

NDC { Porto }

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

Niels Tanis



0101
0101

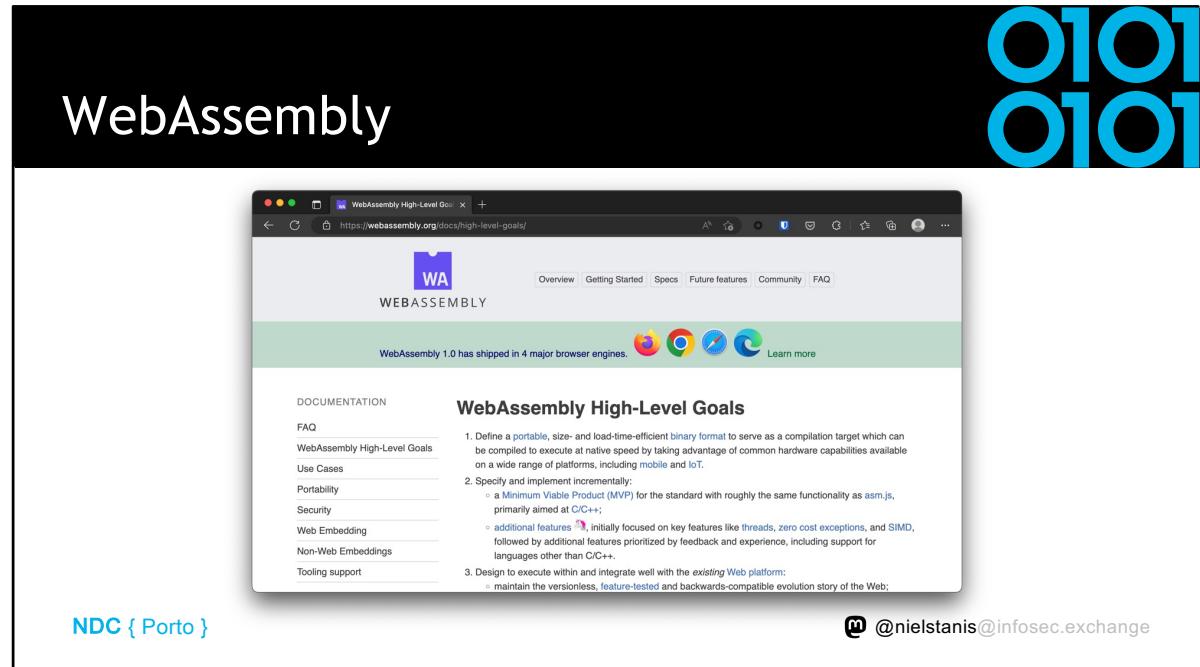
Who am I?

