# 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 Design

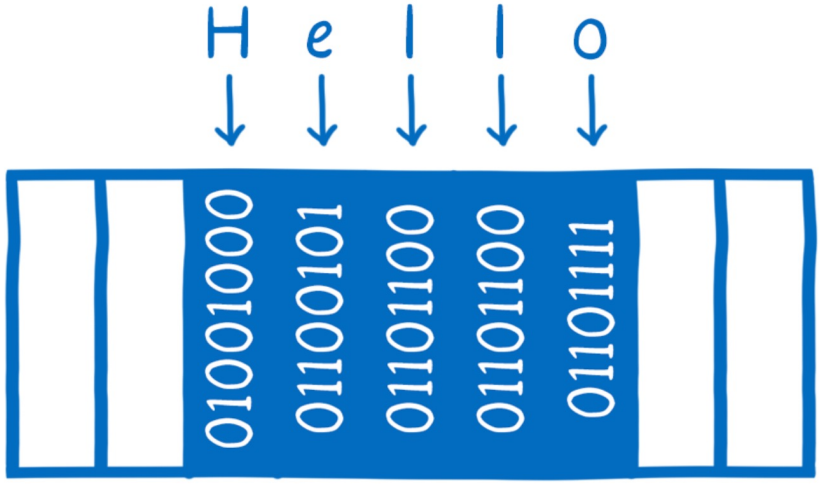
- **Be fast, efficient, and portable**
  - Executed in near-native speed across different platforms
- **Be readable and debuggable**
  - In low-level bytecode but also human readable
- **Keep secure**
  - Run on sandboxed execution environment
- **Don't break the web**
  - Ensure backwards compatibility

# 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

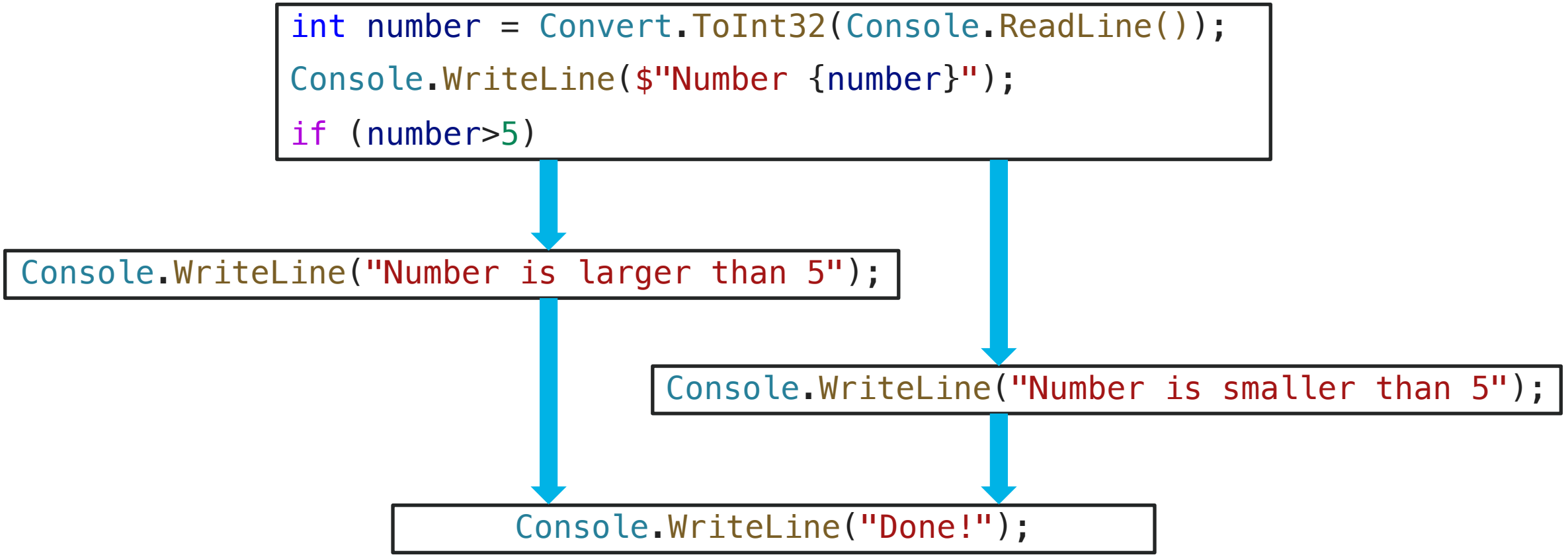
- Isolated per WASM module
- A contiguous, mutable array of uninterpreted bytes



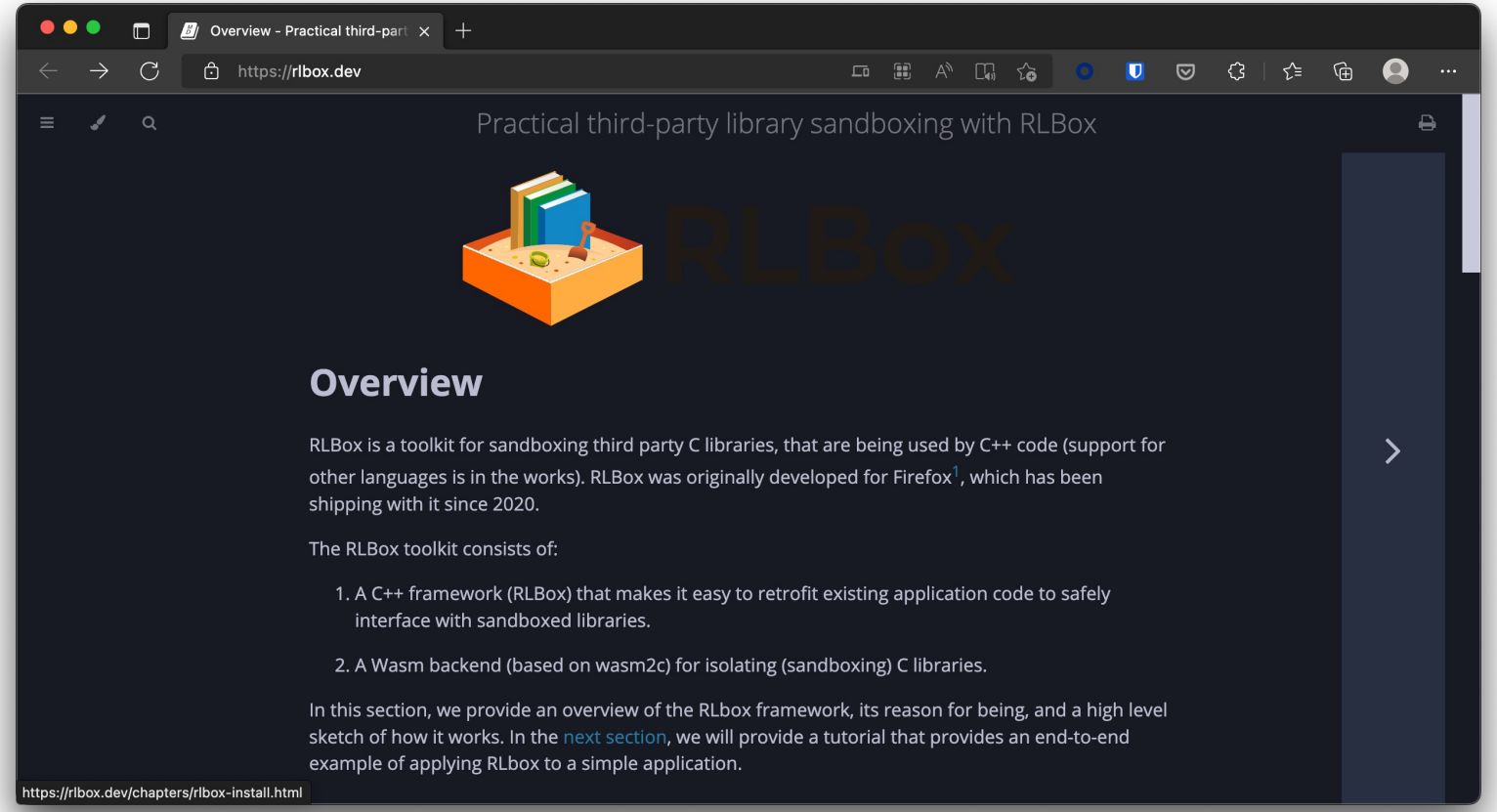
# WebAssembly Control-Flow Integrity

```
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!");
```

