



Using WebAssembly to run, extend, and secure your application

Niels Tanis
Security Researcher



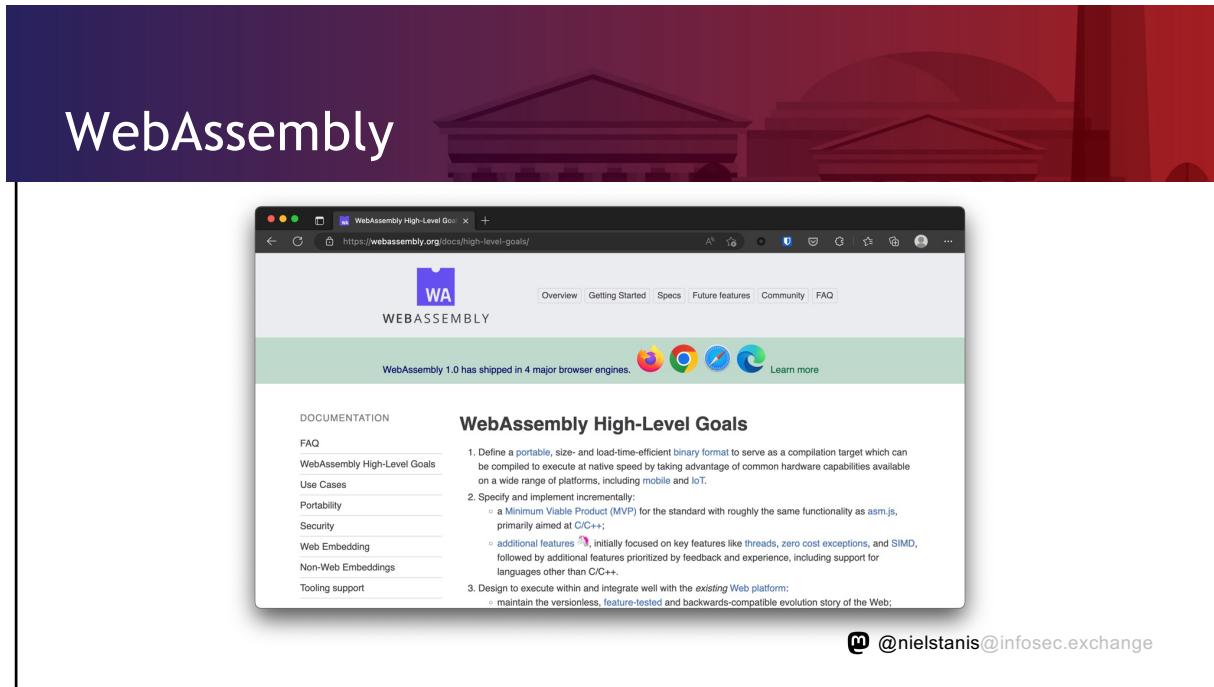
Who am I?

- 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

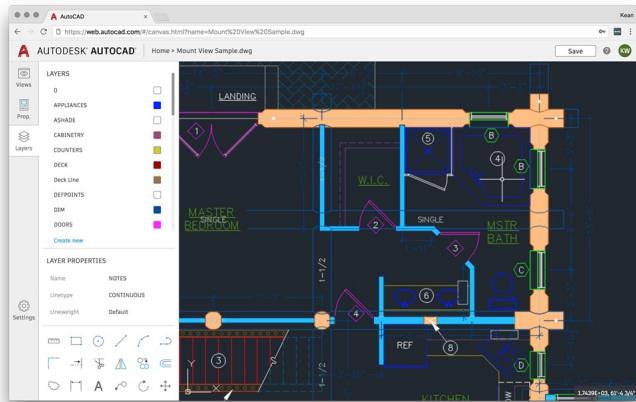
VERACODE



✉️ @nielstanis@infosec.exchange

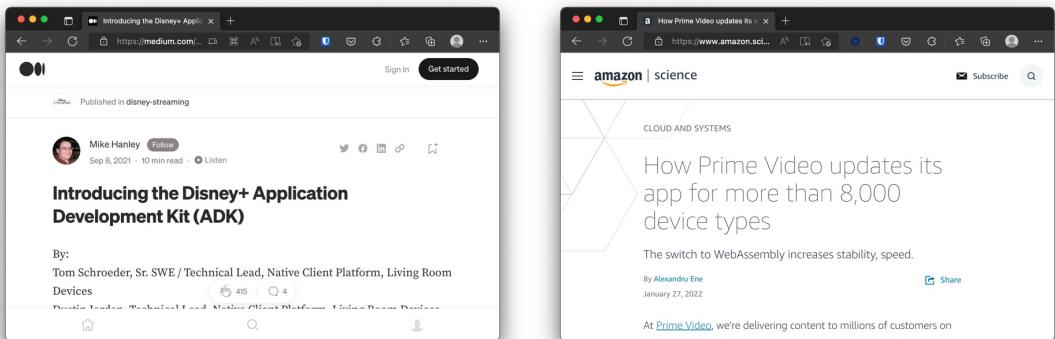
<https://hacks.mozilla.org/files/2019/08/04-01-star-diagram.png>

WebAssembly - AutoCAD



@nielstanis@infosec.exchange

WebAssembly - SDK's



 @nielstanis@infosec.exchange

<https://medium.com/disney-streaming/introducing-the-disney-application-development-kit-adk-ad85ca139073>