- 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



NDC { Porto }

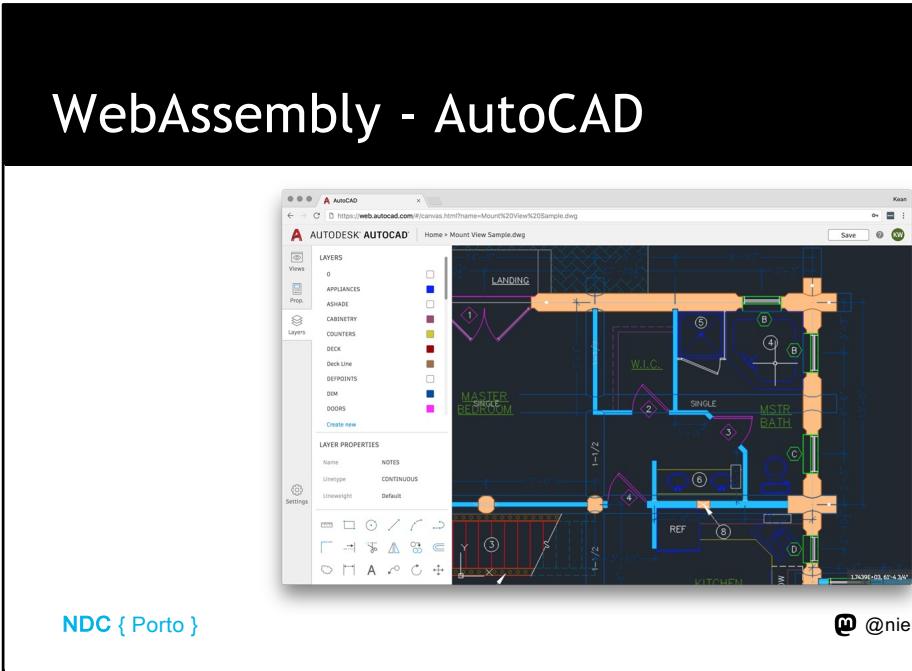
 @nielstanis@infosec.exchange



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

0101
0101

WebAssembly - AutoCAD

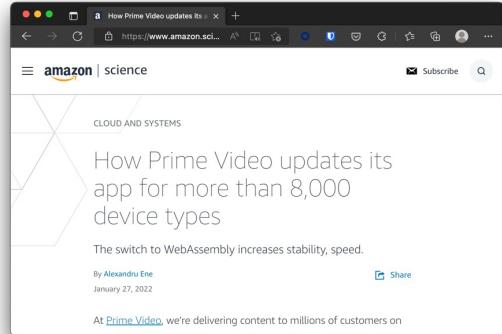
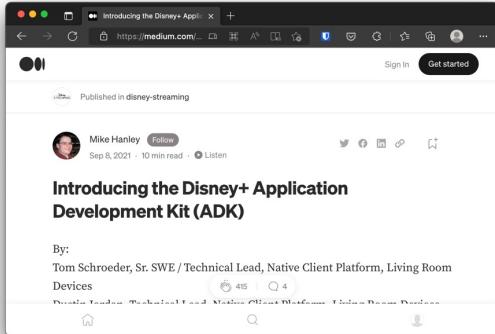


NDC { Porto }

@nielstanis@infosec.exchange

0101
0101

WebAssembly - SDK's

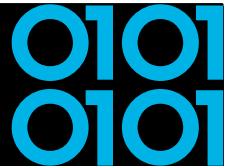


NDC { Porto }

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

NDC { Porto }

@nielstanis@infosec.exchange

WebAssembly Design



WEBASSEMBLY

- **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

NDC { Porto }

@nielstanis@infosec.exchange

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

NDC { Porto }

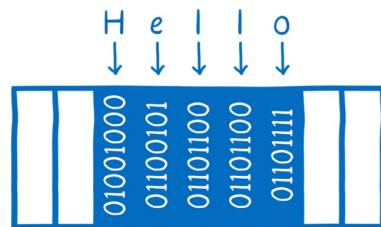
@nielstanis@infosec.exchange

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



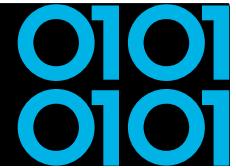
WebAssembly Memory

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



NDC { Porto }

@nielstanis@infosec.exchange

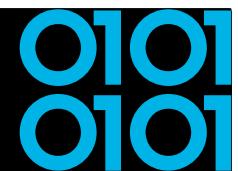


WebAssembly Control-Flow Integrity

```
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!");
```

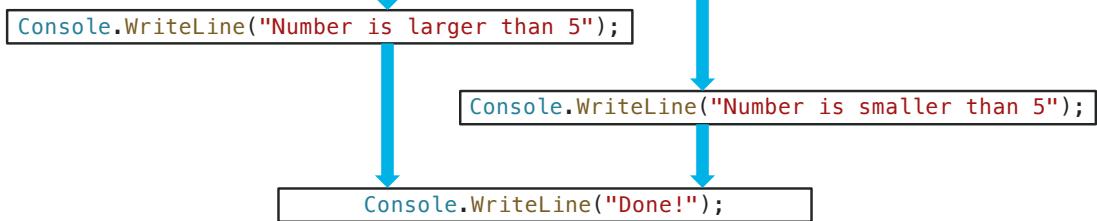
NDC { Porto }

 @nielstanis@infosec.exchange



WebAssembly Control-Flow Integrity

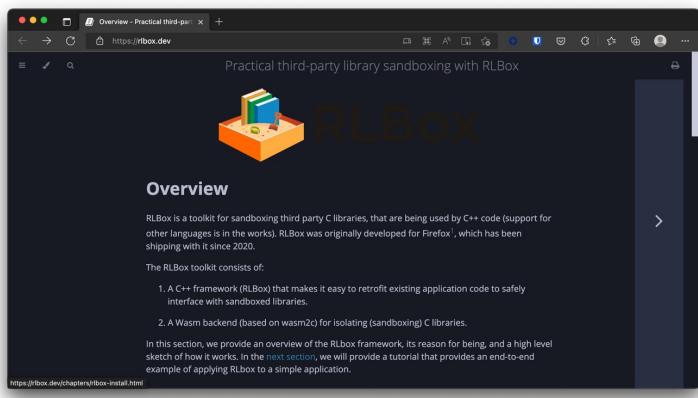
```
int number = Convert.ToInt32(Console.ReadLine());
Console.WriteLine($"Number {number}");
if (number>5)
```