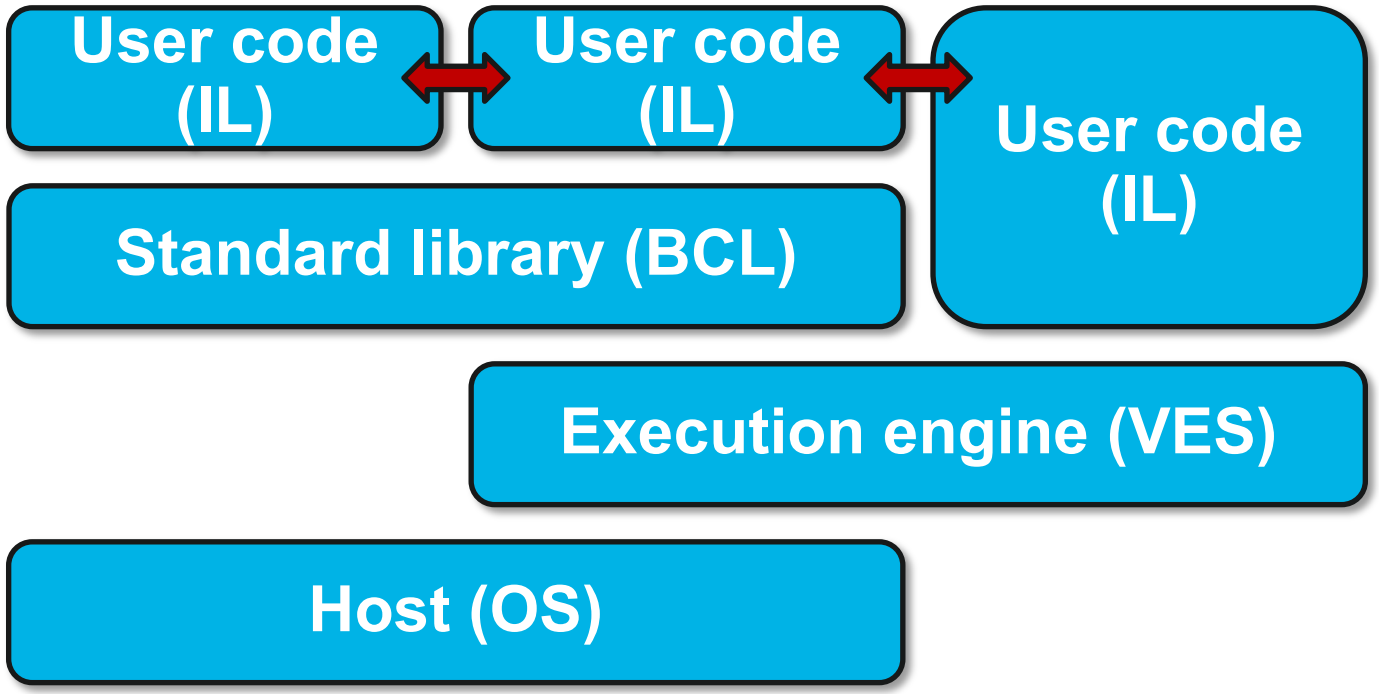
# WebAssembly Control-Flow Integrity



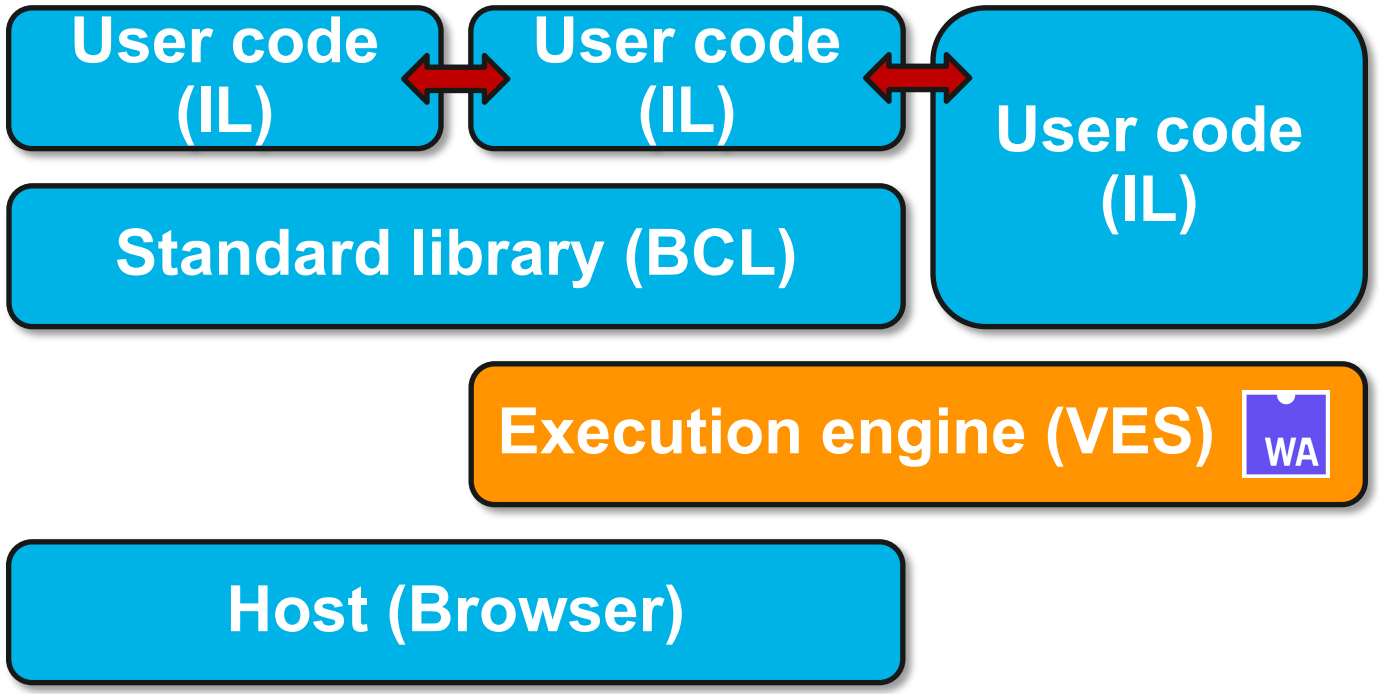
# FireFox RLBox



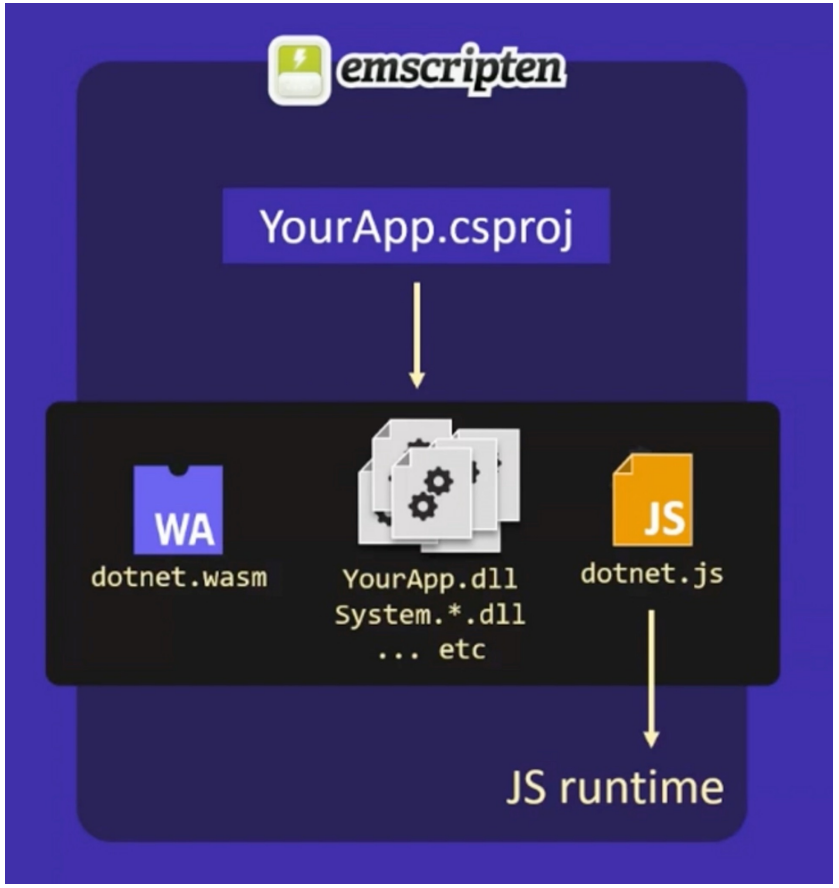
# Running .NET on WebAssembly



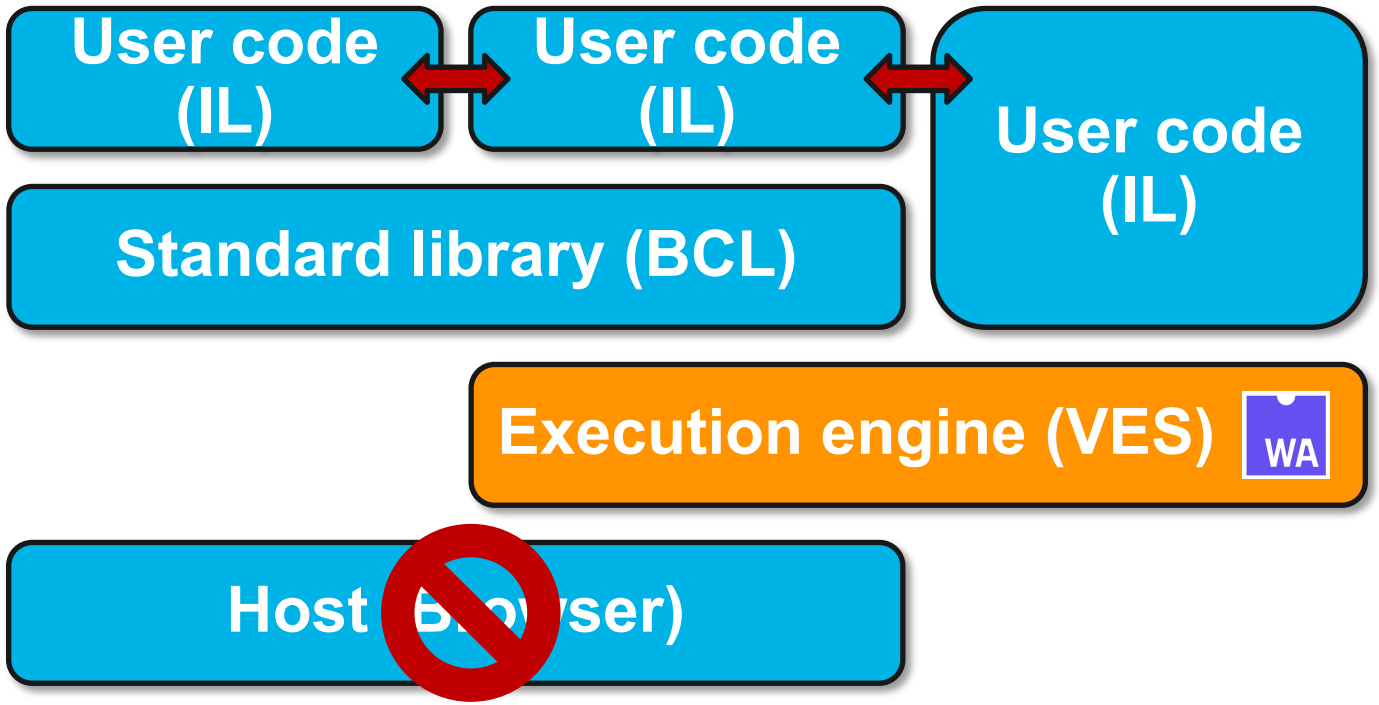
# Running .NET on WebAssembly



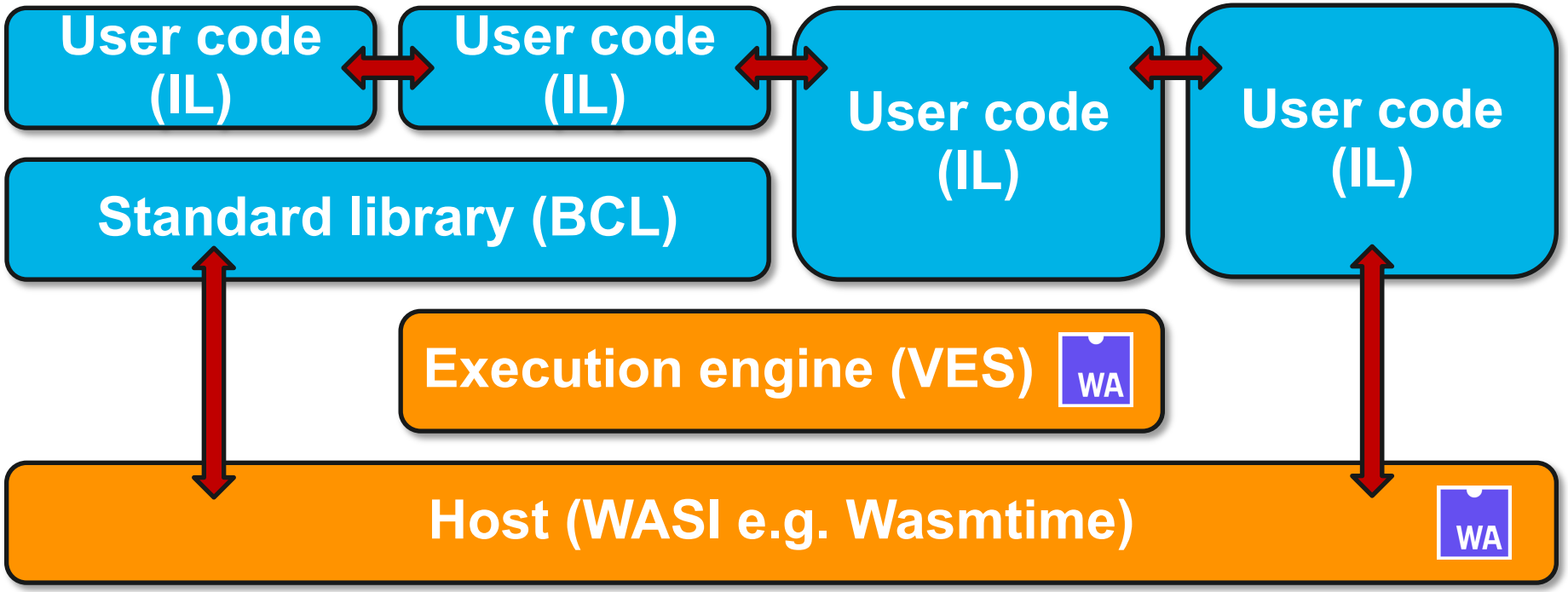
# Blazor WebAssembly



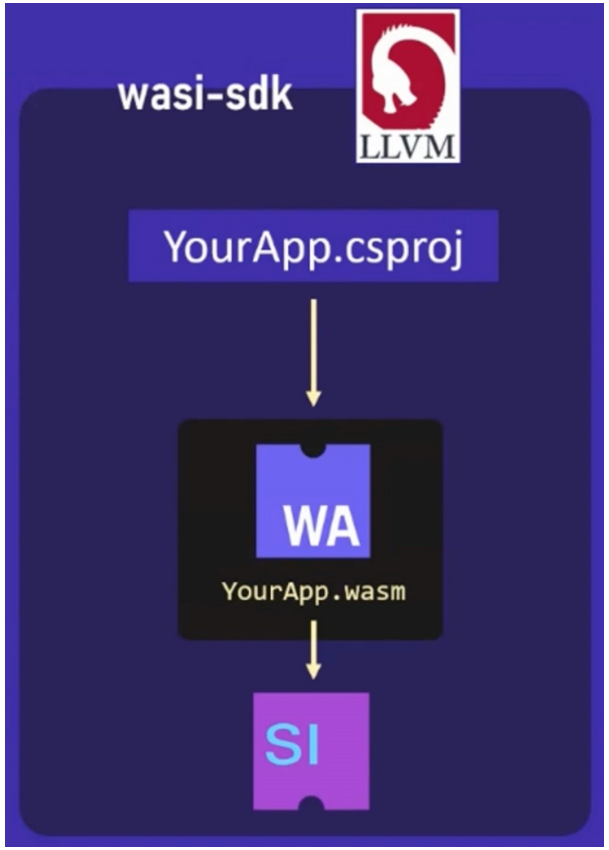
