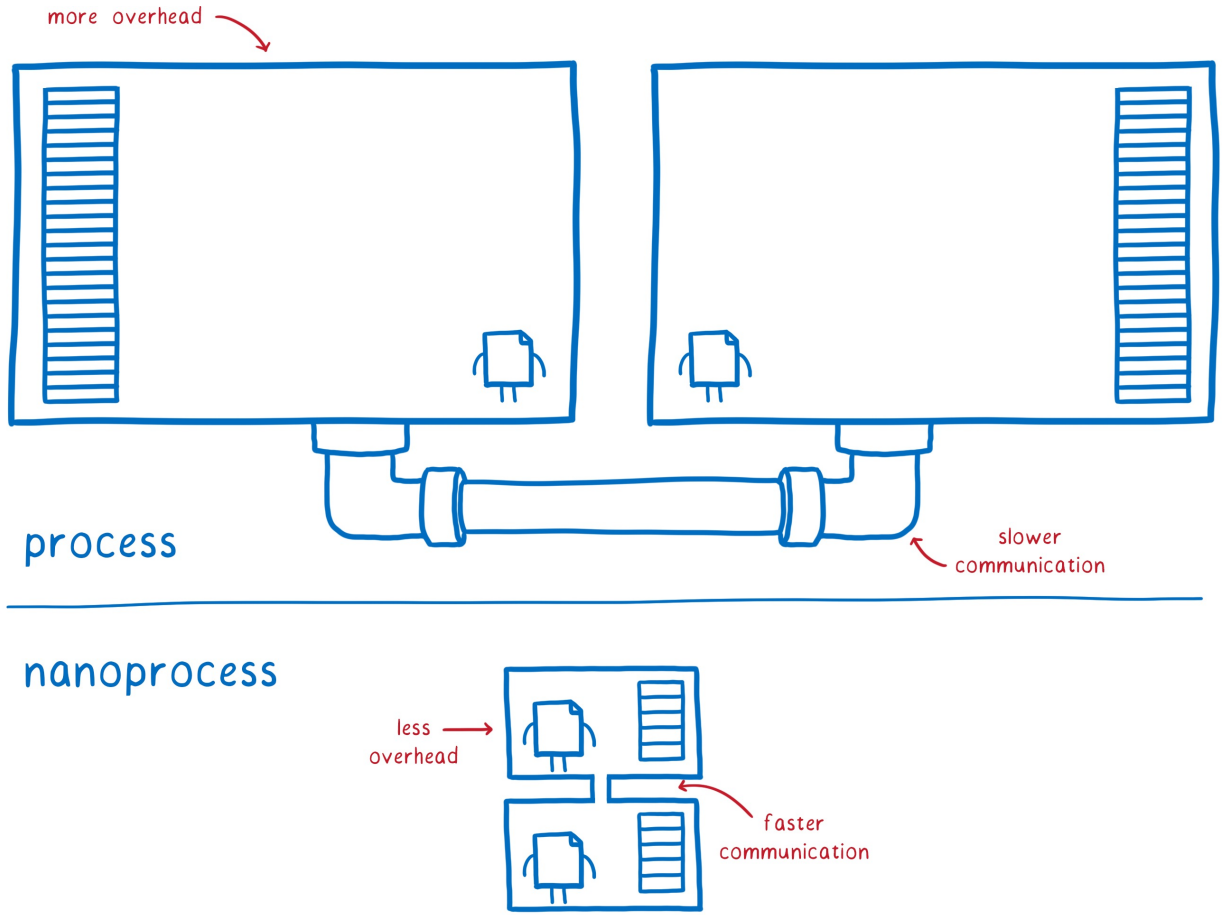
Vulnerable module



Process Isolation



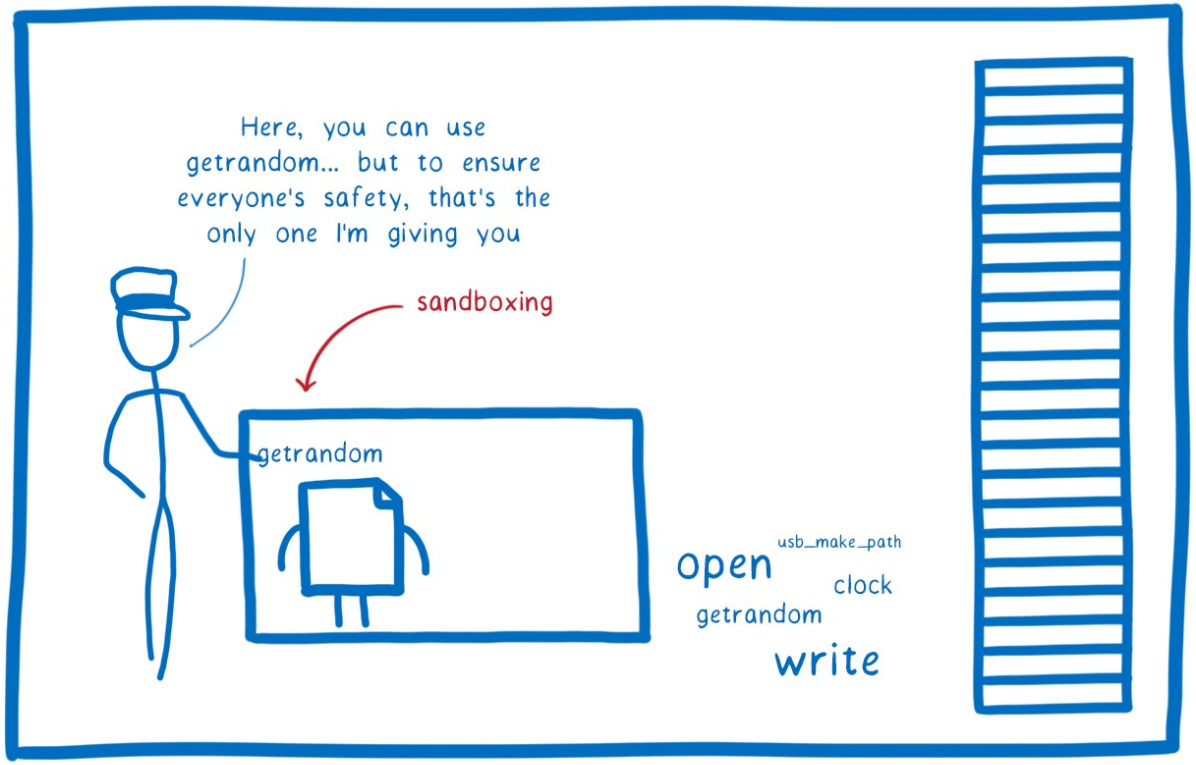
WebAssembly Nano-Process



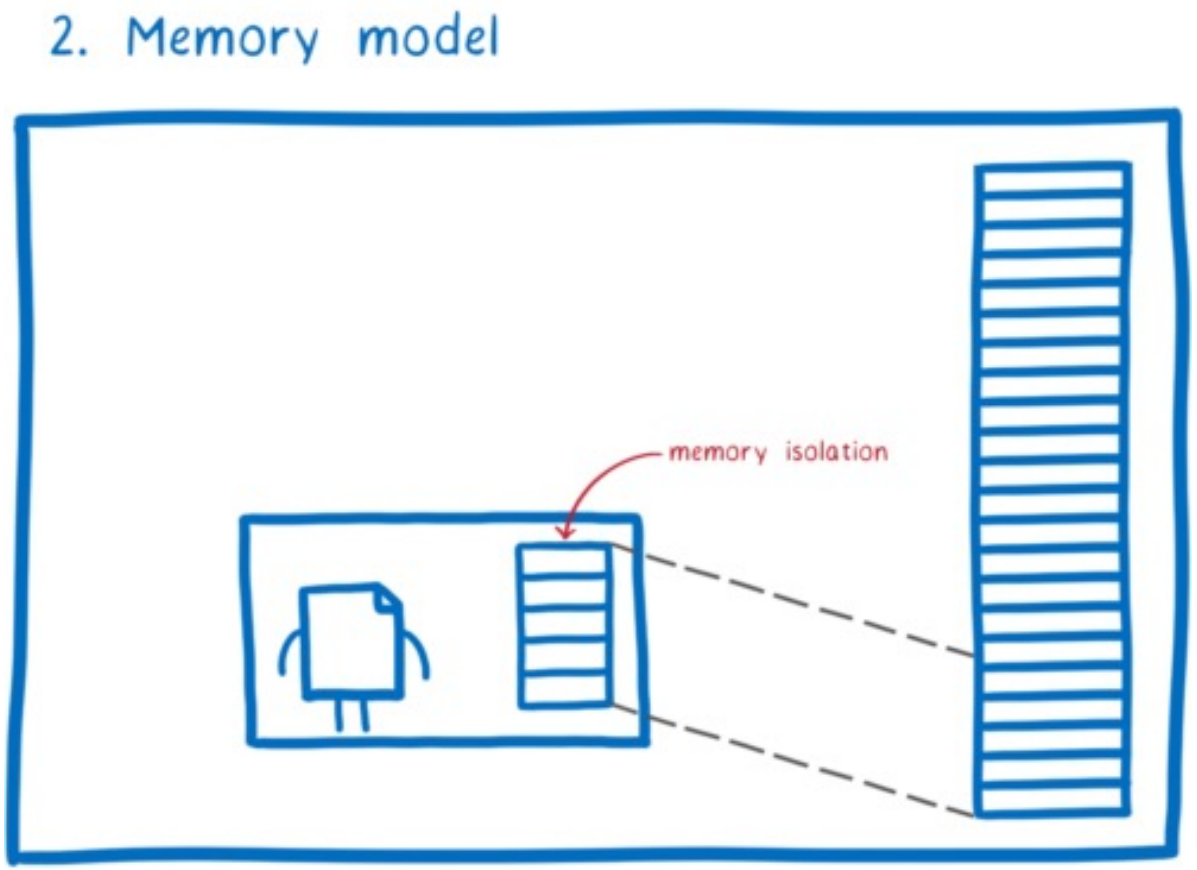
