



Using WebAssembly to run,
extend, and secure your
.NET/Java/Python/Go/...
application

Niels Tanis



O1O1
O1O1

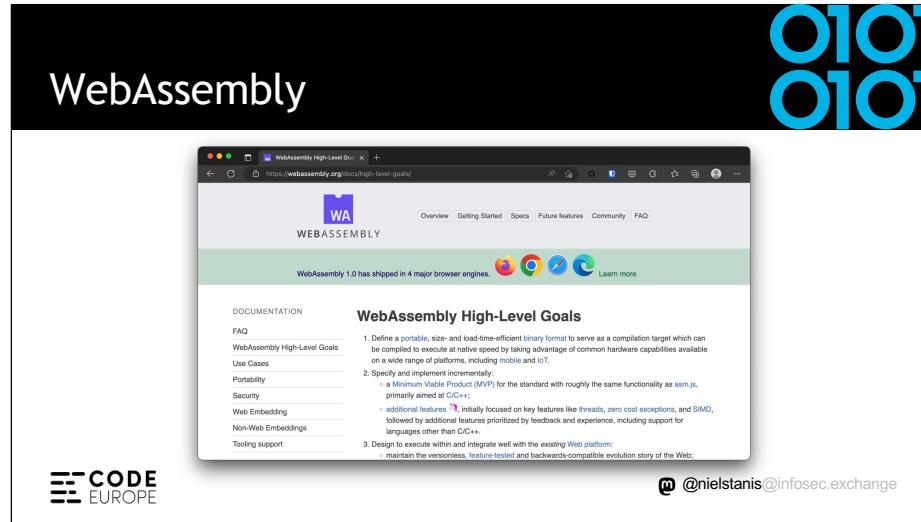
Who am I?

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



@nielstanis@infosec.exchange

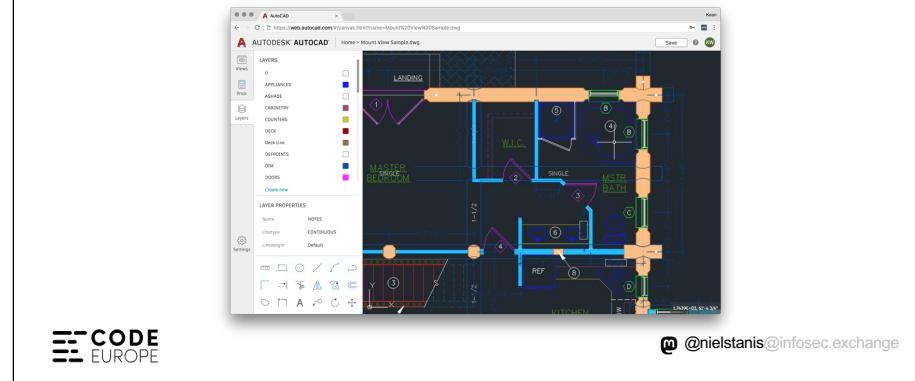




<https://hacks.mozilla.org/files/2019/08/04-01-star-diagram.png>

0101
0101

WebAssembly - AutoCAD



0101 0101

WebAssembly - SDK's

The image shows two side-by-side screenshots of web pages. The left screenshot is from Medium.com, featuring a dark header with the text 'WebAssembly - SDK's' and '0101 0101'. Below the header is a screenshot of a Medium article titled 'Introducing the Disney+ Application Development Kit (ADK)'. The right screenshot is from the Amazon Science blog, showing an article titled 'How Prime Video updates its app for more than 8,000 device types'. Both screenshots include the respective websites' navigation bars and some text from the articles.

CODE EUROPE

Mike Harley Published in disney-streaming

Introducing the Disney+ Application Development Kit (ADK)

By Tom Schroeder, Sr. SWE / Technical Lead, Native Client Platform, Living Room Devices

How Prime Video updates its app for more than 8,000 device types

By Alessandro Iosu

@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



@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



@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



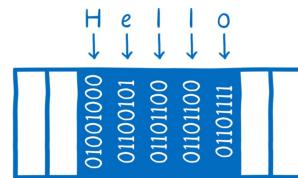
@nielstanis@infosec.exchange

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

WebAssembly Memory

0101
0101

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



@nielstanis@infosec.exchange

CODE
EUROPE

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

0101
0101

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

```
Console.WriteLine("Number is larger than 5");
```

```
Console.WriteLine("Number is smaller than 5");
```

```
Console.WriteLine("Done!");
```