# Running .NET on WebAssembly



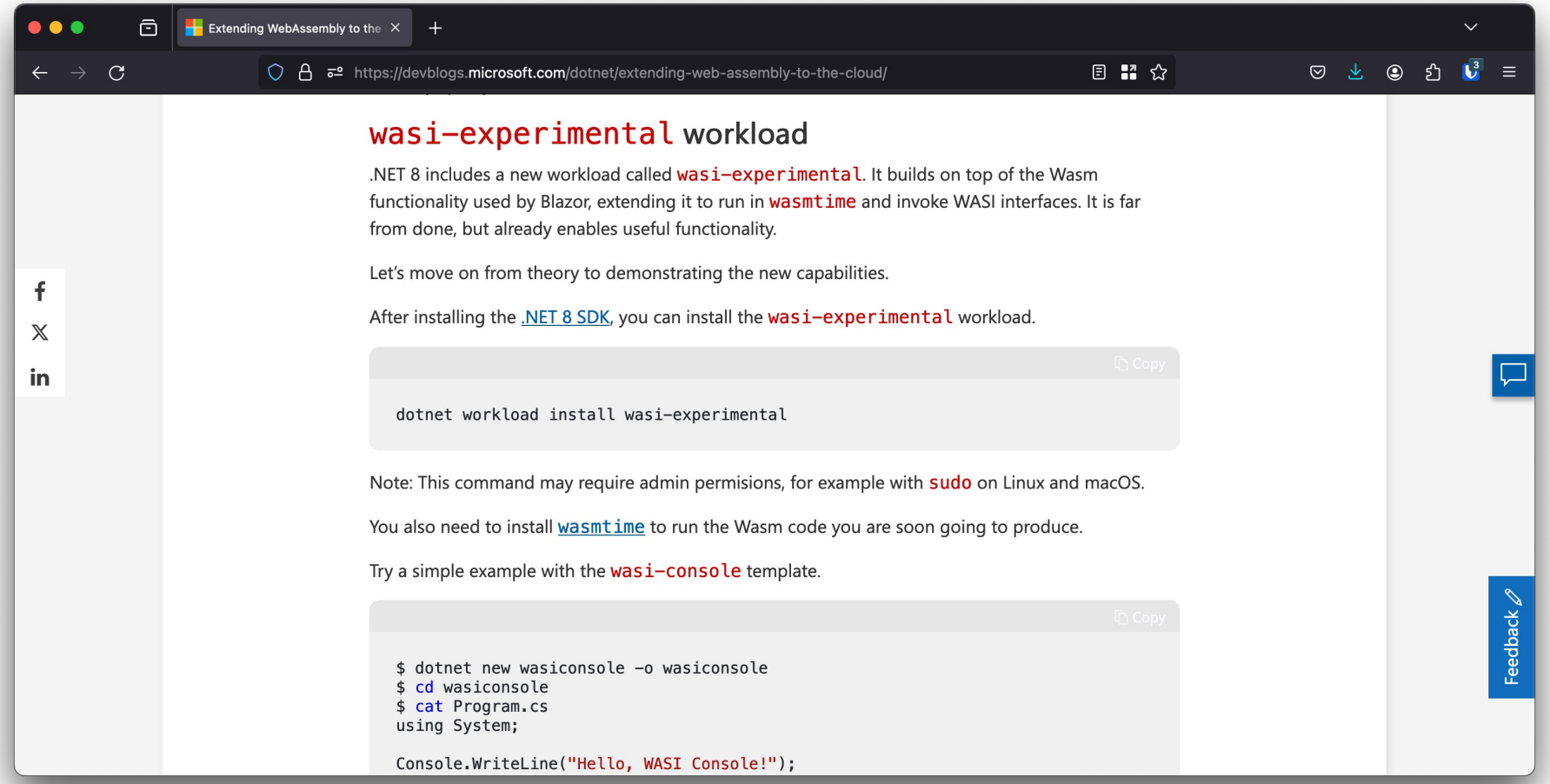
# WebAssembly System Interface WASI



# Experimental WASI SDK for .NET



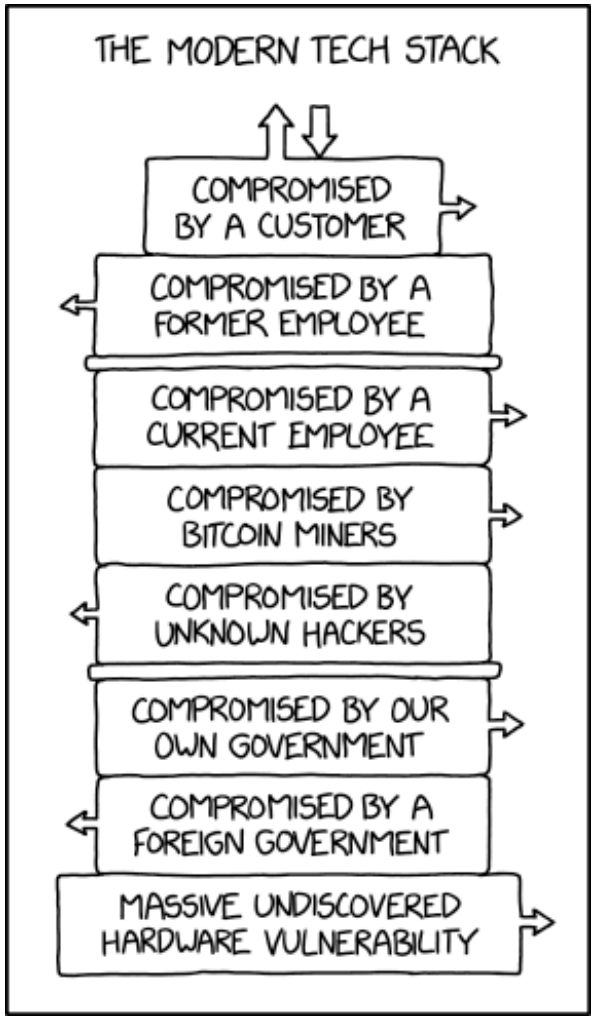
# .NET 8 WASI-Experimental



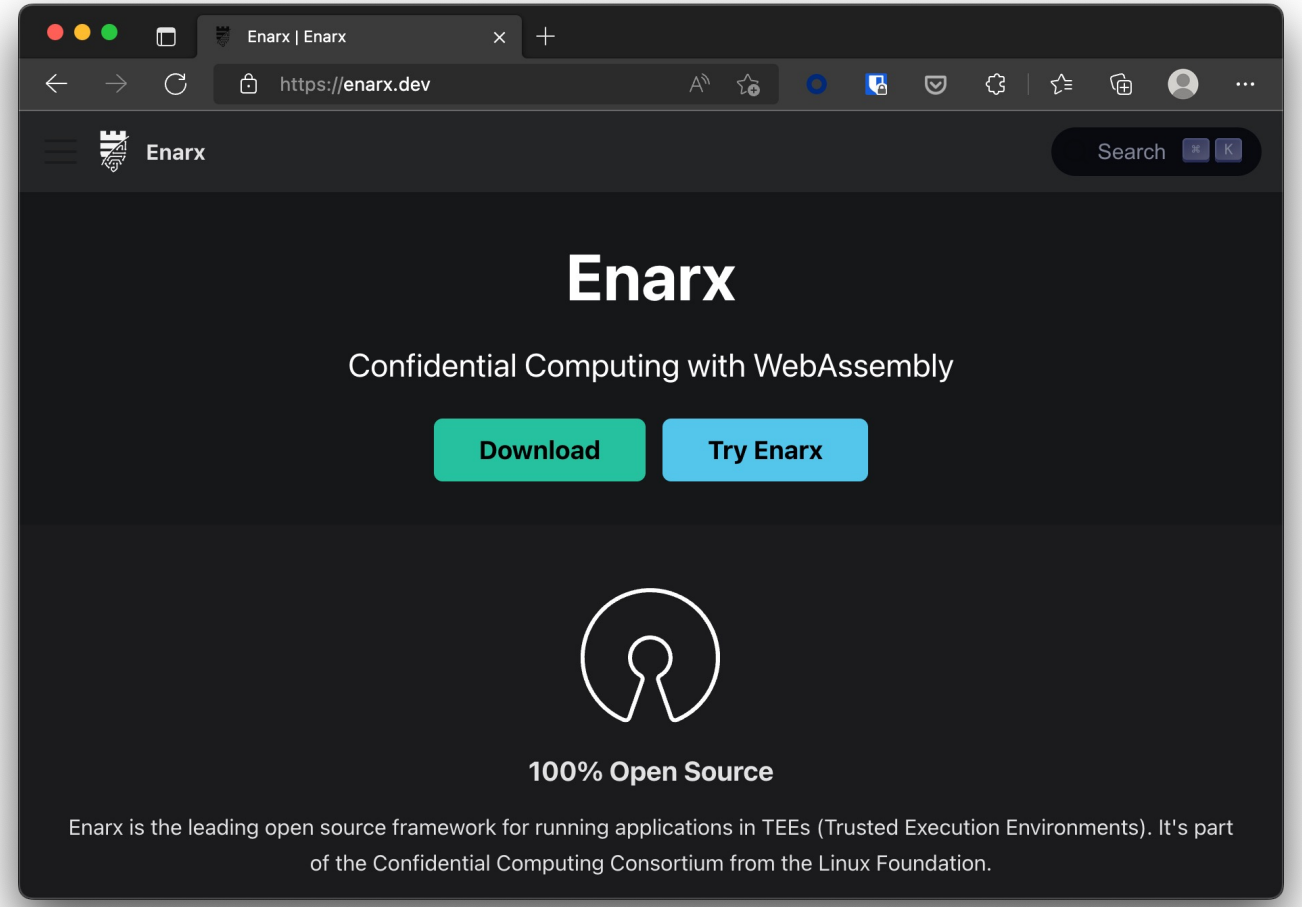
# Extending .NET with WASM

- 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