* not drawn to scale

WebAssembly Nano-Process

1. Sandboxing

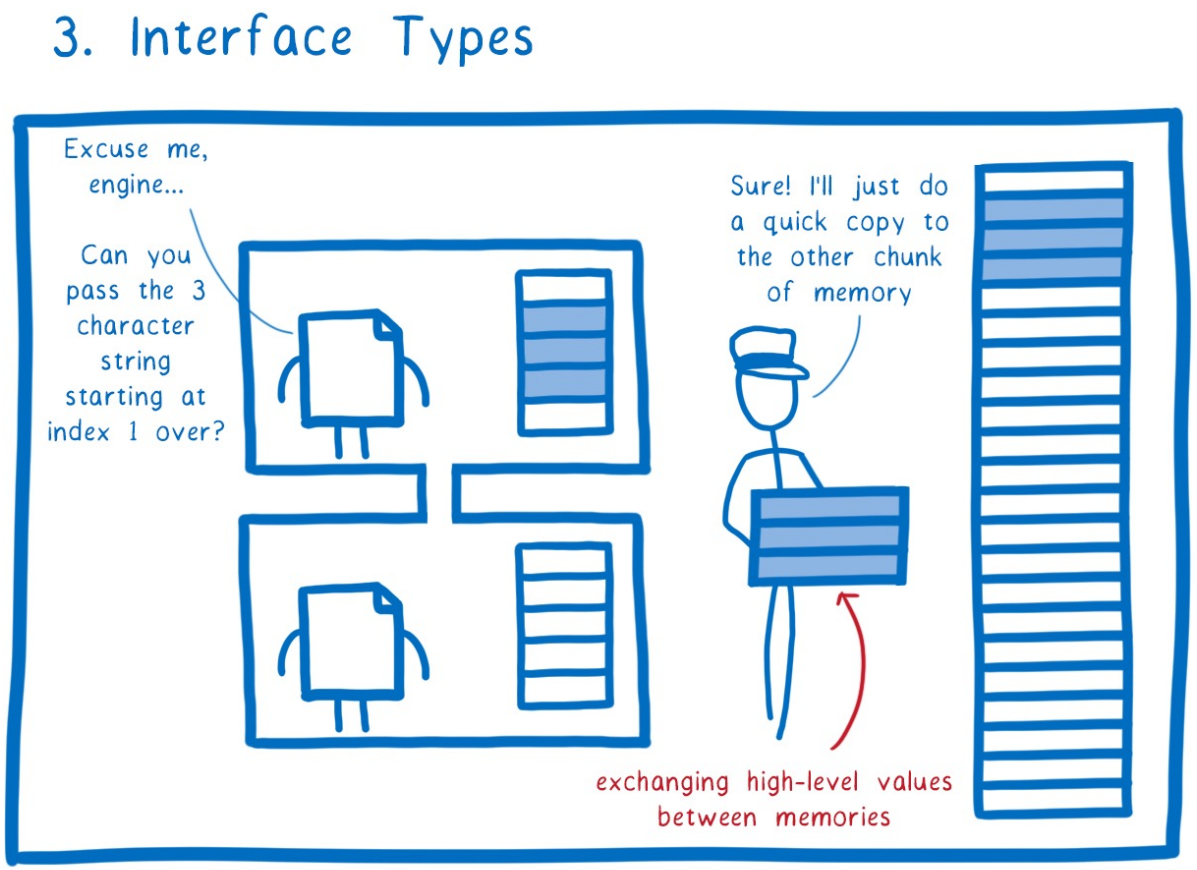


WebAssembly Nano-Process

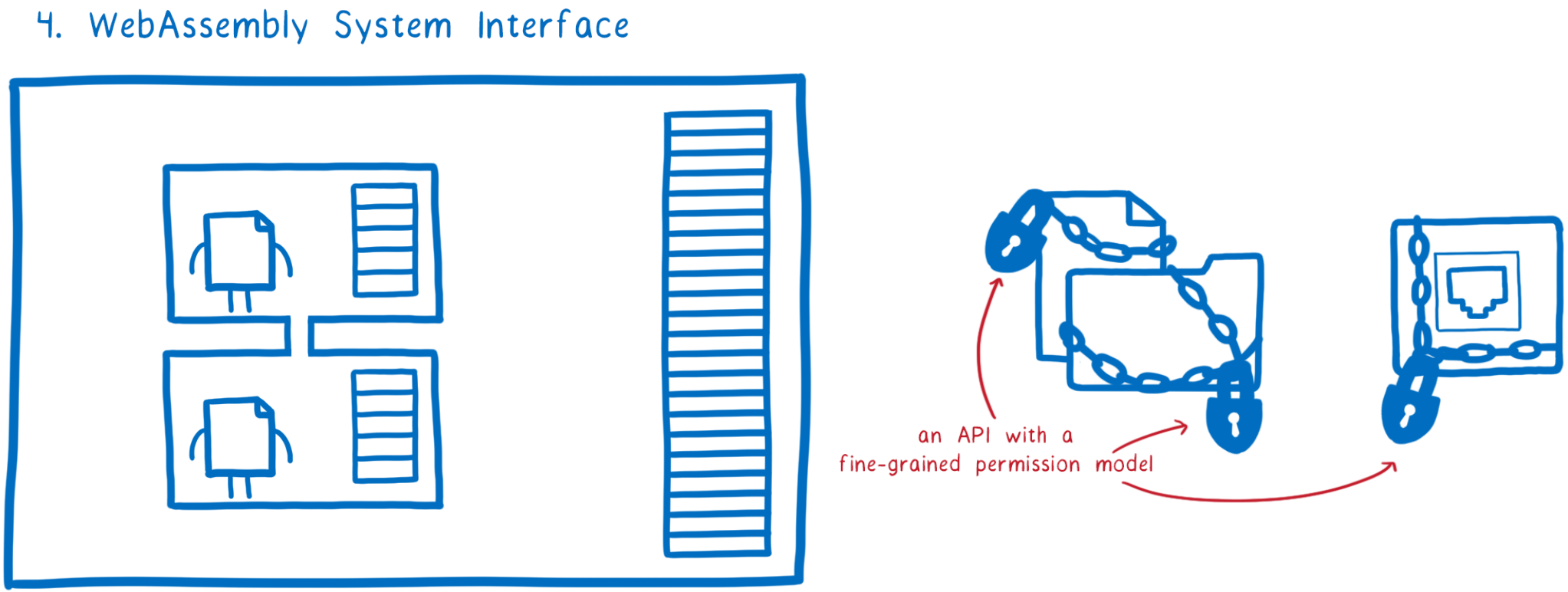