@nielstanis@infosec.exchange



FireFox RLBox

The screenshot shows a web browser displaying the RLBox documentation at <https://rlbox.dev>. The page has a dark theme with a sidebar on the left containing navigation links: 'Overview', 'Setting up RLBox', 'Tutorial', 'Retrofitting sandboxing in a simple application', 'Enterprise sandbox', 'Installing and running tutorial examples', and 'Additional material'. The main content area features a large 'RLBox' logo with a book icon. Below the logo, the 'Overview' section is titled 'Overview' and describes RLBox as a toolkit for sandboxing third-party C libraries. It mentions its original development for Firefox and its current support for other frameworks like WebAssembly. The section includes two numbered points: 'A C++ Framework (RLBox) that makes it easy to retrofit existing application code to safely interface with sandboxed libraries.' and 'A Wasm backend (based on wasmc) for isolating (sandboxing) C libraries.' A note at the bottom states that the documentation is a work in progress.

CODE
EUROPE

@nielstanis@infosec.exchange

<https://rlbox.dev/>

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



Running .NET on WebAssembly

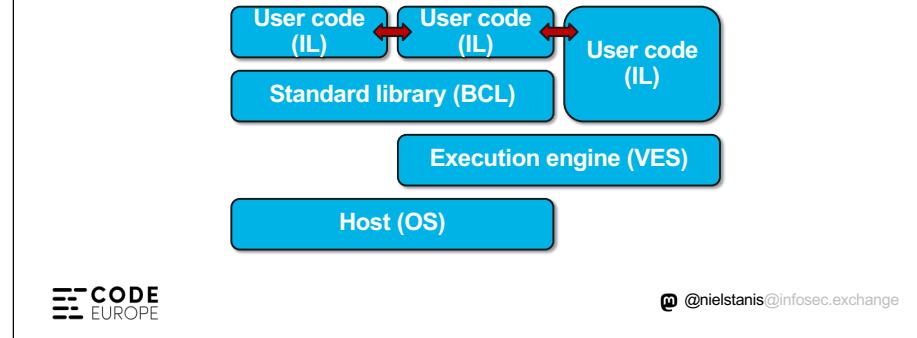


Diagram:

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



Running .NET on WebAssembly

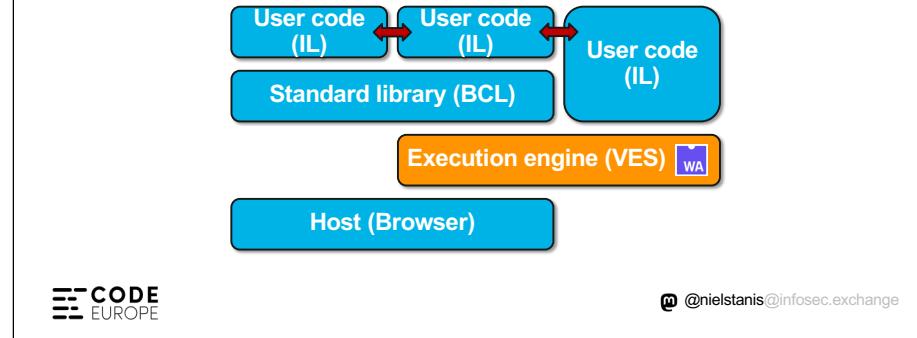


Diagram:

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

0101
0101

Blazor WebAssembly



CODE
EUROPE

@nielstanis@infosec.exchange



Running .NET on WebAssembly

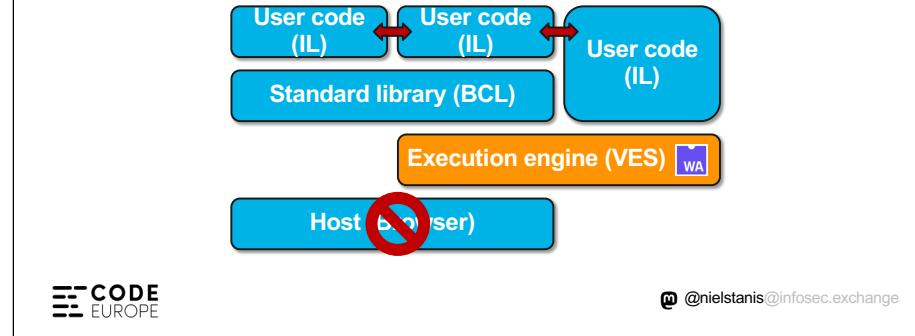


Diagram:

<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 recall .NET Standard? 😊



✉️ @nielstanis@infosec.exchange

Docker vs WASM & WASI

Solomon Hykes op Twitter: "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!"