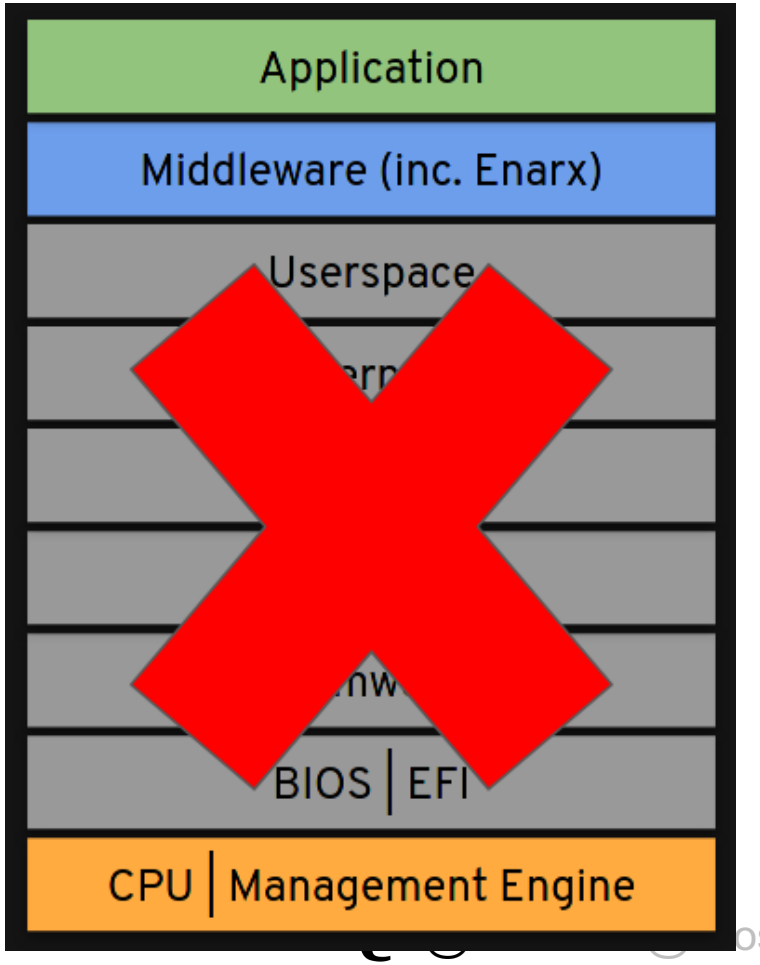


# Enarx


Enarx | Enarx
https://enarx.dev
Enarx
Search
Enarx
Confidential Computing with WebAssembly
Download
Try Enarx
100% Open Source
Enarx is the leading open source framework for running applications in TEEs (Trusted Execution Environments). It's part of the Confidential Computing Consortium from the Linux Foundation.