NDC { Porto }

@nielstanis@infosec.exchange

FireFox RLBox



NDC { Porto }

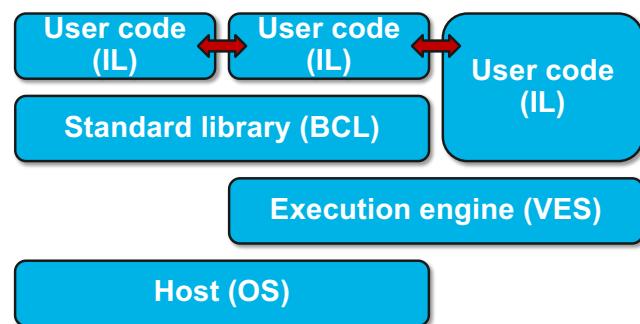
 @nielstanis@infosec.exchange

<https://rlbox.dev/>

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

Running .NET on WebAssembly

0101
0101



NDC { Porto }

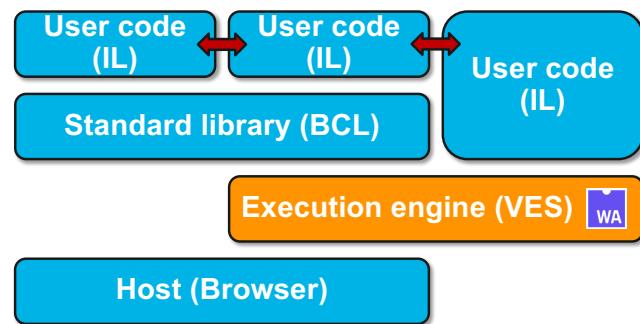
@nielstanis@infosec.exchange

Diagram:

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

Running .NET on WebAssembly

0101
0101



NDC { Porto }

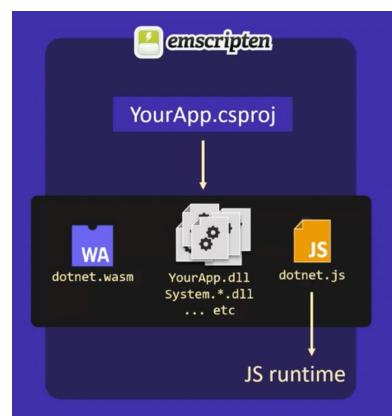
👤 @nielstanis@infosec.exchange

Diagram:

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

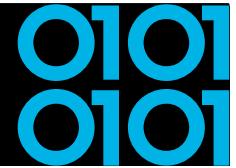
0101
0101

Blazor WebAssembly

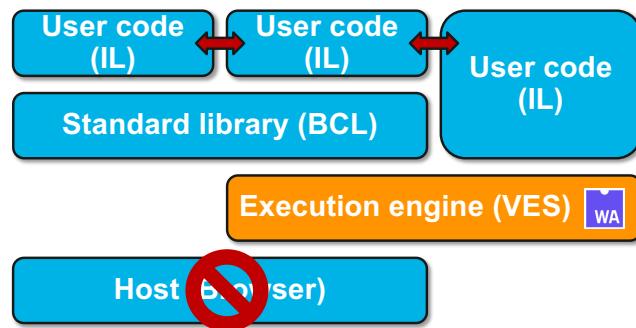


NDC { Porto }

@nielstanis@infosec.exchange



Running .NET on WebAssembly



NDC { Porto }

👤 @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

NDC { Porto }

 @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 recall .NET Standard? 😊

NDC { Porto }

 @nielstanis@infosec.exchange

0101
0101

Docker vs WASM & WASI

The screenshot shows a Twitter web client interface. On the left is a sidebar with icons for Twitter, Home, Search, Direct Messages, Notifications, and Profile. The main area displays a tweet from Solomon Hykes (@solomonstre) with the following content:

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!

Below this tweet is a reply from Lin Clark (@linclark) dated 27 mrt. 2019:

WebAssembly running outside the web has a huge future. And that future gets one giant leap closer today with...

Announcing WASI: A system interface for running WebAssembly outside the web (and inside it too)
hacks.mozilla.org/2019/03/standards/

At the bottom of the tweet card, it says "D deze collectie weergeven".

At the bottom right of the main window, there is a small profile icon and the handle @nielstanis@infosec.exchange.

NDC { Porto }

0101 0101

Docker vs WASM & WASI



NDC { Porto }

 @nielstanis@infosec.exchange

0101
0101

Docker & WASM

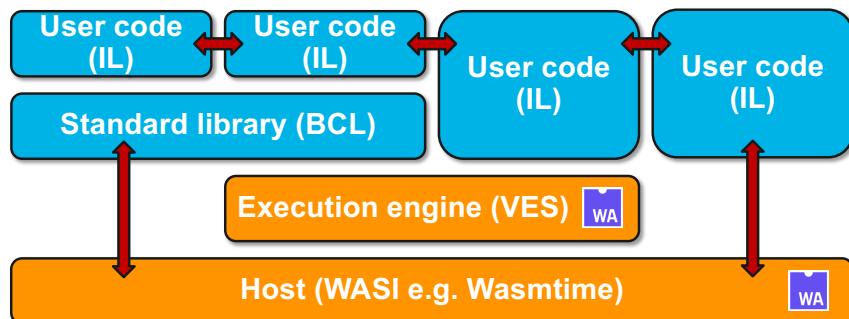
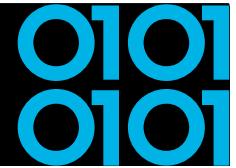
The image contains two side-by-side screenshots of the Docker website. The left screenshot shows a landing page for 'Introducing the Docker+Wasm Technical Preview' by Michael Irwin, dated Oct 24 2022. It features a diagram showing the Docker Engine at the top, followed by a 'containerd' box which branches into three separate 'shim' boxes: 'containerd-shim', 'containerd-shim', and 'containerd-wasm shim'. Each 'shim' box contains a 'runc' box, which in turn contains a 'Container process'. The right screenshot shows a similar diagram but with a 'WasmModule' box instead of 'Container process' under the 'containerd-wasm shim' box. Below both diagrams is a section titled 'Let's look at an example!' with a command-line snippet:

```
docker run -dp 8080:8080 --name=wasm-example --runtime=io.containerd.wasmedge.v1 --platform=wasi/wasm32 michaelirvin244/wasm-example
```