[Tweet vertaald](#)

CODE EUROPE

@nielstanis@infosec.exchange



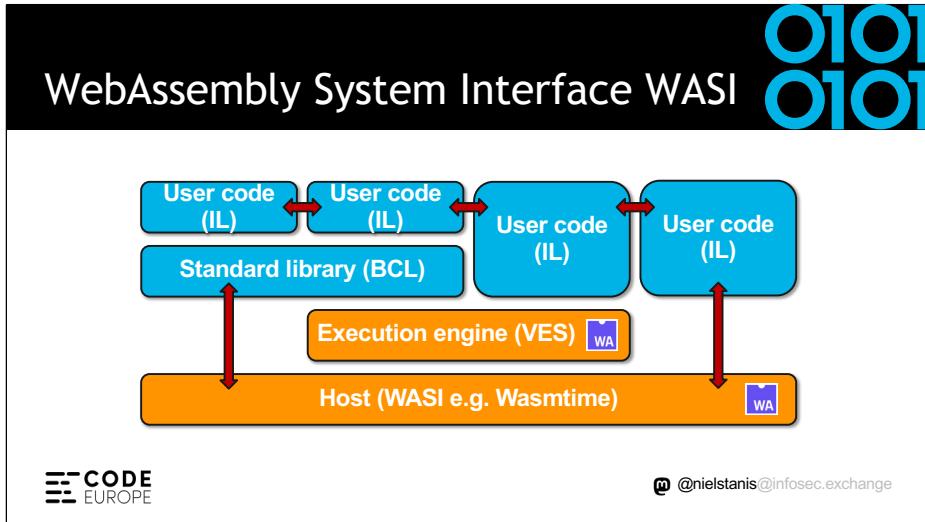
O1O1
O1O1

Docker & WASM

The screenshot displays the Docker+Wasm Technical Preview website. On the left, there's a header with the Docker+Wasm logo and a sub-header: "Introducing the Docker+Wasm Technical Preview". Below this is a bio for Michael Irwin, dated Oct 24 2022. A note states: "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." On the right, there's a diagram titled "Let's look at an example!" showing the Docker Engine interacting with a "container" which contains "Container process" and "Wasm Module". Below the diagram is a command-line example:

```
docker run -d -p 8080:8080 --wasm-exe=ls --runlevel=io.containerd.wasmedge.v1 -g /var/run/wasmkit2.sock michaelirwin24/wasm-example
```

At the bottom right, there's a Twitter handle: @nielstanis@infosec.exchange.



Experimental WASI SDK for .NET

O1O1
O1O1



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



Experimental WASI SDK for .NET

The screenshot shows a GitHub issue page for a repository named 'dotnet/runtime'. The specific issue is titled 'WASI support tracking' with the ID #65895. The issue was opened by SteveSandersonMS on February 25, 2022. The description of the issue states: '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 supported WASI-ready messa...'. The issue has 14 of 21 tasks completed. Labels include 'wasi', 'wasi-build-mono', and 'wasi-tracking'. The GitHub interface shows various tabs like Code, Issues, Pull requests, Discussions, Actions, Projects, Security, and Insights.