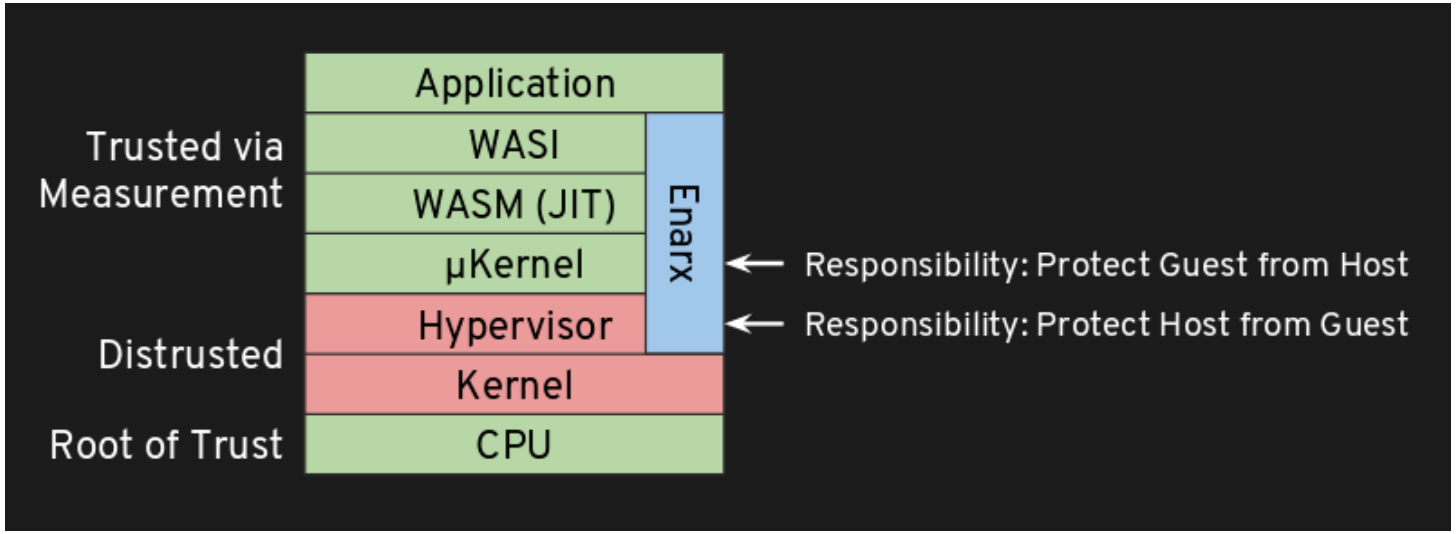
# Enarx Threat Model

- 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

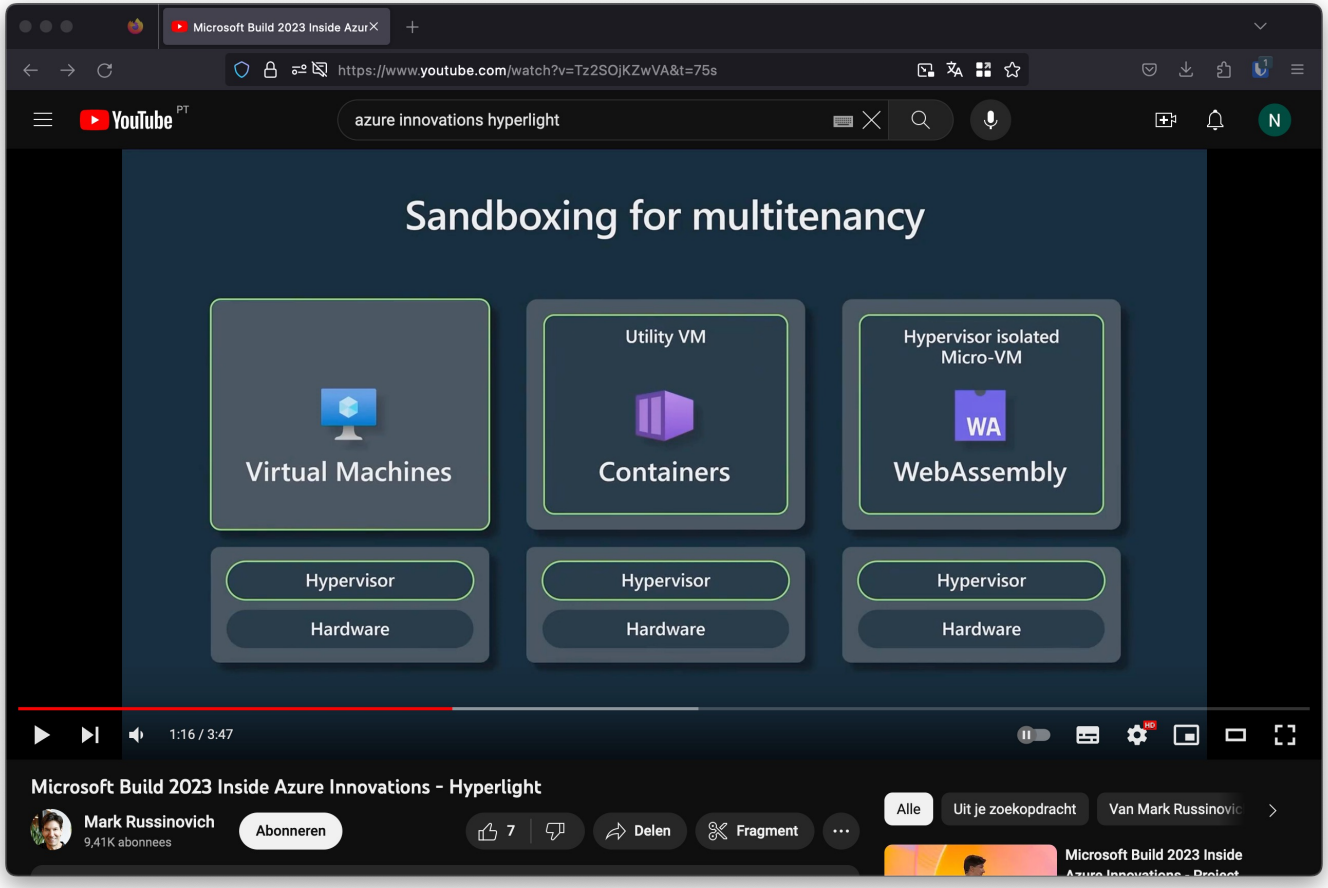
- Leverages Trusted Execution Environment (TEE) direct on processor
  - AMD's SEV, Intel's SGX and IBM's PEF
- Attestation of hardware and Enarx runtime