NDC { Porto }

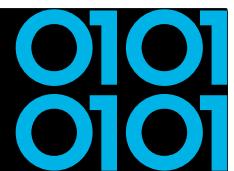
@nielstanis@infosec.exchange

WebAssembly System Interface WASI

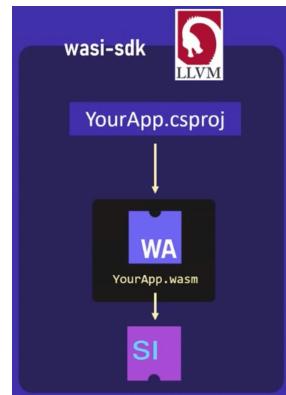


NDC { Porto }

@nielstanis@infosec.exchange



Experimental WASI SDK for .NET

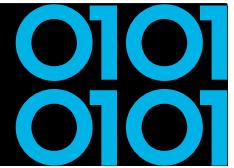


NDC { Porto }

@nielstanis@infosec.exchange

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

Experimental WASI SDK for .NET



WASI support tracking #65895

SteveSandersonMS commented on Feb 26, 2022 · edited by pavelavarva

Description

This issue is to track known issues in the early WASI-enabled runtime builds. These need to be resolved in order to have a proper portable WASI-ready release.

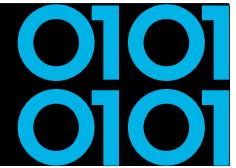
- Add wasi-wasi RID
- Add new RID for WASI #77790
 - (API Process): Report undefined `Value` #K78380 API change
- Build `System.Native.wasi` using WASI SDK instead of using the Emscripten-built binary.
 - MSBuild files for it and generate `wasi_wasi_Invoke.g.h`
 - Enable pinvoke for all APIs exposed by `System.Native`. Currently there's a small hardcoded list.
 - Fix [wasi] compile with `libs.native` #79046

NDC { Porto }

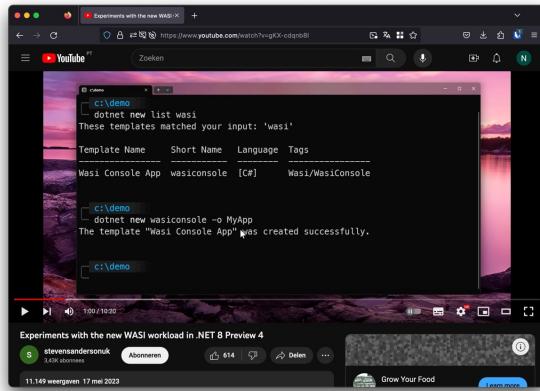
@nielstanis@infosec.exchange

<https://github.com/dotnet/runtime/issues/65895>

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



Experimental WASI workload .NET8

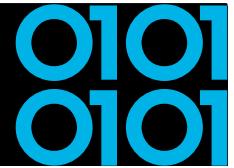


NDC { Porto }

@nielstanis@infosec.exchange

<https://github.com/dotnet/runtime/issues/65895>

<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
- Demo time!