<https://www.amazon.science/blog/how-prime-video-updates-its-app-for-more-than-8-000-device-types>

Agenda

- Introduction
- WebAssembly 101
- Using WebAssembly to ...
 - ... run your application
 - ... extend your application
 - ... secure your application
- Conclusion
- Q&A

 @nielstanis@infosec.exchange

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

 @nielstanis@infosec.exchange

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



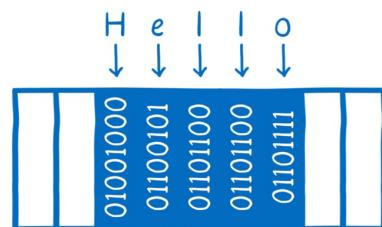
WEBASSEMBLY

 @nielstanis@infosec.exchange

<https://hacks.mozilla.org/2017/02/creating-and-working-with-webassembly-modules/>
<https://webassembly.org/>

WebAssembly Memory

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



 @nielstanis@infosec.exchange

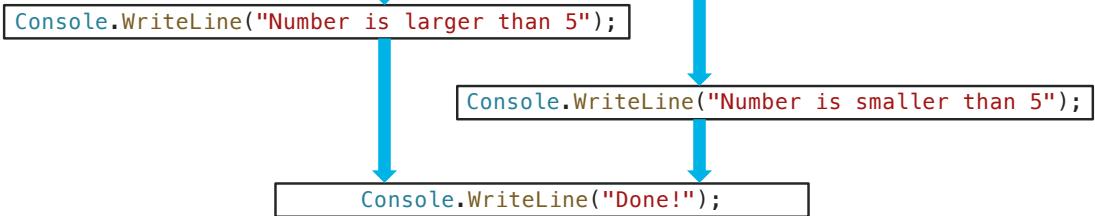
WebAssembly Control-Flow Integrity

```
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!");
```