CODE
EUROPE

@nielstanis@infosec.exchange

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

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

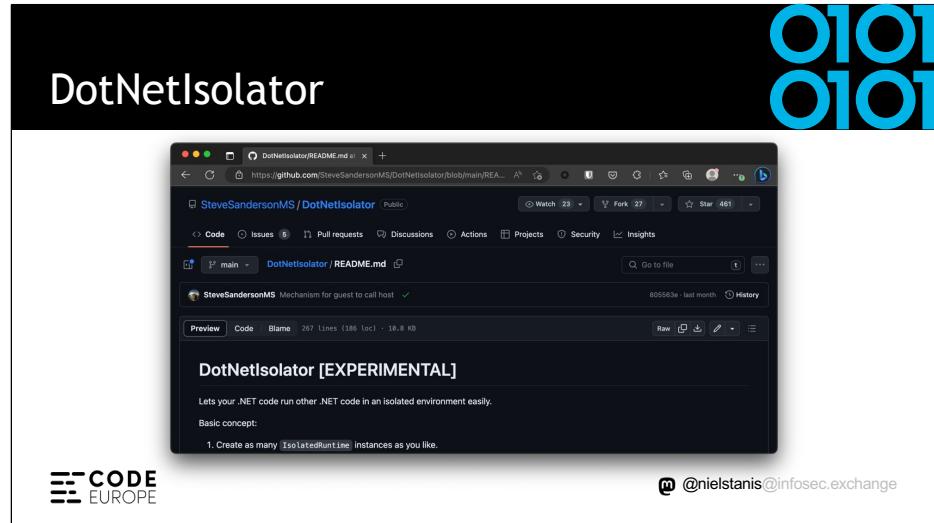


Extending .NET with WASM

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

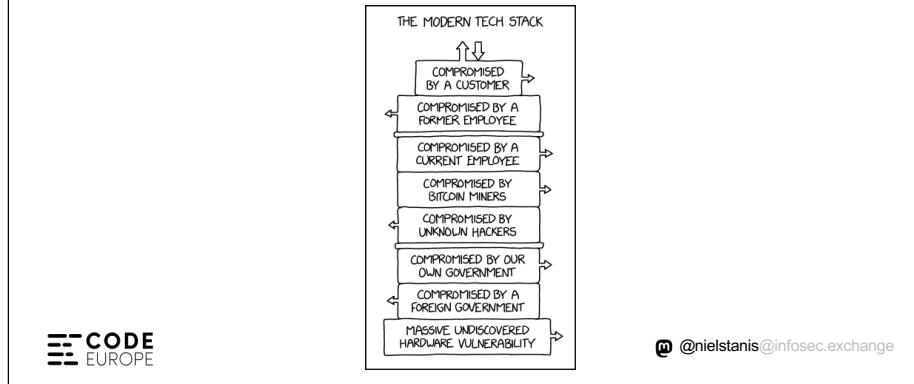


@nielstanis@infosec.exchange

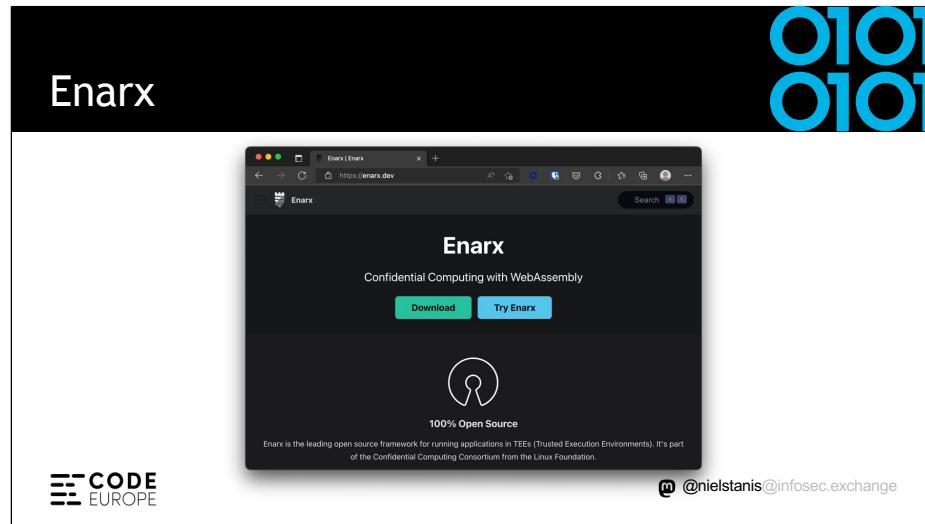


0101
0101

Trusted Computing - XKCD 2166



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



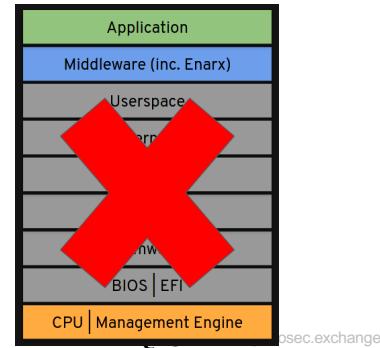
<https://enarx.dev/>

O1O1
O1O1

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

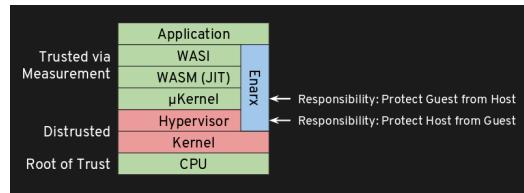
CODE
EUROPE





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



CODE
EUROPE

@nielstanis@infosec.exchange

WASM - What's next?