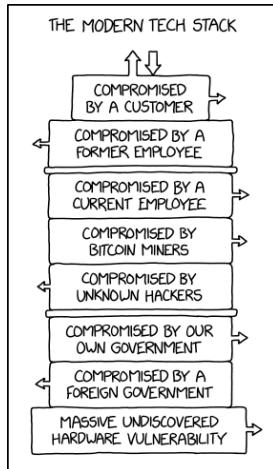
NDC { Porto }

@nielstanis@infosec.exchange

0101
0101

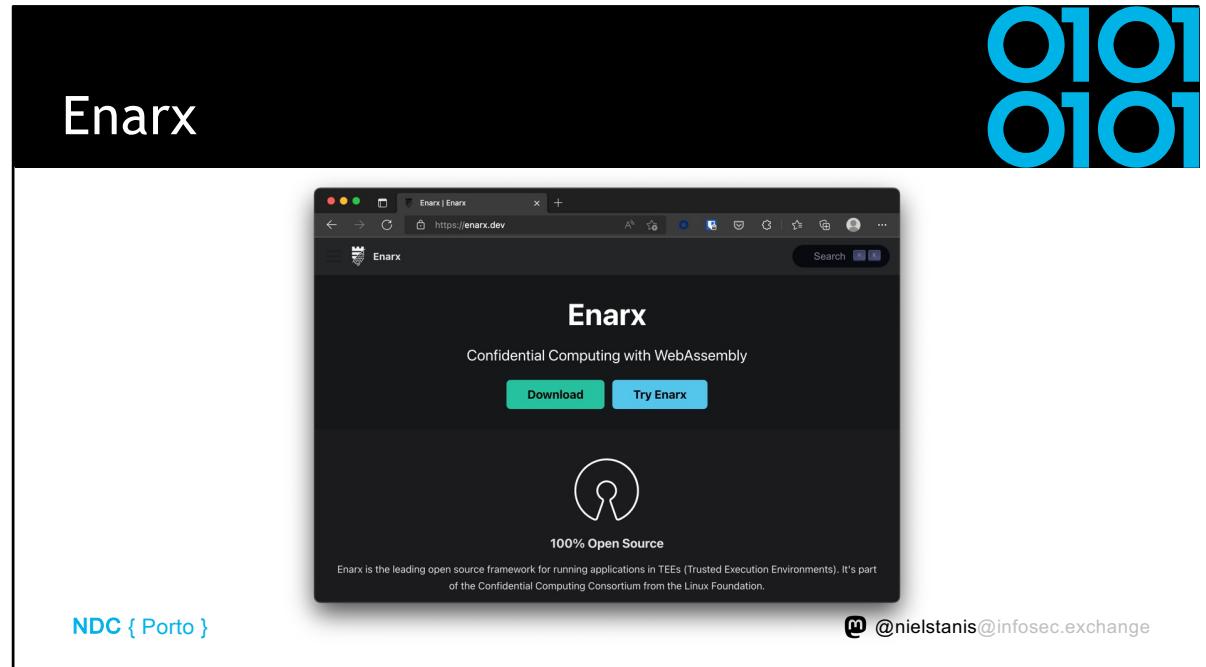
Trusted Computing - XKCD 2166

NDC { Porto }



 @nielstanis@infosec.exchange

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

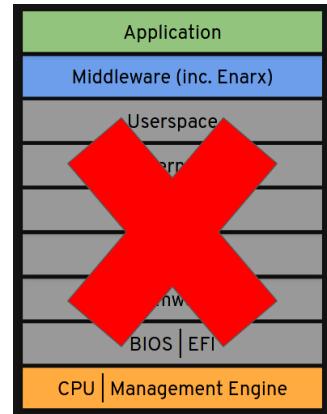


<https://enarx.dev/>

0101
0101

Enarx Threat Model

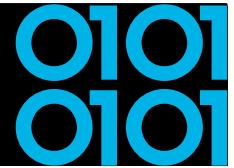
- Don't trust the host
- Don't trust the host owner
- Don't trust the host operator
- Hardware cryptographically verified
- Software audited and cryptographically verified