@nielstanis@infosec.exchange

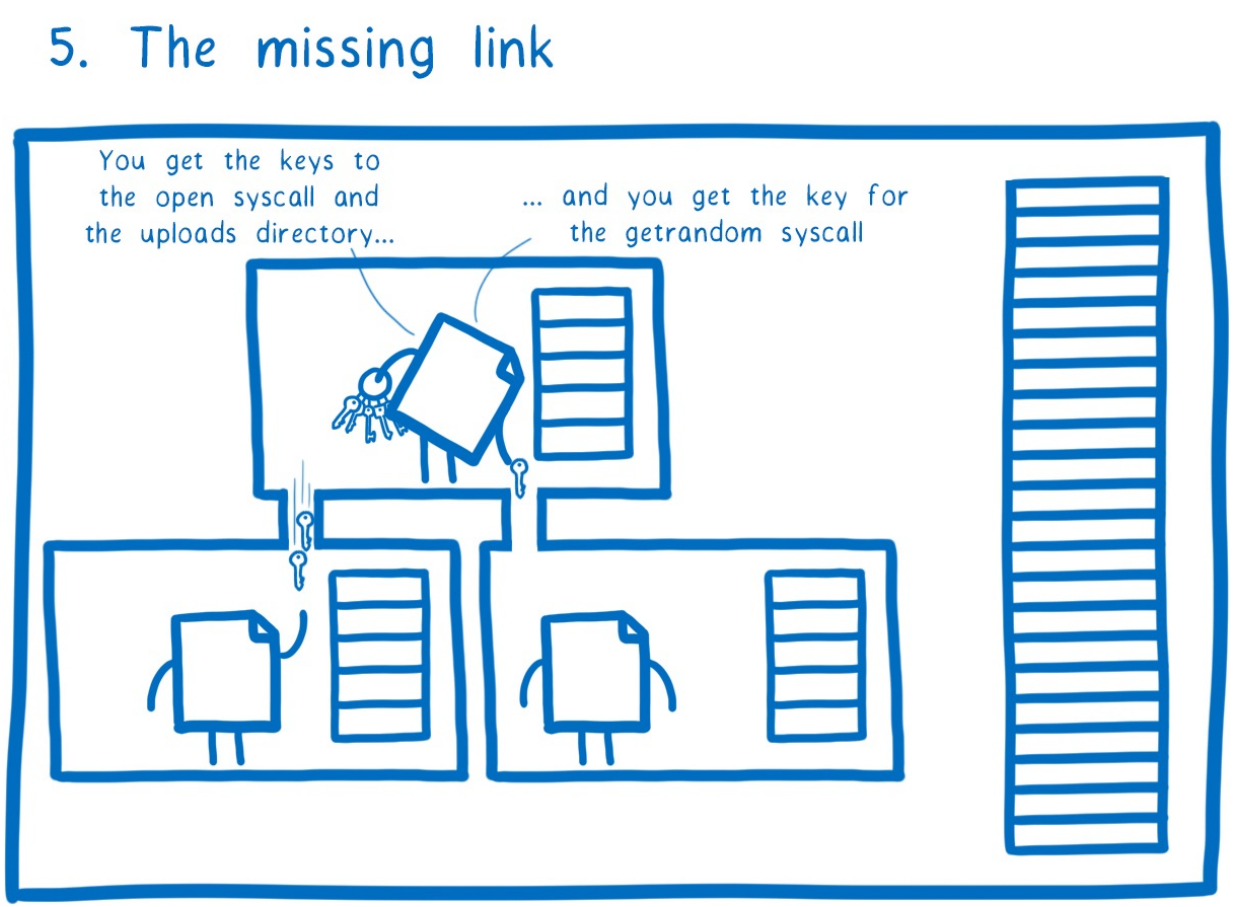
WebAssembly Nano-Process



WebAssembly Nano-Process

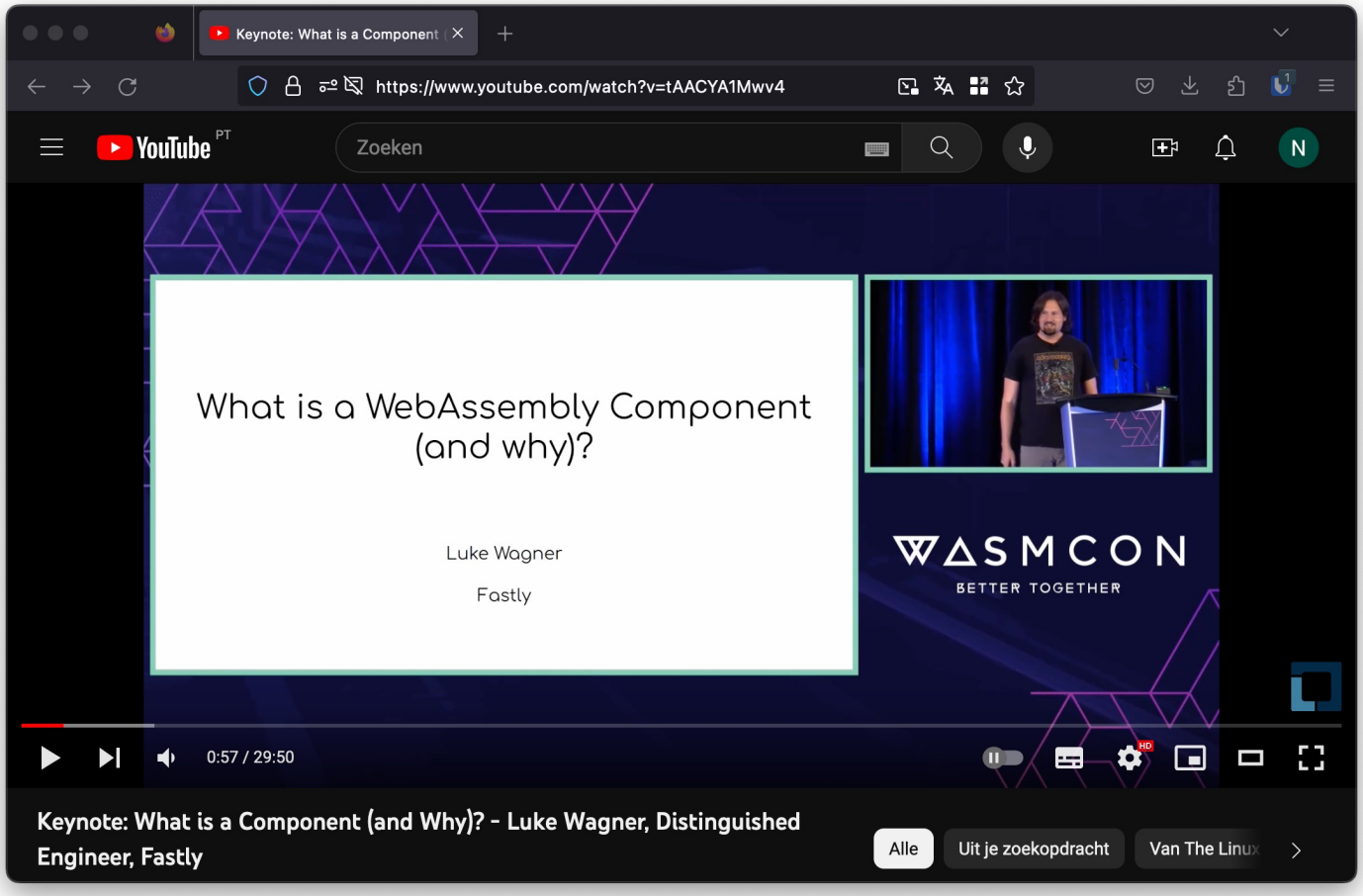


WebAssembly Nano-Process