0101
0101

composition of an
average code base



CODE
EUROPE

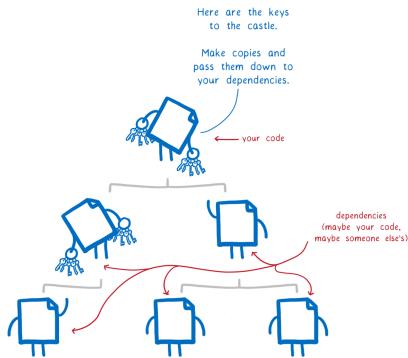
@nielstanis@infosec.exchange

0101
0101

Dependencies

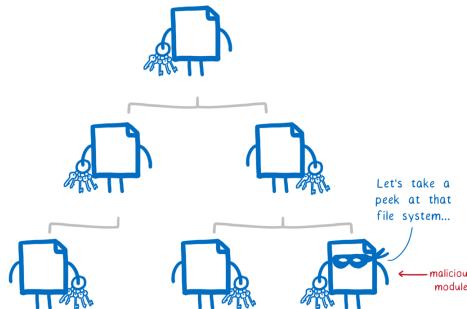
CODE
EUROPE

iis@infosec.exchange



0101
0101

Malicious module

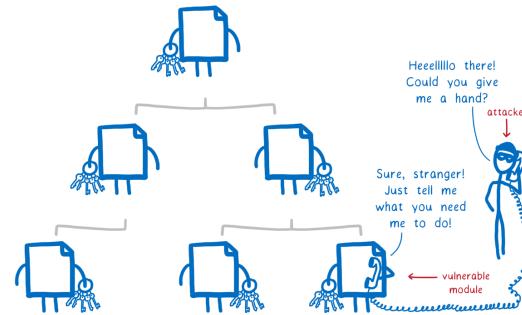


CODE
EUROPE

@nielstanis@infosec.exchange

Vulnerable module

0101
0101

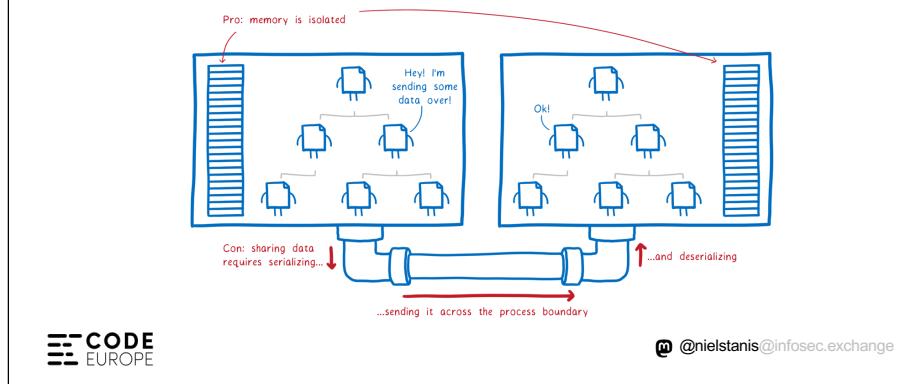


CODE
EUROPE

@nielstanis@infosec.exchange

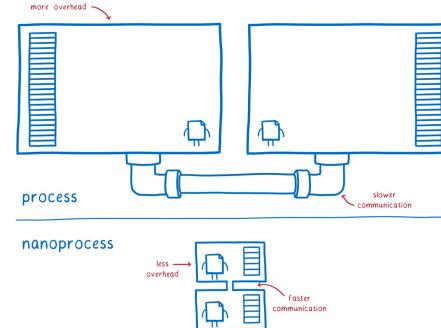
0101
0101

Process Isolation



WebAssembly Nano-Process

0101
0101



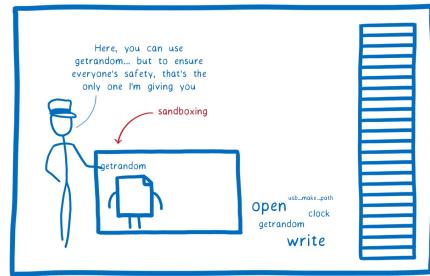
CODE
EUROPE

* not drawn to scale
@nielstanis@infosec.exchange

0101
0101

WebAssembly Nano-Process

1. Sandboxing



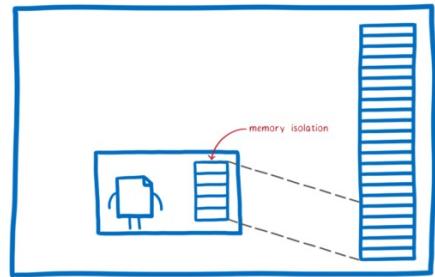
CODE
EUROPE

@nielstanis@infosec.exchange

WebAssembly Nano-Process

0101
0101

2. Memory model



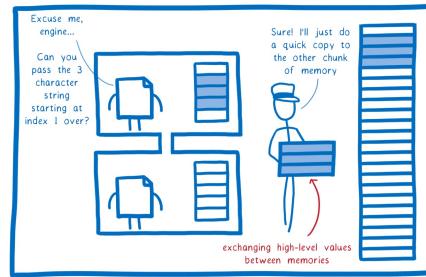
@nielstanis@infosec.exchange

CODE
EUROPE

0101
0101

WebAssembly Nano-Process

3. Interface Types



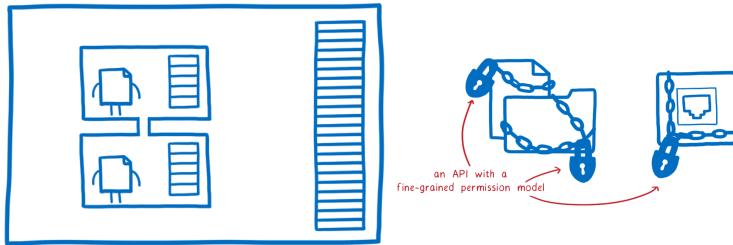
@nielstanis@infosec.exchange

CODE
EUROPE

0101
0101

WebAssembly Nano-Process

4. WebAssembly System Interface



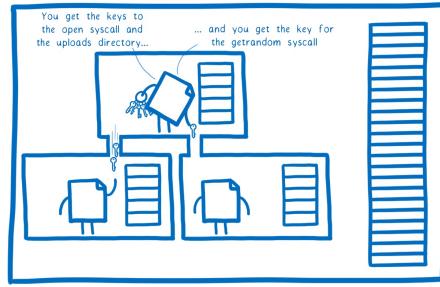