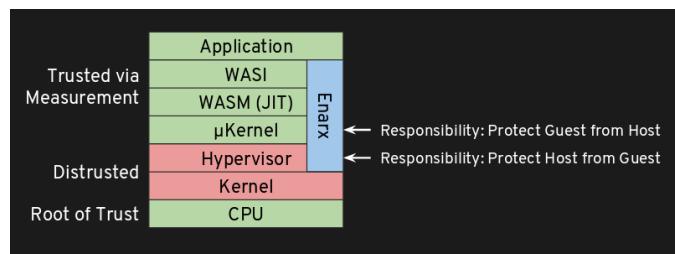
NDC { Porto }

 @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

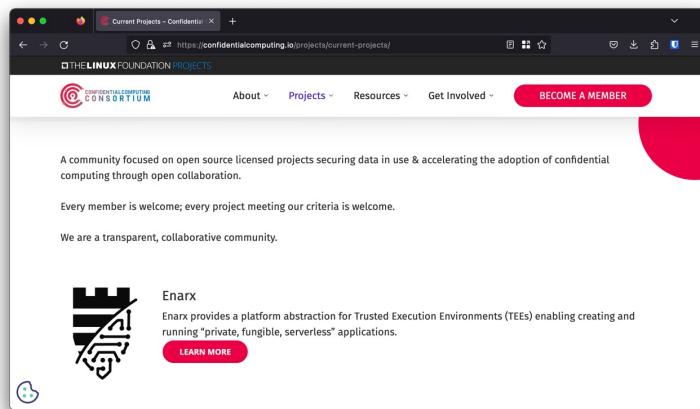


NDC { Porto }

@nielstanis@infosec.exchange

Confidential Computing Consortium

0101
0101

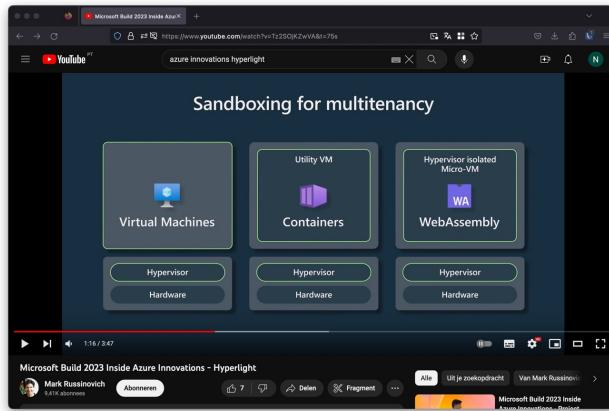


NDC { Porto }

 @nielstanis@infosec.exchange

0101
0101

Project Hyperlight

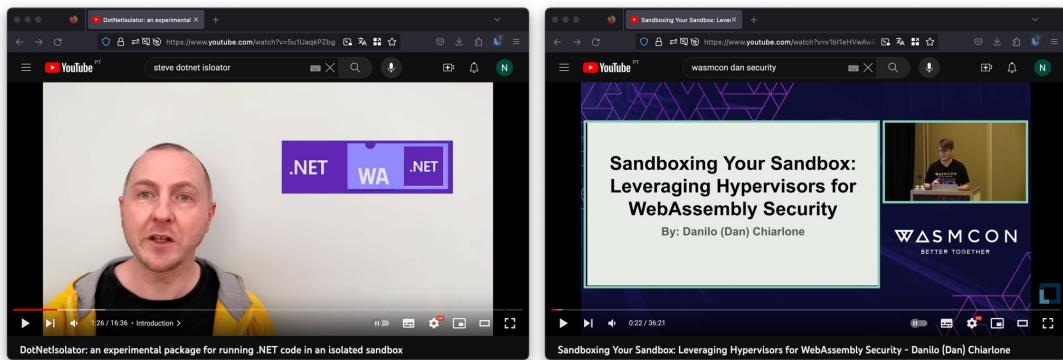


NDC { Porto }

 @nielstanis@infosec.exchange

DotNetIsolator & Project Hyperlight

0101
0101



NDC { Porto }

 @nielstanis@infosec.exchange

0101
0101

WASM - What's next?

composition of an
average code base



NDC { Porto }

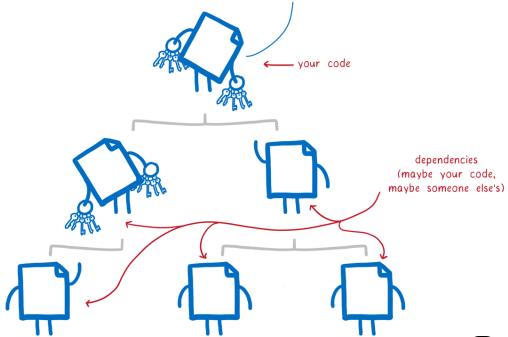
 @nielstanis@infosec.exchange

0101
0101

Dependencies

Here are the keys
to the castle.

Make copies and
pass them down to
your dependencies.