@nielstanis@infosec.exchange

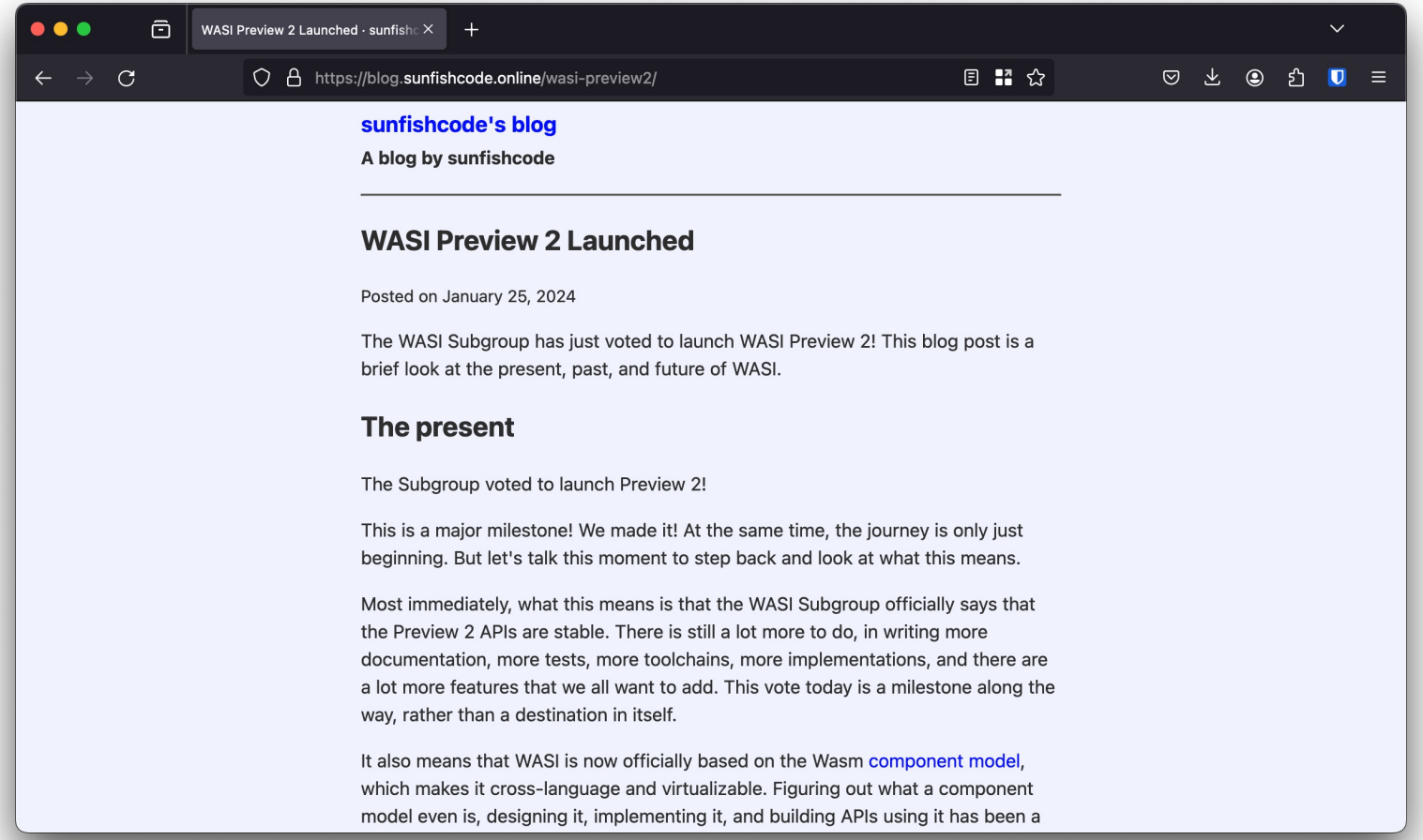
WebAssembly Component Model



SWETUGG

@nielstanis@infosec.exchange

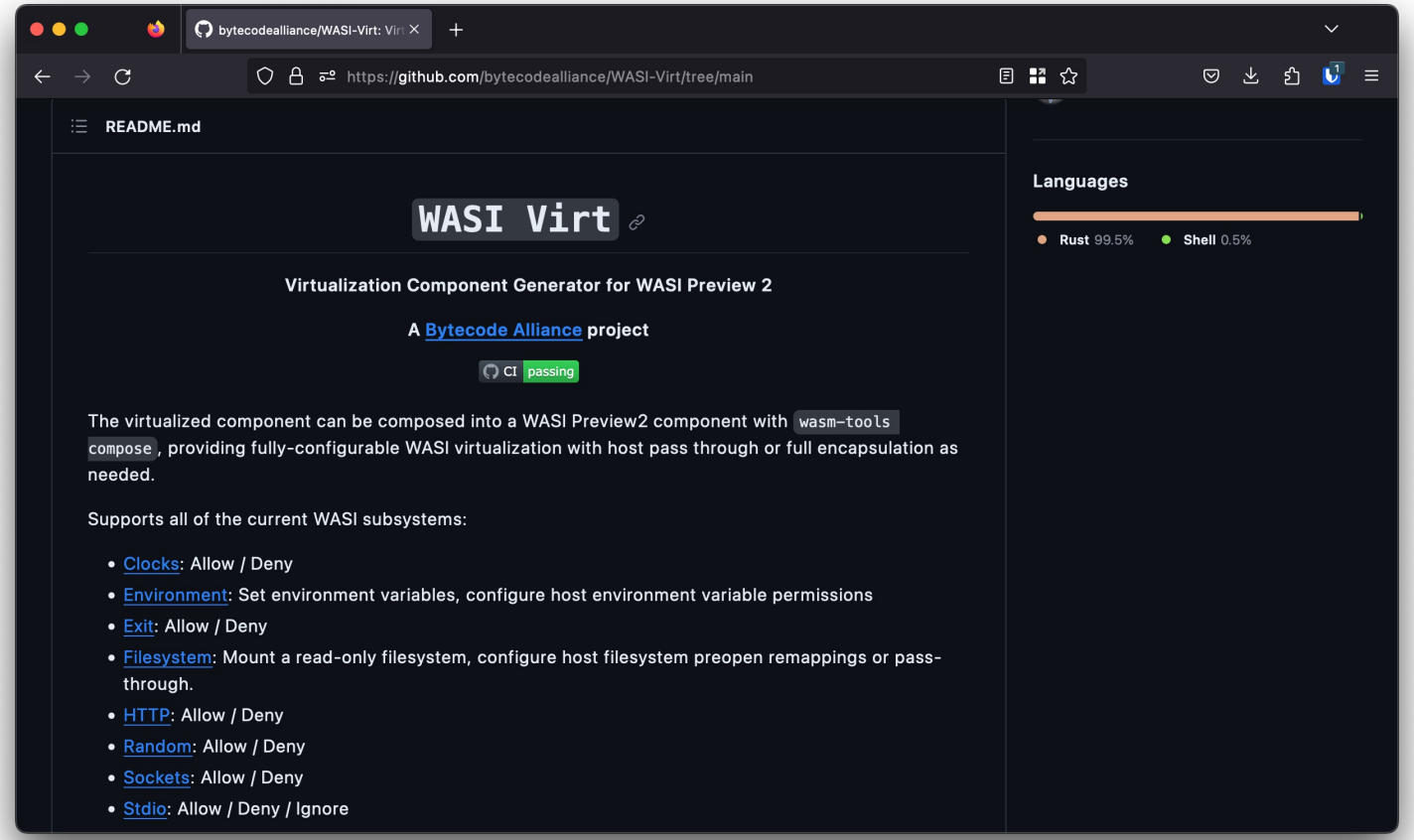
WASI Preview 2