CODE
EUROPE

@nielstanis@infosec.exchange

0101
0101

WebAssembly Nano-Process

5. The missing link



✉ @nielstanis@infosec.exchange

CODE
EUROPE

WebAssembly Component Model



- Cloud Native WASM Day EU 2023
 - Evolution of Wasm: Past, Present, Future - Bailey Hayes, Cosmonic
https://youtu.be/6_BRLqxiZPU
 - WASI and the Cloud - Jiaxiao Zhou, Microsoft & Dan Gohman, Fastly
https://youtu.be/5WQRT62V_VU
 - Future of Component Tooling - Peter Huene & Guy Bedford, Fastly
<https://youtu.be/JClwpc7x4jU>

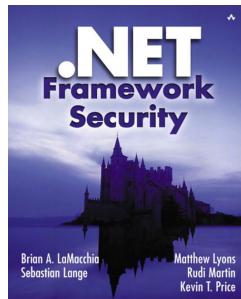
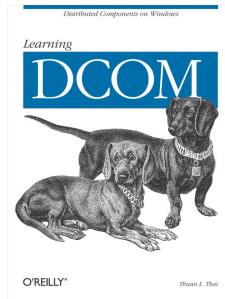


✉ @nielstanis@infosec.exchange

https://youtu.be/6_BRLqxiZPU
https://youtu.be/5WQRT62V_VU
<https://youtu.be/JClwpc7x4jU>

0101
0101

Have we seen this before?



CODE
EUROPE

@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



✉ @nielstanis@infosec.exchange

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



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



✉ @nielstanis@infosec.exchange



Questions?

- <https://github.com/nielstanis/codeeurope2023-wasm>
- ntanis at Veracode.com
- @nielstanis@infosec.exchange
- <https://blog.fennec.dev>
- Dziękuję! Thank you!



✉️ @nielstanis@infosec.exchange