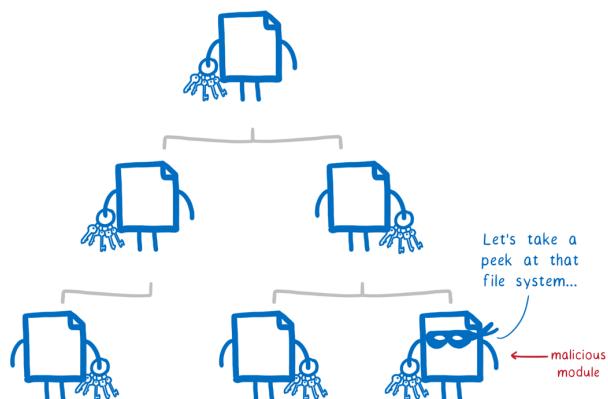


NDC { Porto }

 @nielstanis@infosec.exchange

0101
0101

Malicious module

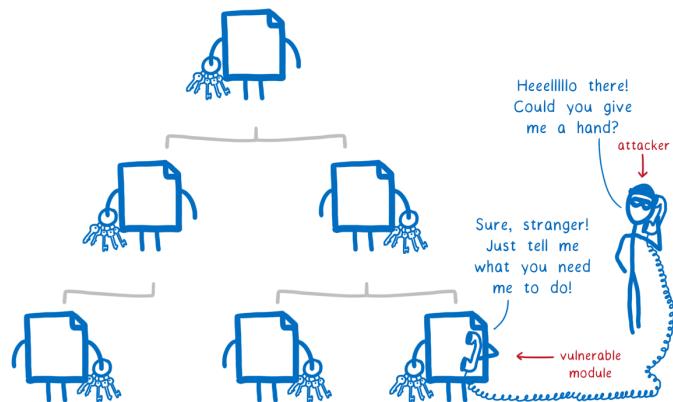


NDC { Porto }

 @nielstanis@infosec.exchange

0101
0101

Vulnerable module

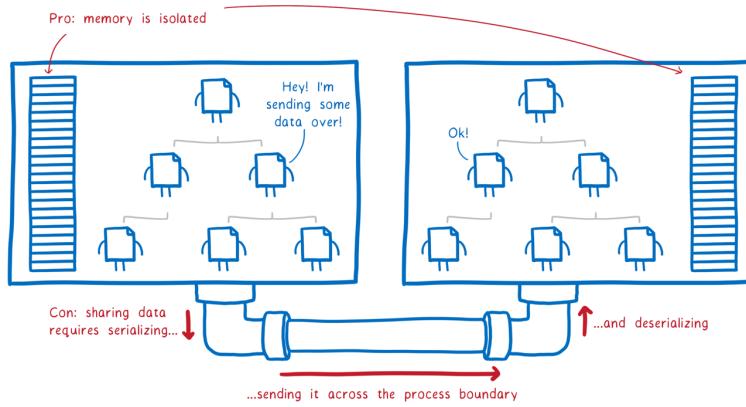


NDC { Porto }

@nielstanis@infosec.exchange

0101
0101

Process Isolation

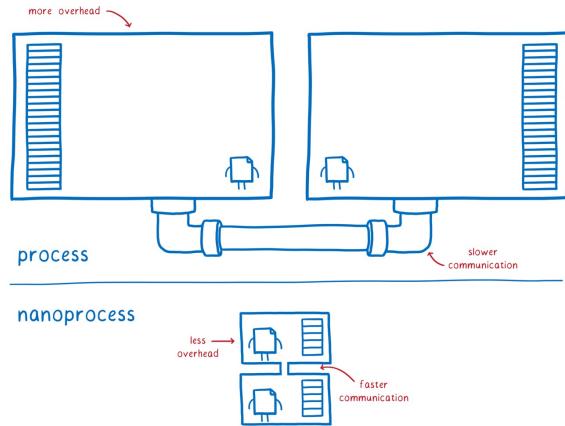


NDC { Porto }

 @nielstanis@infosec.exchange

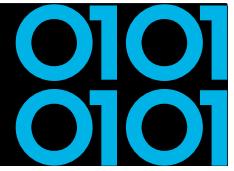
0101
0101

WebAssembly Nano-Process



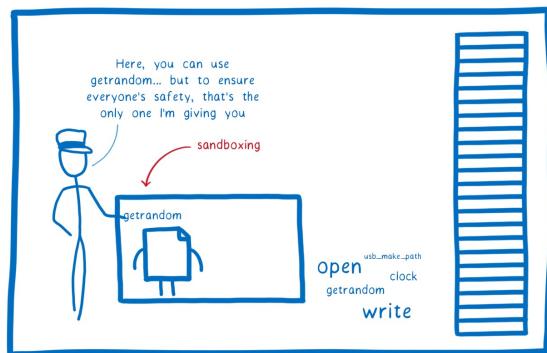
NDC { Porto }

* not drawn to scale @nielstanis@infosec.exchange



WebAssembly Nano-Process

1. Sandboxing



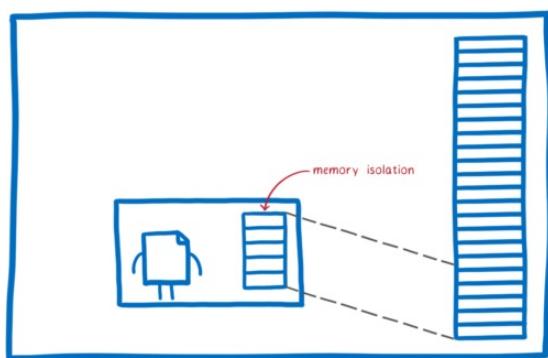
NDC { Porto }

 @nielstanis@infosec.exchange

0101
0101

WebAssembly Nano-Process

2. Memory model



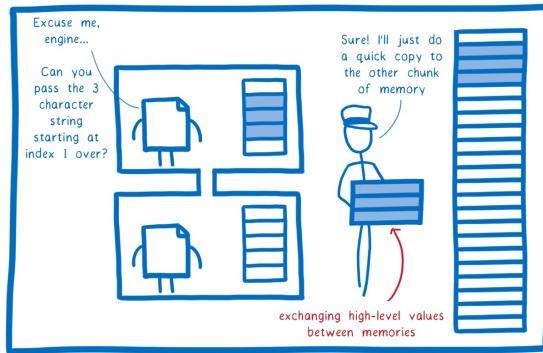
NDC { Porto }

@nielstanis@infosec.exchange

0101
0101

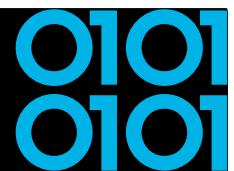
WebAssembly Nano-Process

3. Interface Types



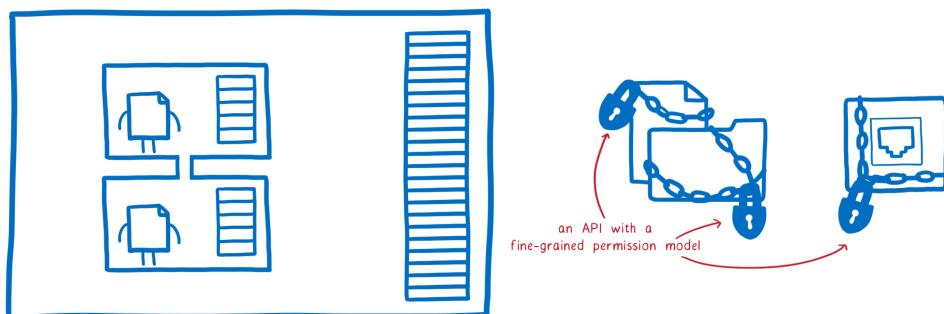
NDC { Porto }

 @nielstanis@infosec.exchange



WebAssembly Nano-Process

4. WebAssembly System Interface



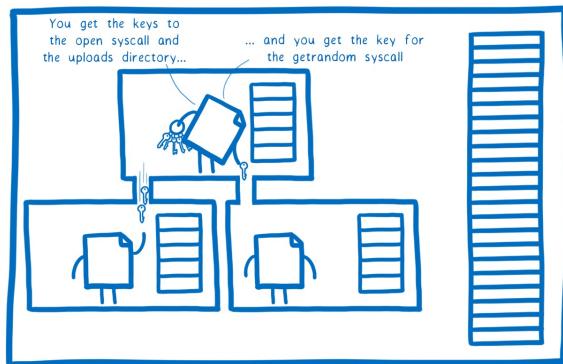
NDC { Porto }

 @nielstanis@infosec.exchange

0101
0101

WebAssembly Nano-Process

5. The missing link



NDC { Porto }

 @nielstanis@infosec.exchange

0101
0101

WebAssembly Component Model