SWETUGG

@nielstanis@infosec.exchange

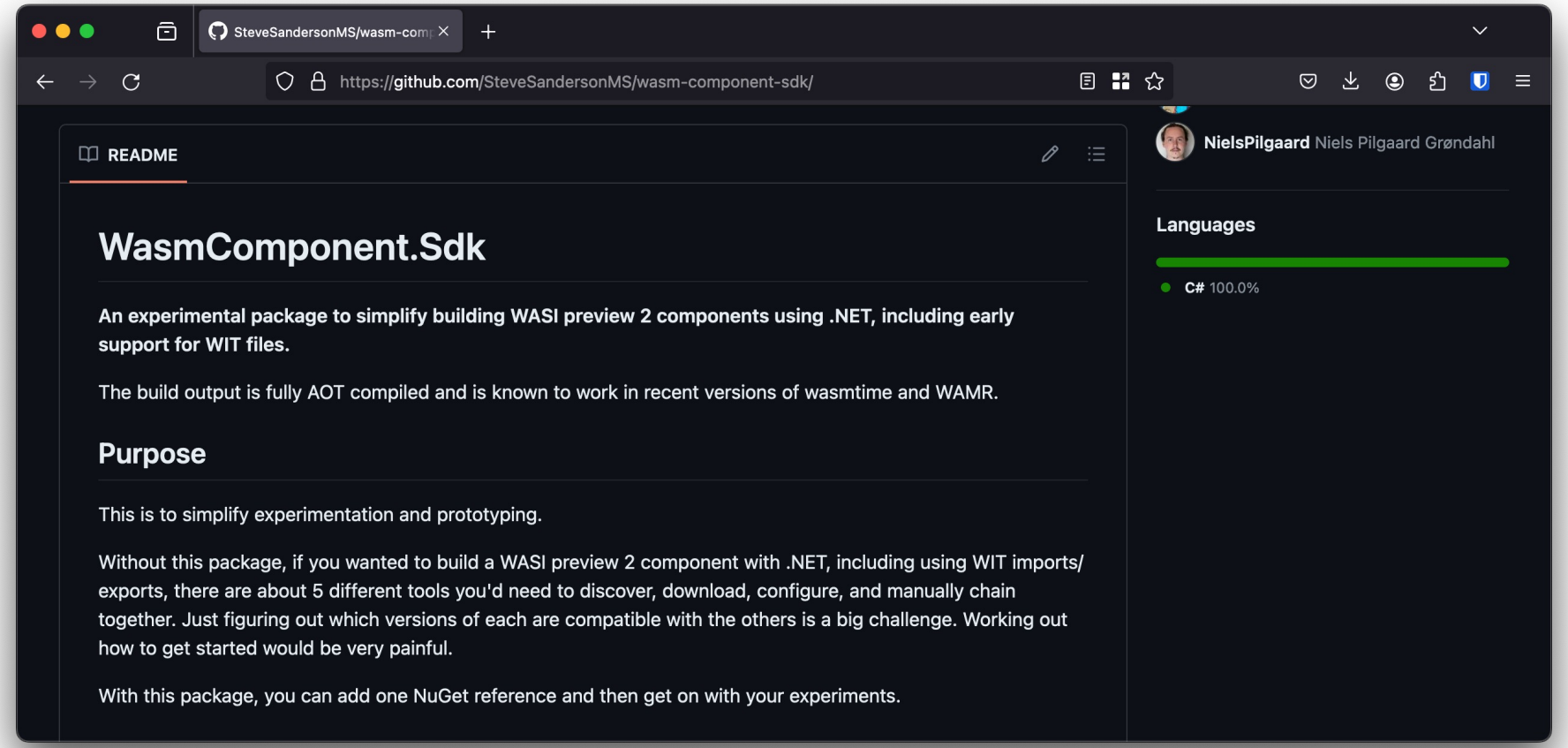
WASI Virt



SWETUGG

@nielstanis@infosec.exchange

WasmComponent.SDK



SWETUGG

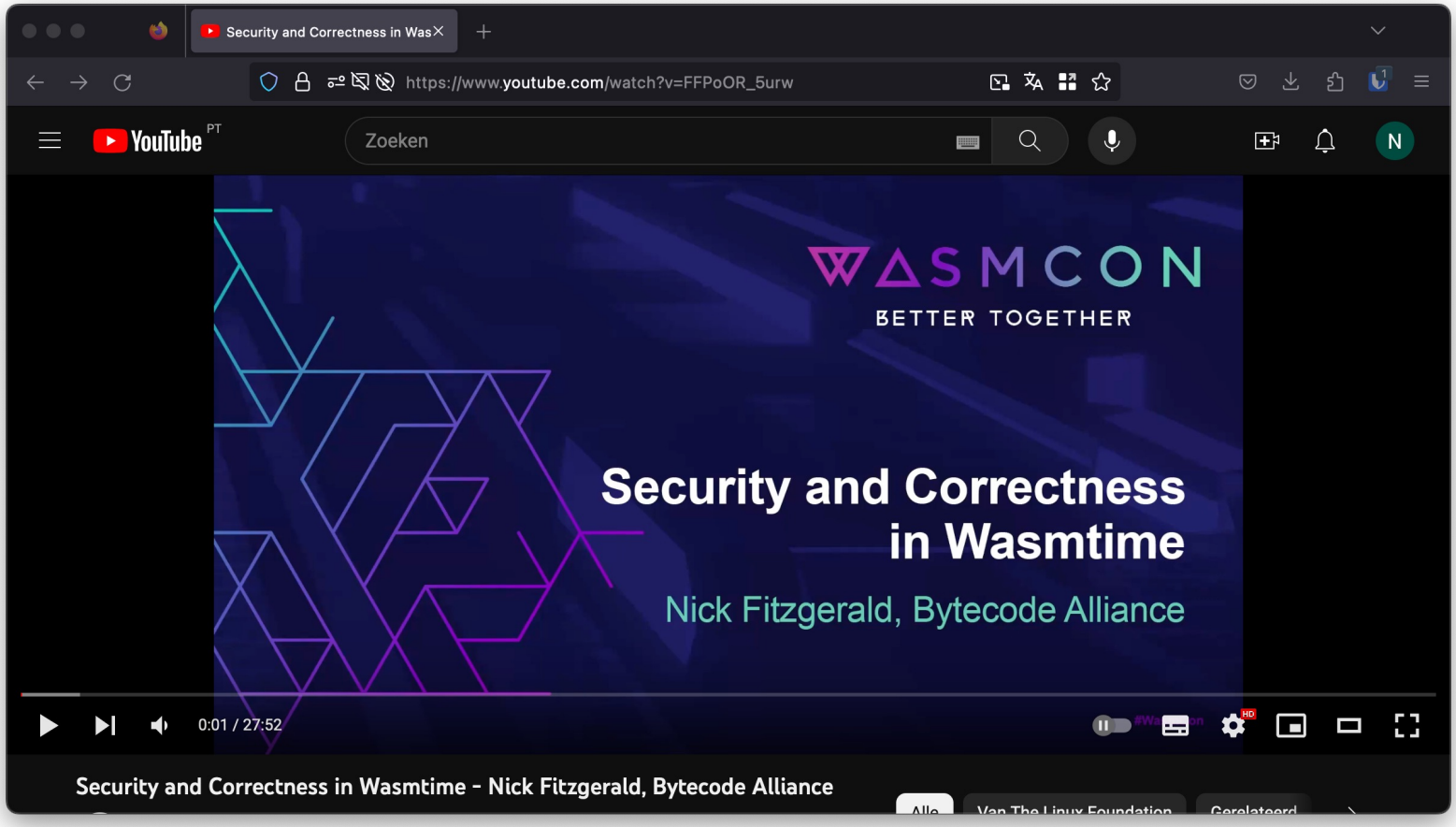
@nielstanis@infosec.exchange

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

Runtimes and Security



SWETUGG

@nielstanis@infosec.exchange

Conclusion

- Cloud Native ❤️ WebAssembly
- WebAssembly has a lot of potential to be used to run, extend, and secure your applications!
- Its as secure as the WebAssembly runtime implementation!
- WASI Preview 2 big milestone; now tooling can be implemented!

Questions?

- https://github.com/nielstanis/swetugg2024
- ntanis at Veracode.com
- @nielstanis@infosec.exchange
- https://blog.fennec.dev
- Tack! Thank you!