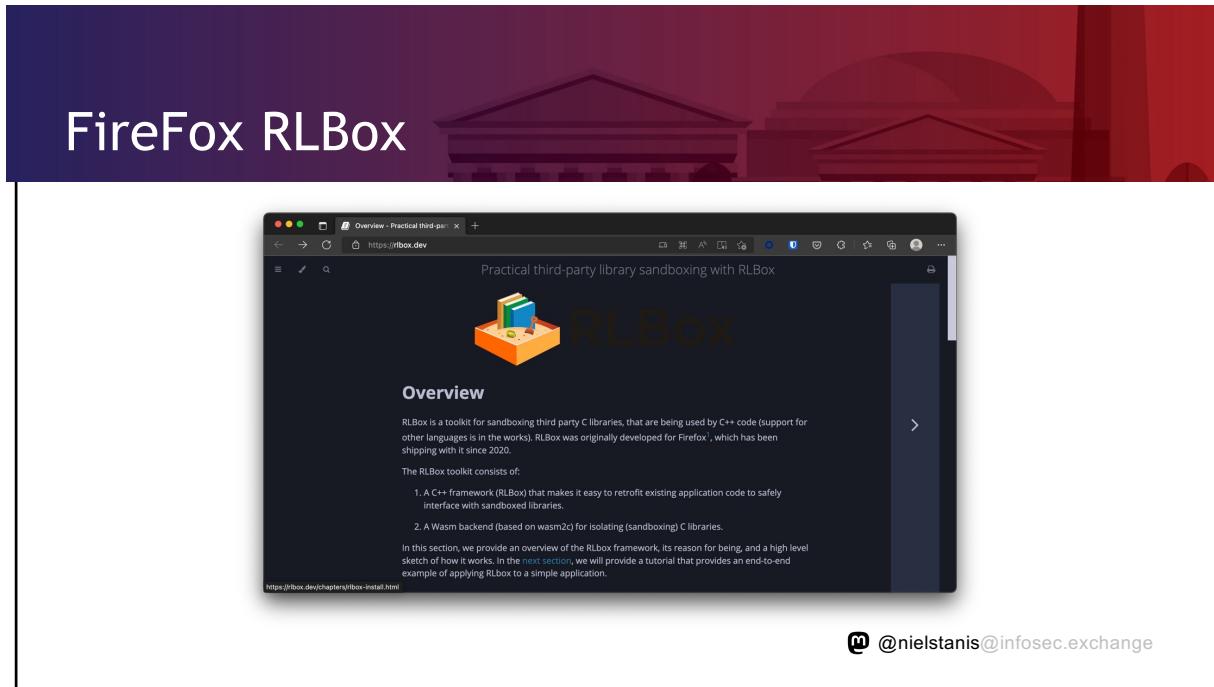
 @nielstanis@infosec.exchange

WebAssembly Control-Flow Integrity

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



@nielstanis@infosec.exchange

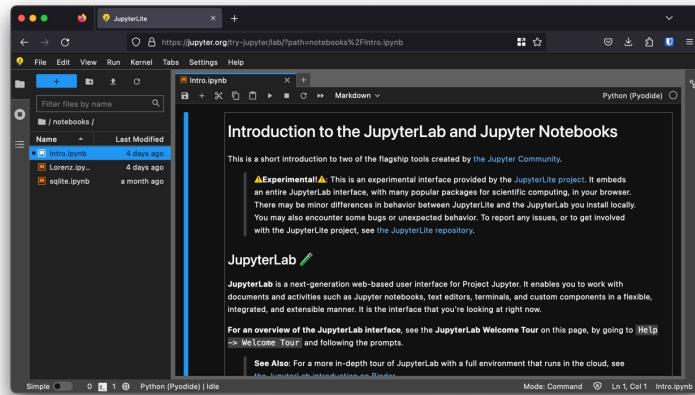


<https://rlbox.dev/>

<https://hacks.mozilla.org/2020/02/securing-firefox-with-webassembly/>

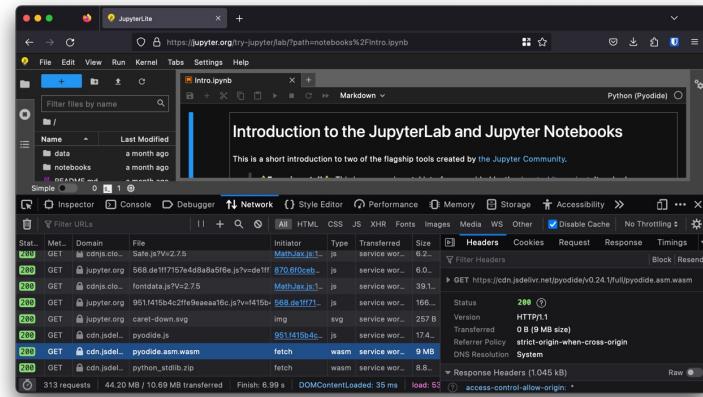
 @nielstanis@infosec.exchange

Python on WASM



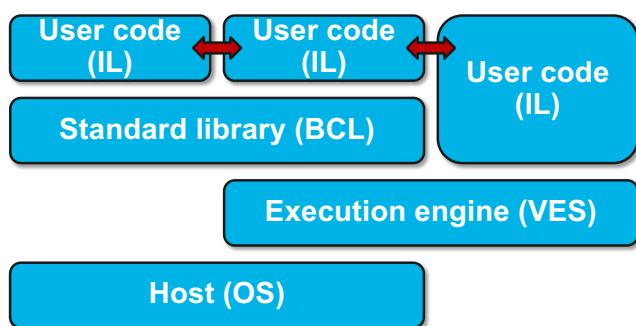
 @nielstanis@infosec.exchange

Python on WASM



 @nielstanis@infosec.exchange

Running .NET on WebAssembly

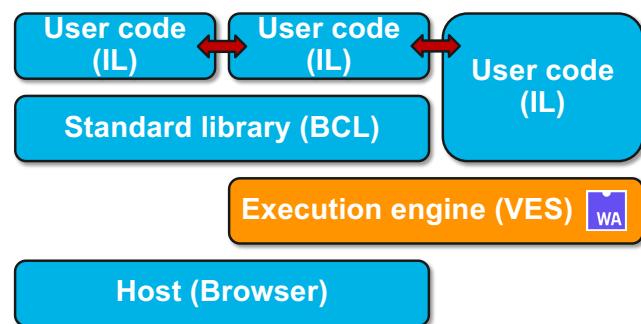


@nielstanis@infosec.exchange

Diagram:

<https://github.com/itowlson/wasmday22/blob/main/slides/Wasm%20Interfaces%20and%20.NET.pptx>

Running .NET on WebAssembly

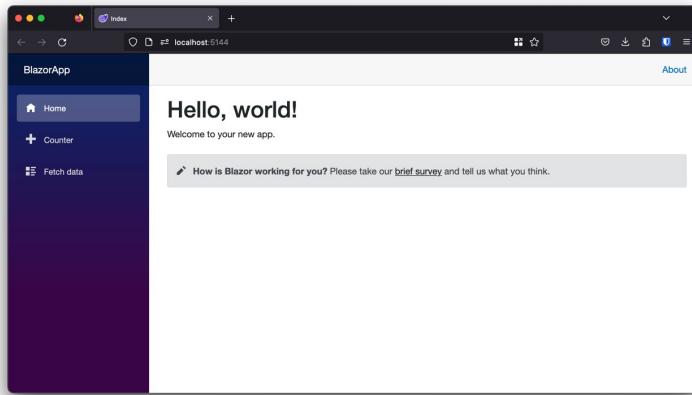


👤 @nielstanis@infosec.exchange

Diagram:

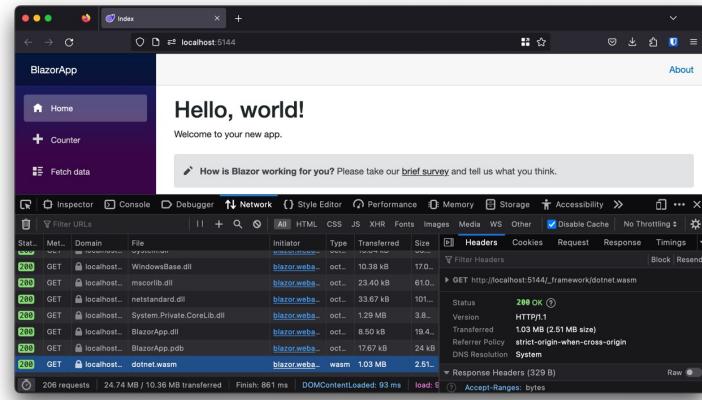
<https://github.com/itowlson/wasmday22/blob/main/slides/Wasm%20Interfaces%20and%20.NET.pptx>