A screenshot of a YouTube video player window. The video title is "Keynote: What is a Component? (and Why)? - Luke Wagner, Distinguished Engineer, Fasty". The video is 29:50 minutes long, currently at 0:57. The video content shows a man speaking on stage at a conference booth for "WASMCON BETTER TOGETHER". The video player interface includes a search bar, a 'Zoeken' button, and a navigation bar with back, forward, and search icons.

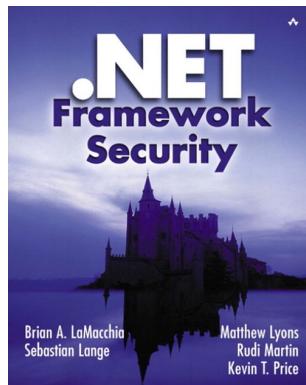
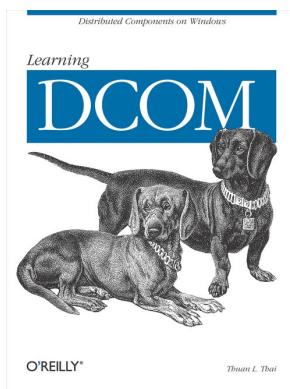
NDC { Porto }

[@nielstanis@infosec.exchange](https://www.youtube.com/watch?v=tAACYA1Mwv4)

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

0101
0101

Have we seen this before?



NDC { Porto }

 @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

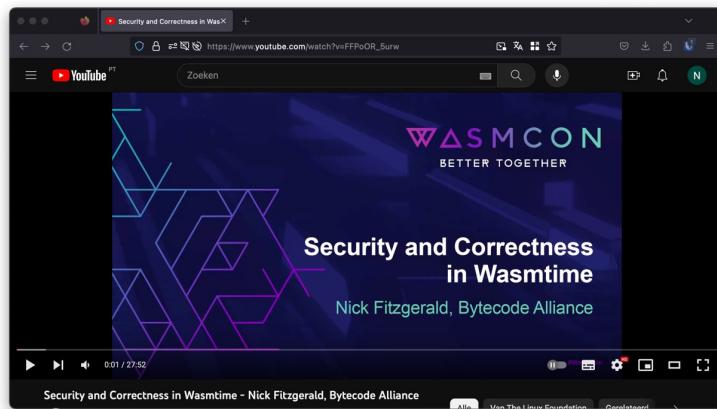
NDC { Porto }

 @nielstanis@infosec.exchange

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

Runtimes and Security

0101
0101



NDC { Porto }

 @nielstanis@infosec.exchange

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

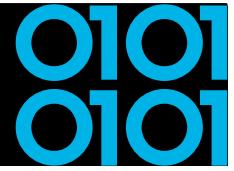


Conclusion

- Cloud Native ❤️ WebAssembly
- WebAssembly has a lot of potential to be used to run, extend, and secure your .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

NDC { Porto }

@nielstanis@infosec.exchange



Questions?

- <https://github.com/niestanis/ndcporto2023-wasm>
- niestanis at Veracode.com
- @niestanis@infosec.exchange
- <https://blog.fennec.dev>
- Obrigado! Thank you!

NDC { Porto }

@niestanis@infosec.exchange