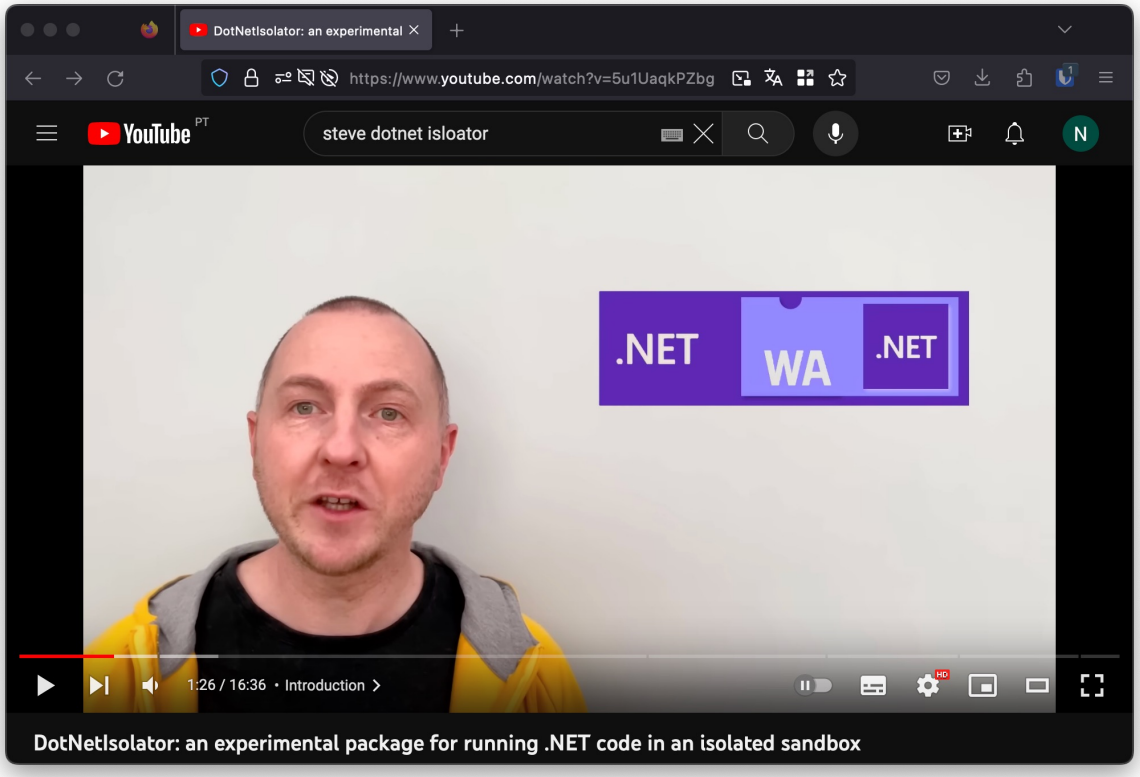


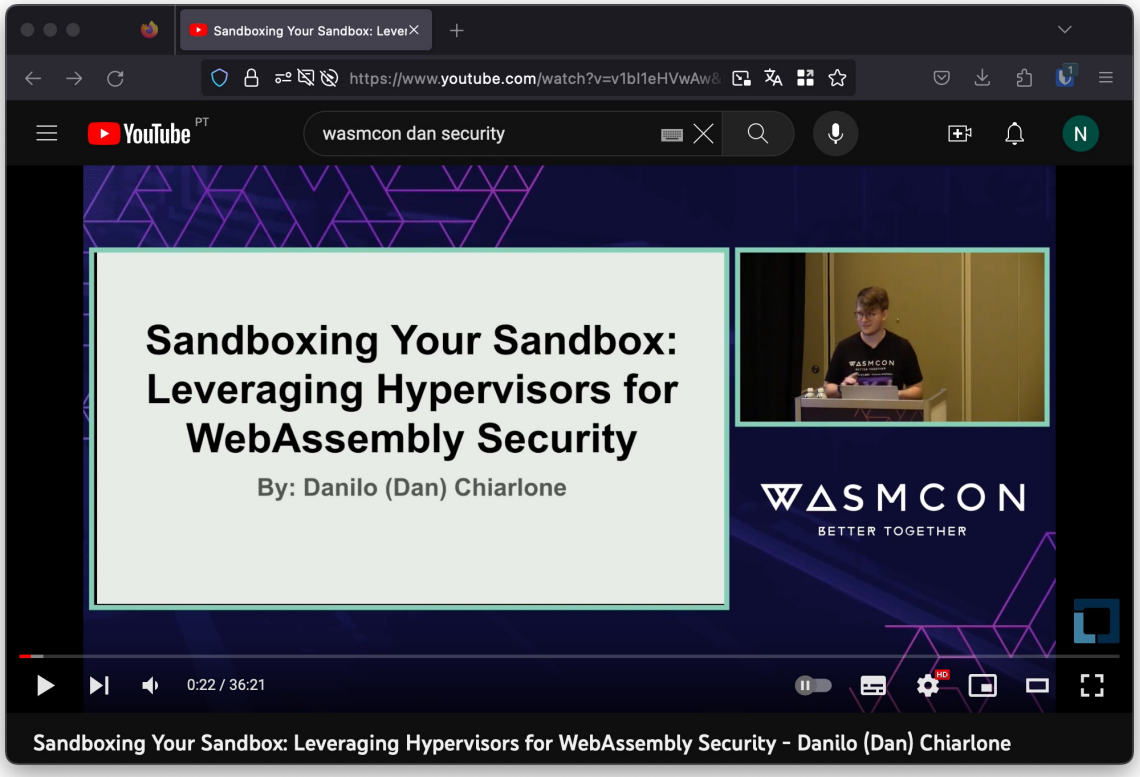

@nielstanis@infosec.exchange

# Project Hyperlight

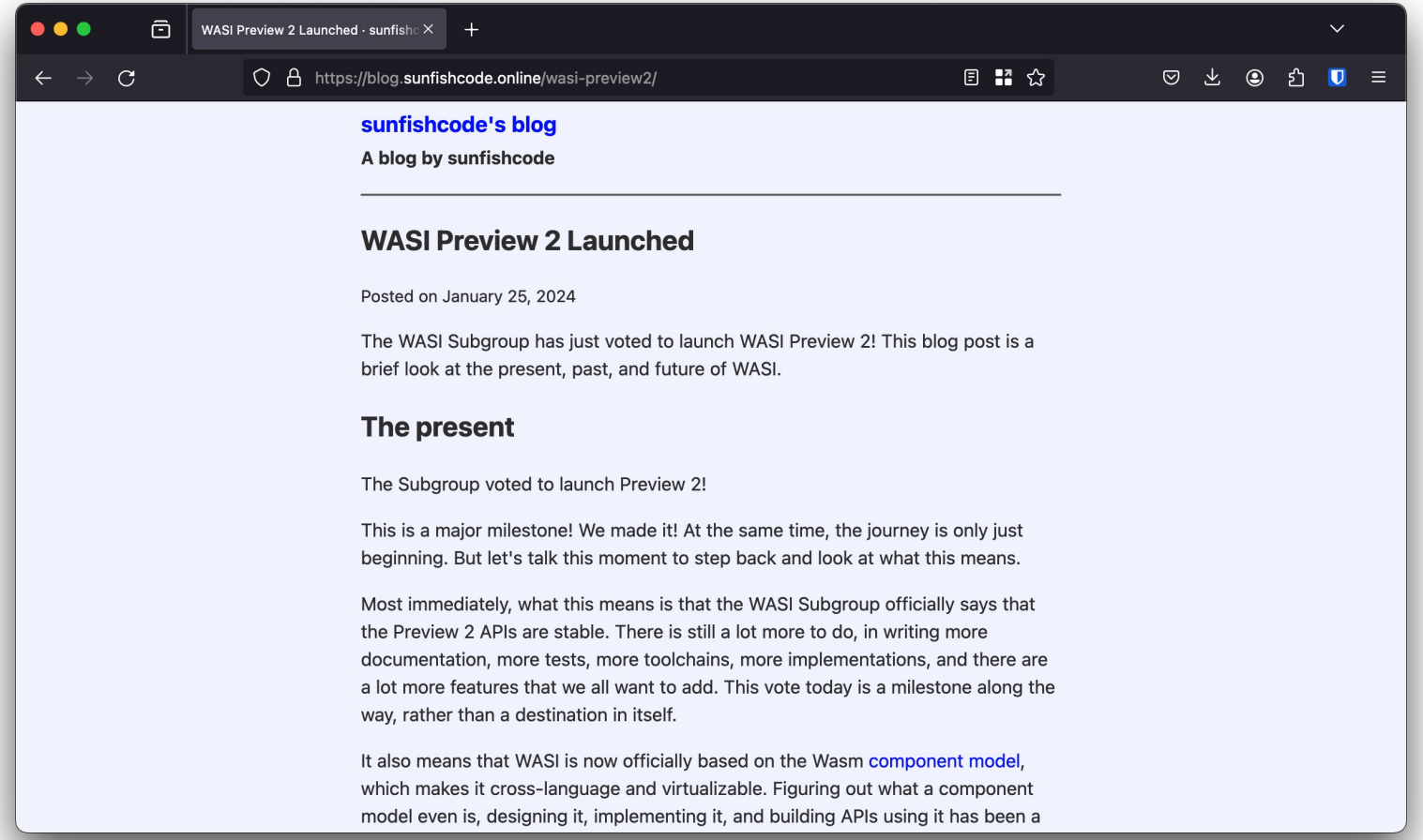


# DotNetIsolator & Project Hyperlight

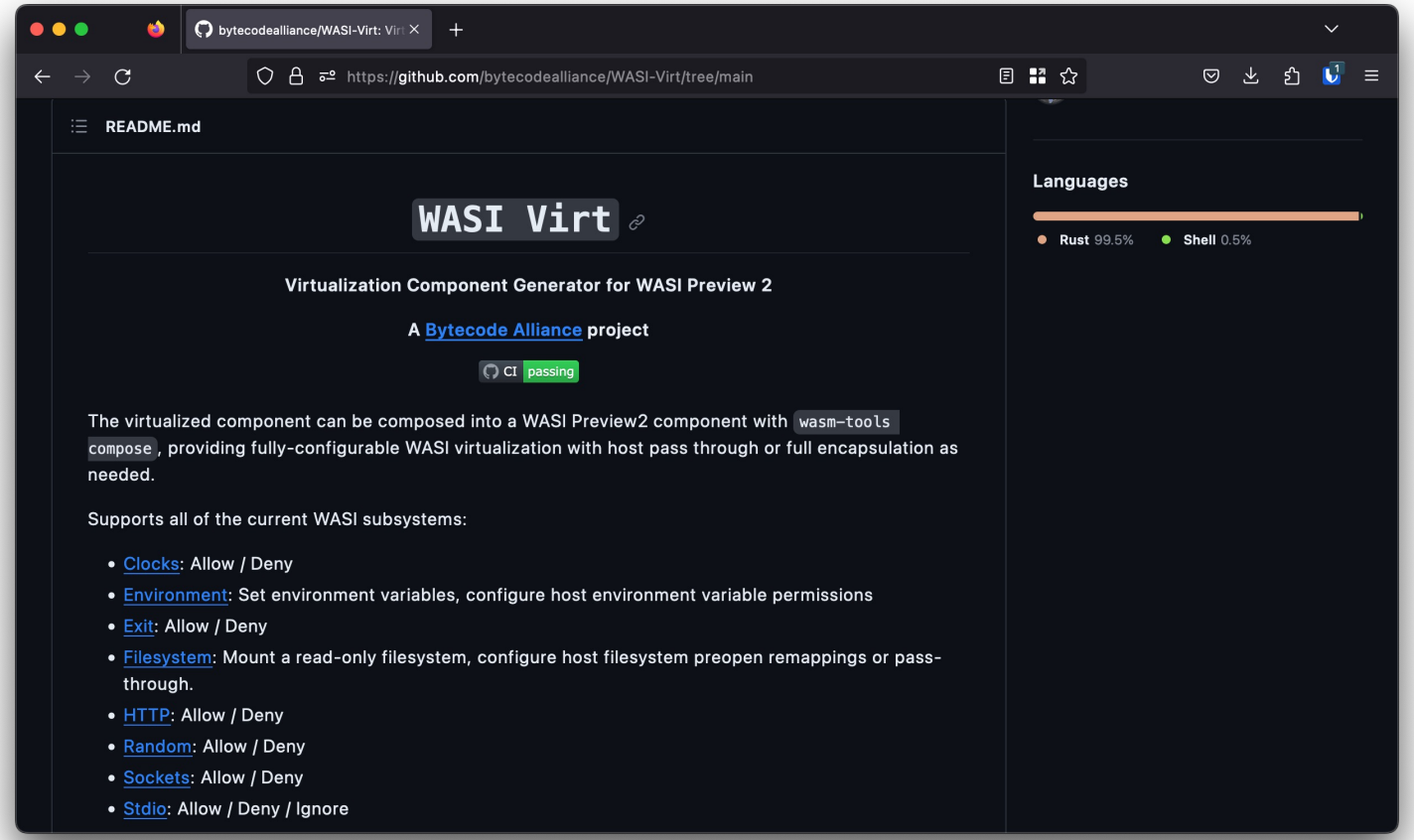


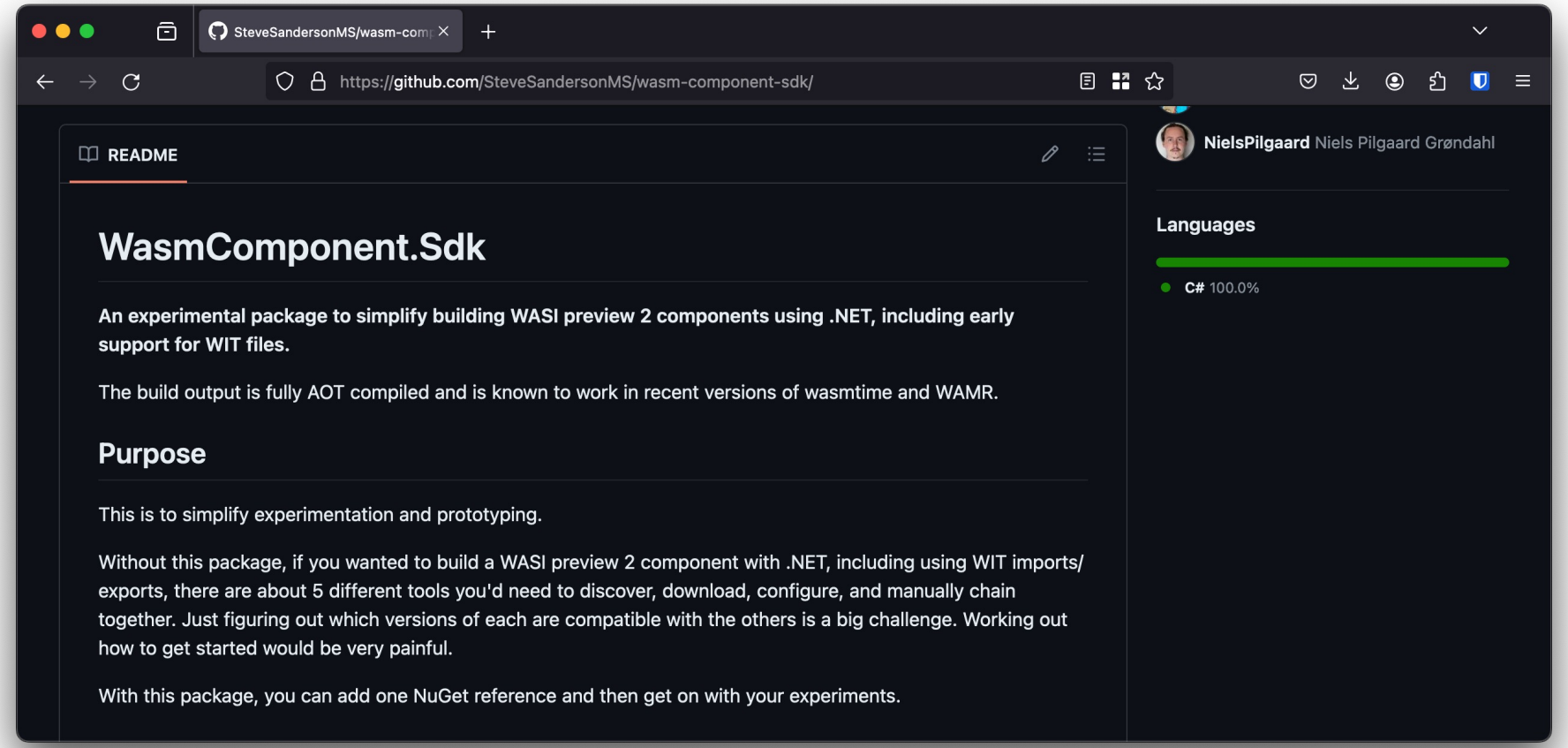
# WASI Preview 2



# WASI Virt


bytecodealliance/WASI-Virt: Virt
https://github.com/bytecodealliance/WASI-Virt/tree/main
README.md
WASI Virt
Virtualization Component Generator for WASI Preview 2