Blazor WebAssembly



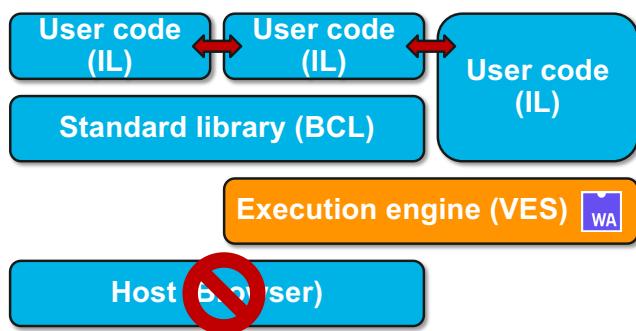
 @nielstanis@infosec.exchange

Blazor WebAssembly



 @nielstanis@infosec.exchange

Running .NET on WebAssembly



@nielstanis@infosec.exchange

Diagram:

<https://github.com/itowlson/wasmday22/blob/main/slides/Wasm%20Interfaces%20and%20.NET.pptx>

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

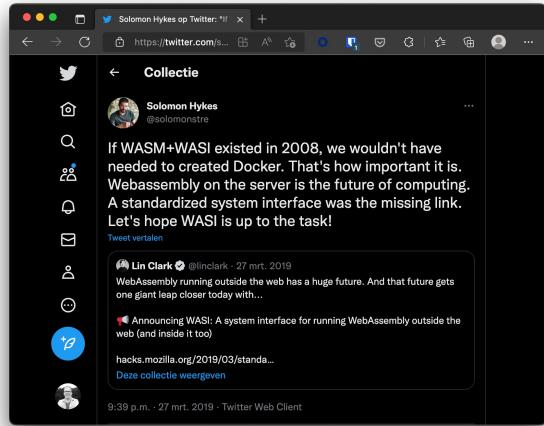
 @nielstanis@infosec.exchange

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 familiar with .NET Standard? 😊

 @nielstanis@infosec.exchange

Docker vs WASM & WASI



 @nielstanis@infosec.exchange

Docker vs WASM & WASI



 @nielstanis@infosec.exchange

Docker & WASM

The image displays two side-by-side screenshots of a Docker+Wasm Technical Preview landing page. The left screenshot shows the main introduction with the title "Introducing the Docker+Wasm Technical Preview" and a photo of Michael Irwin. The right screenshot shows a detailed diagram of the Docker Engine architecture, illustrating how a Wasm module is integrated into the container structure.

Left Screenshot Content:

- Title:** Introducing the Docker+Wasm Technical Preview
- Author:** MICHAEL IRWIN
- Date:** Oct 24 2022
- Text:** 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.

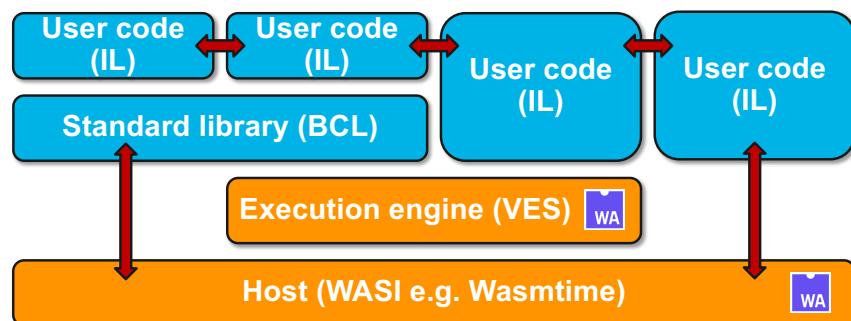
Right Screenshot Content:

- Diagram Description:** The diagram illustrates the Docker Engine architecture. At the top is the "Docker Engine". Below it is a box labeled "containerd". Inside "containerd" are three boxes: "containerd-shim", "containerd-shim", and "containerd-wasm shim". Each "containerd-shim" box contains a "runc" box, which in turn contains a "Container process". The "containerd-wasm shim" box contains a "wasmedge" box, which in turn contains a "Wasm Module".
- Text:** Let's look at an example!
- Text:** 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
```

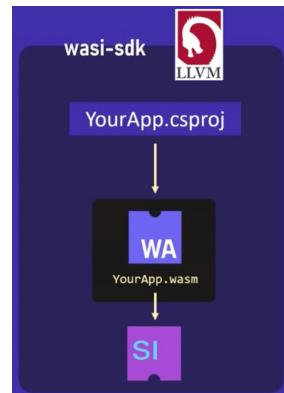
@nielstanis@infosec.exchange

WebAssembly System Interface WASI



@nielstanis@infosec.exchange

Experimental WASI SDK for .NET



@nielstanis@infosec.exchange

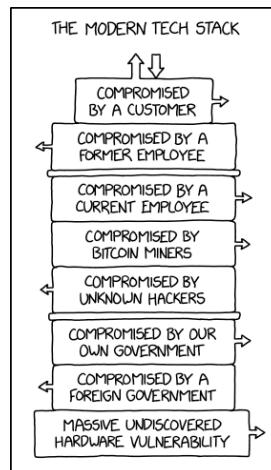
<https://github.com/SteveSandersonMS/dotnet-wasi-sdk>

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!

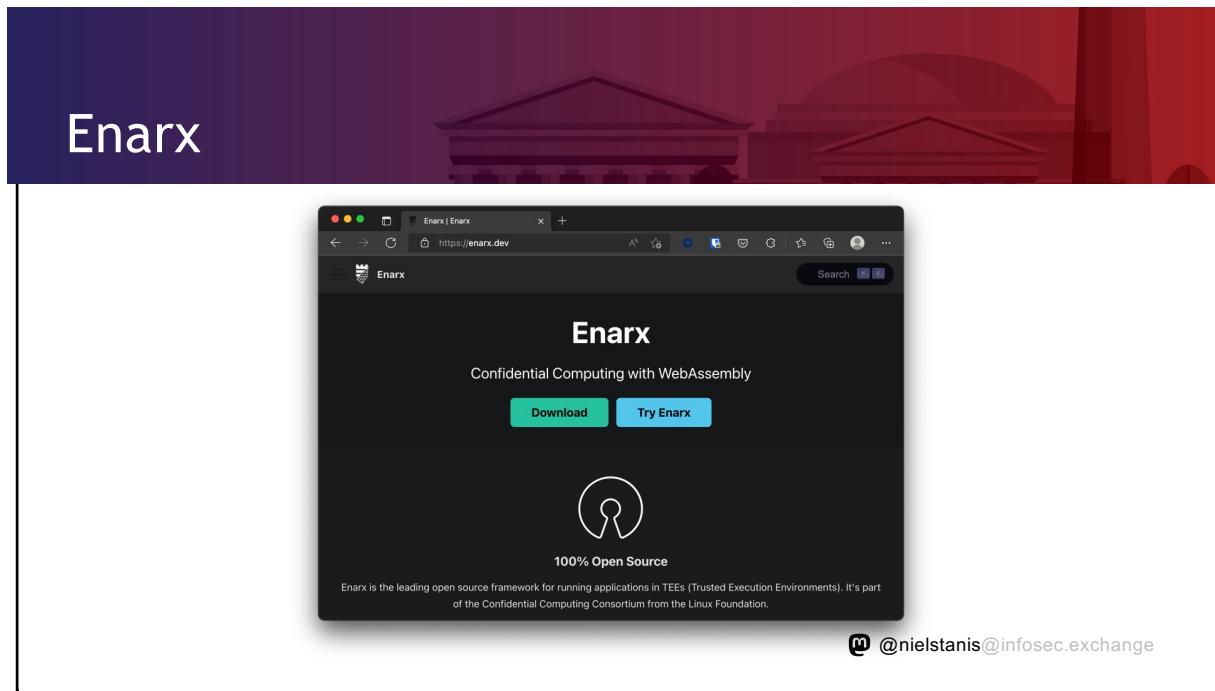
 @nielstanis@infosec.exchange

Trusted Computing - XKCD 2166



@nielstanis@infosec.exchange