A Bytecode Alliance project
CI passing
The virtualized component can be composed into a WASI Preview2 component with wasm-tools compose, providing fully-configurable WASI virtualization with host pass through or full encapsulation as needed.
Supports all of the current WASI subsystems:
Clocks: Allow / Deny
Environment: Set environment variables, configure host environment variable permissions
Exit: Allow / Deny
Filesystem: Mount a read-only filesystem, configure host filesystem preopen remappings or pass-through.
HTTP: Allow / Deny
Random: Allow / Deny
Sockets: Allow / Deny
Stdio: Allow / Deny / Ignore
Languages
Rust 99.5%
Shell 0.5%

# WasmComponent.SDK

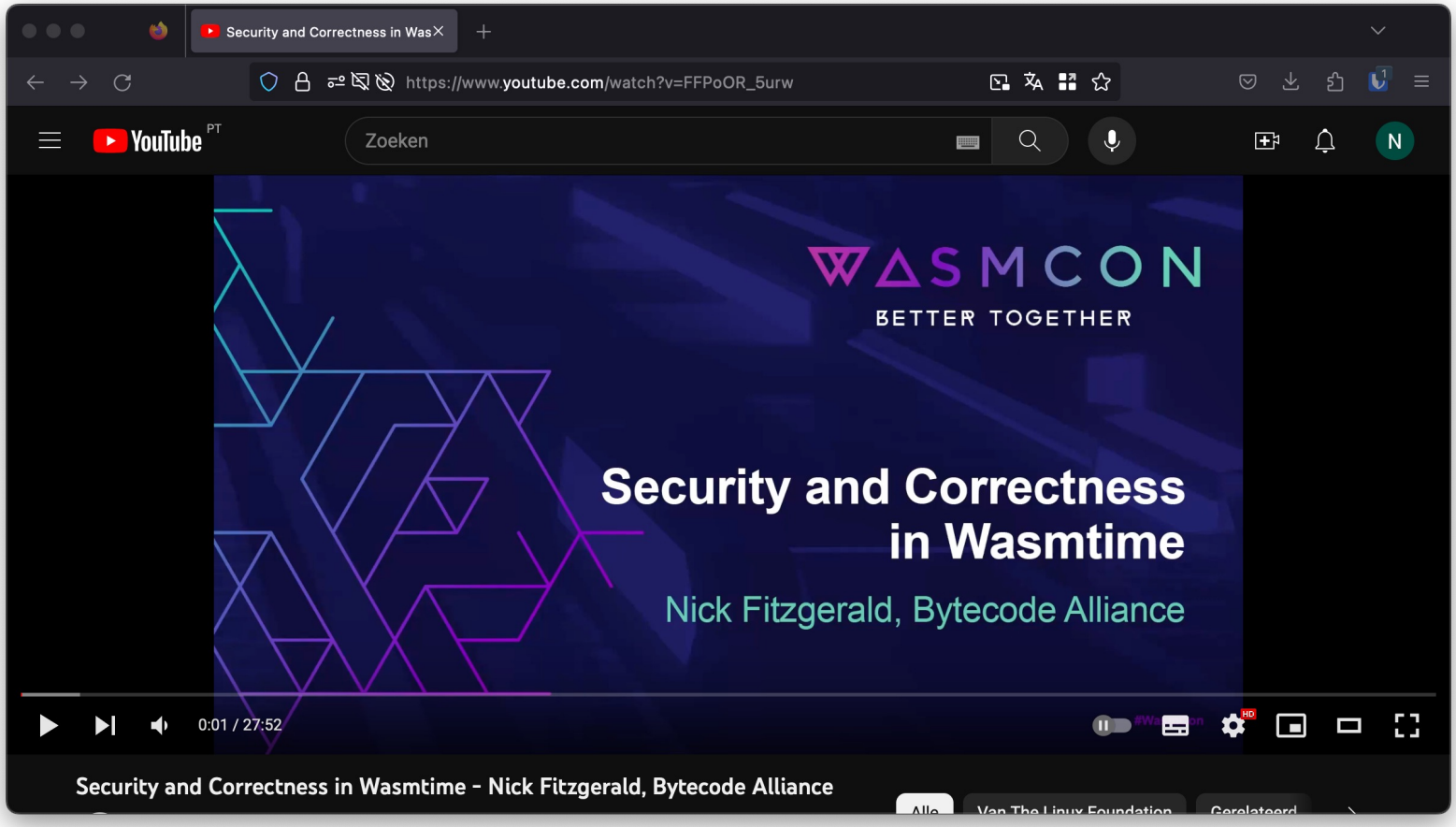


# Runtimes and Security

- 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



# 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/fermyonstream2024
- ntanis at Veracode.com
- @nielstanis@infosec.exchange
- https://blog.fennec.dev
- Bedankt! Thank you!