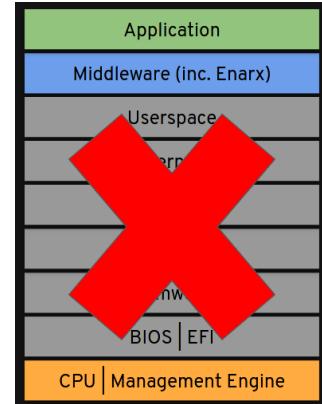
<https://xkcd.com/2166/>



<https://enarx.dev/>

Enarx Threat Model

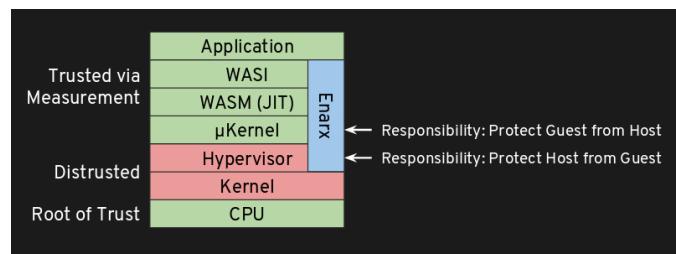
- Don't trust the host
- Don't trust the host owner
- Don't trust the host operator
- Hardware cryptographically verified
- Software audited and cryptographically verified



 @nielstanis@infosec.exchange

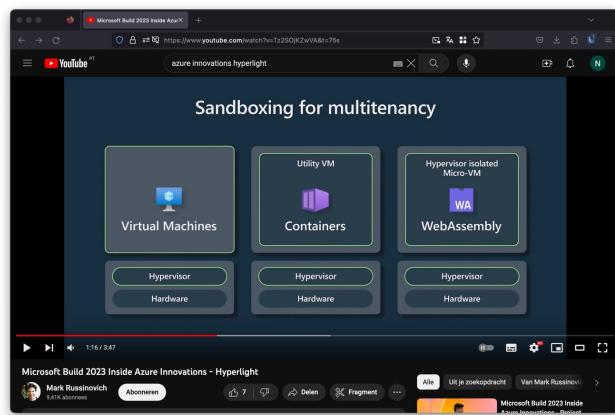
Enarx

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



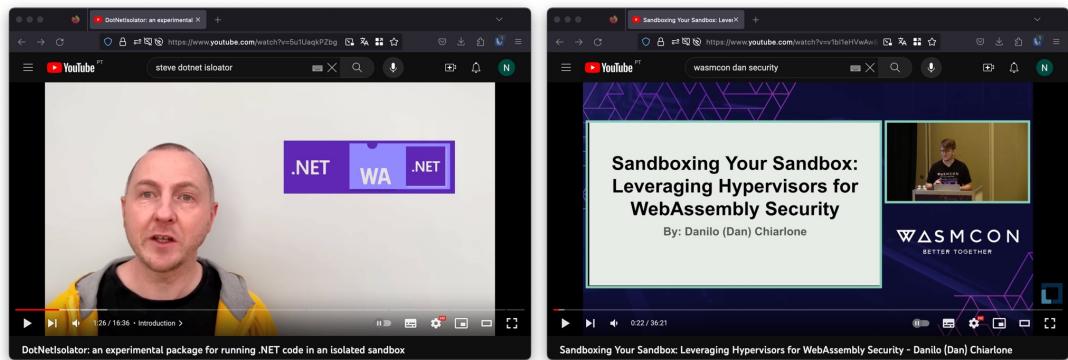
@nielstanis@infosec.exchange

Project Hyperlight



 @nielstanis@infosec.exchange

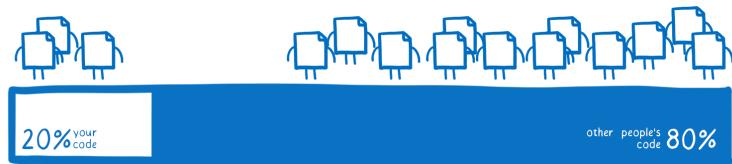
DotNetIsolator & Project Hyperlight



 @nielstanis@infosec.exchange

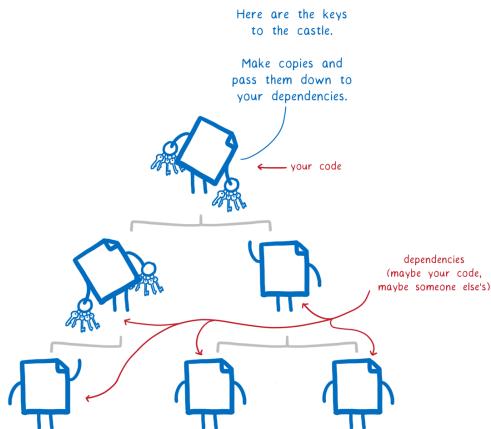
WASM - What's next?

composition of an
average code base



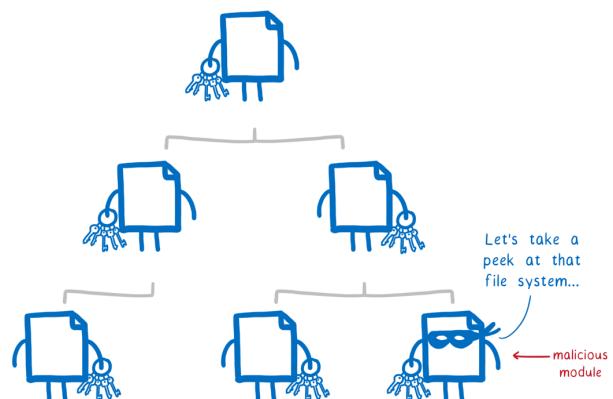
 @nielstanis@infosec.exchange

Dependencies



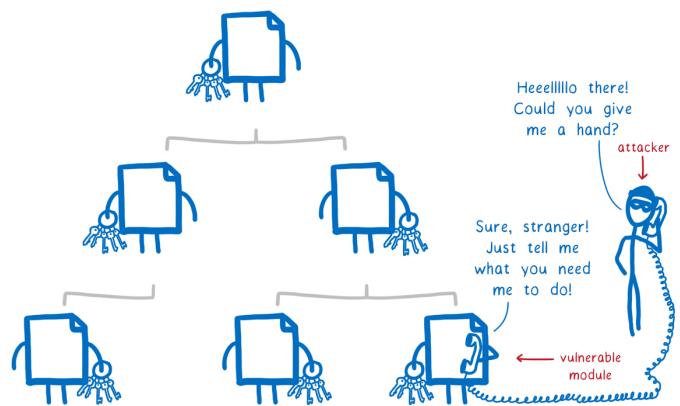
 @nielstanis@infosec.exchange

Malicious module



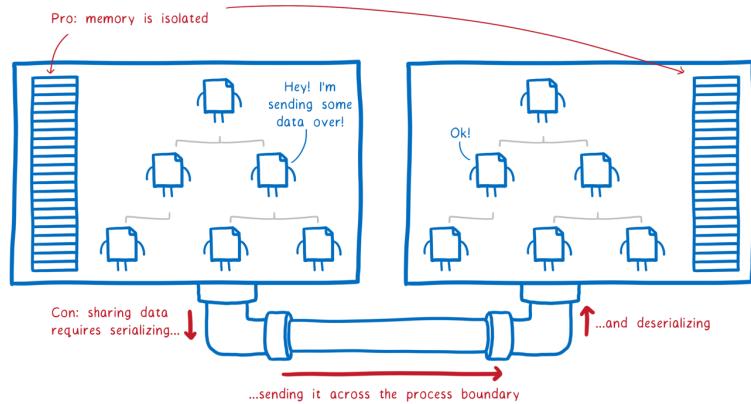
 @nielstanis@infosec.exchange

Vulnerable module



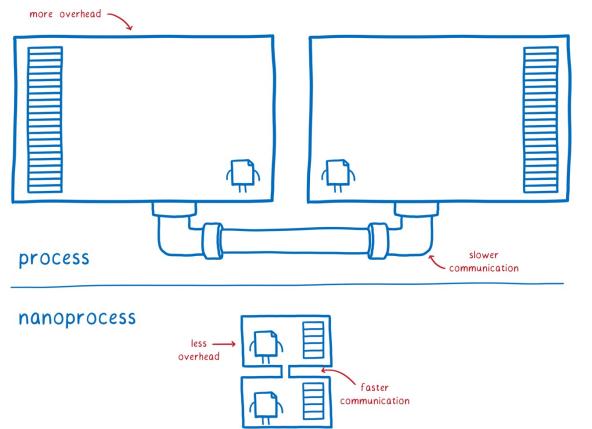
 @nielstanis@infosec.exchange

Process Isolation



 @nielstanis@infosec.exchange

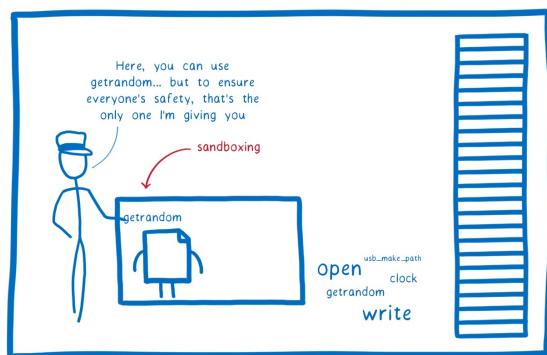
WebAssembly Nano-Process



* not drawn to scale
m @nielstanis@infosec.exchange

WebAssembly Nano-Process

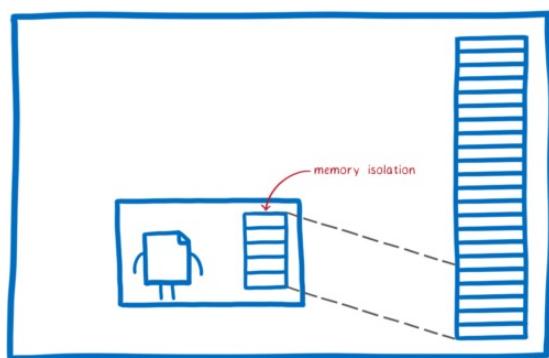
1. Sandboxing



 @nielstanis@infosec.exchange

WebAssembly Nano-Process

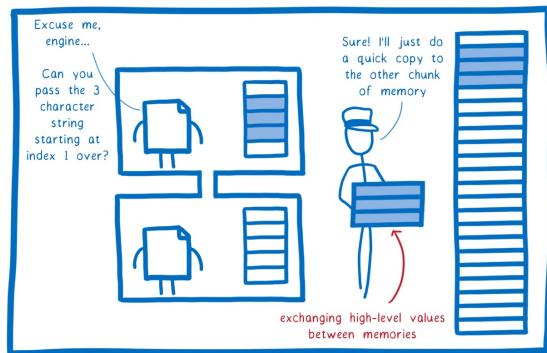
2. Memory model



 @nielstanis@infosec.exchange

WebAssembly Nano-Process

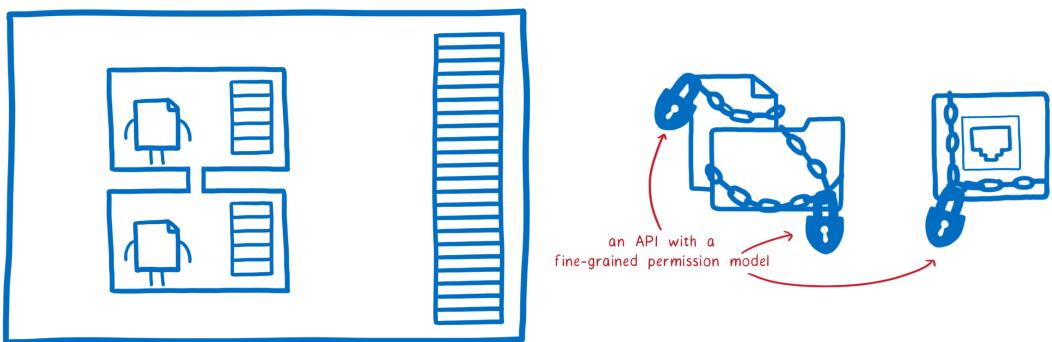
3. Interface Types



 @nielstanis@infosec.exchange

WebAssembly Nano-Process

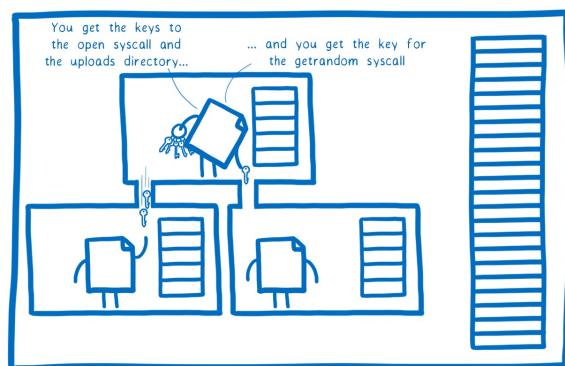
4. WebAssembly System Interface



 @nielstanis@infosec.exchange

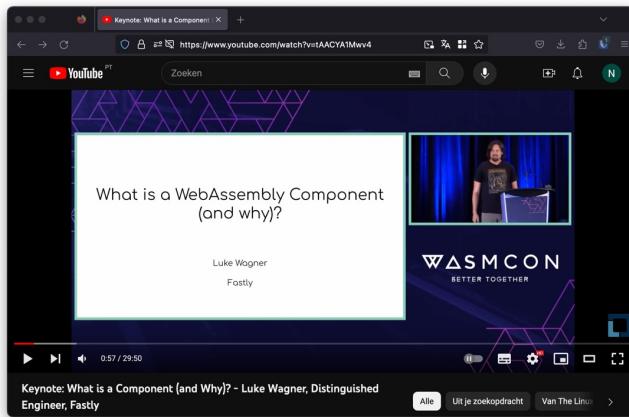
WebAssembly Nano-Process

5. The missing link



 @nielstanis@infosec.exchange

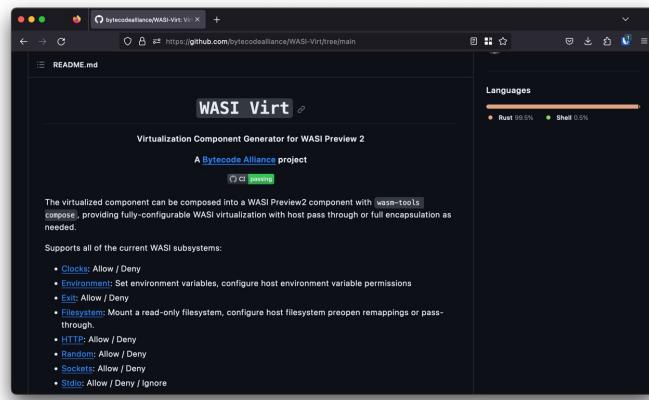
WebAssembly Component Model



 @nielstanis@infosec.exchange

<https://www.youtube.com/watch?v=tAACYA1Mwv4>

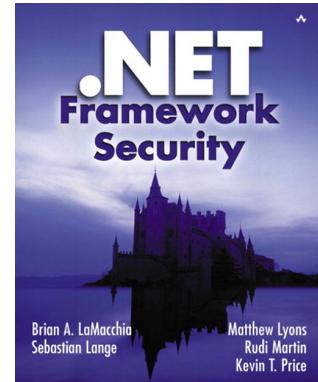
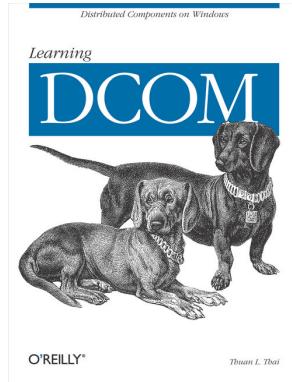
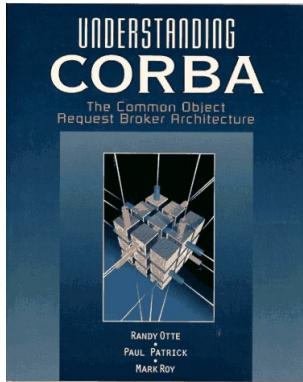
WebAssembly Component Model



 @nielstanis@infosec.exchange

<https://www.youtube.com/watch?v=tAACYA1Mwv4>

Have we seen this before?



 @nielstanis@infosec.exchange

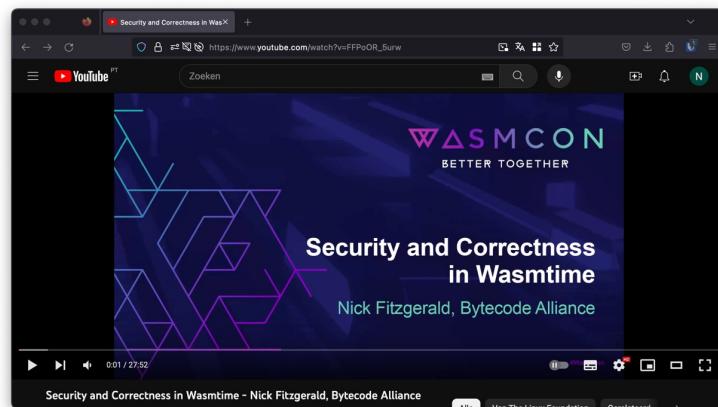
Runtimes and Security

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

 @nielstanis@infosec.exchange

<https://bytecodealliance.org/articles/security-and-correctness-in-wasmtime>

Runtimes and Security



 @nielstanis@infosec.exchange

https://www.youtube.com/watch?v=FFPoOR_5urw

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!
- I like top-down approach Bytecode Alliance is taking in moving forward, feedback will change/influence

 @nielstanis@infosec.exchange

Questions?

- <https://github.com/nielstanis/appsecdc2023>
- ntanis at Veracode.com
- @nielstanis@infosec.exchange
- <https://blog.fennec.dev>
- Thank you!

 @nielstanis@infosec.exchange



THANK YOU