

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

Niels Tanis



The Veracode logo watermark is a large, semi-transparent watermark covering the background of the slide. It consists of a grid of binary digits (0s and 1s) in a dark gray font. Superimposed on this grid is the word "VERACODE" in a bold, sans-serif font. The "O" in "VERACODE" is blue, matching the color of the large binary digits at the bottom of the slide.

0101
0101

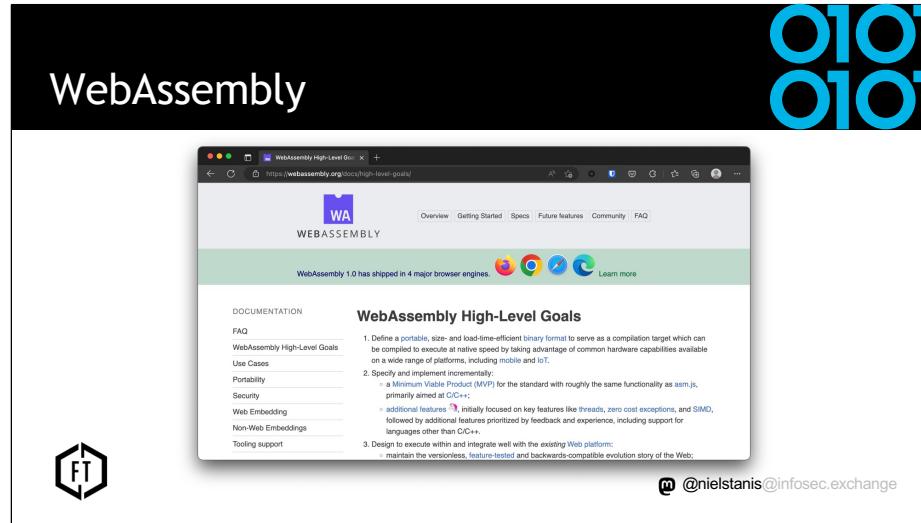
Who am I?

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



 @nielstanis@infosec.exchange

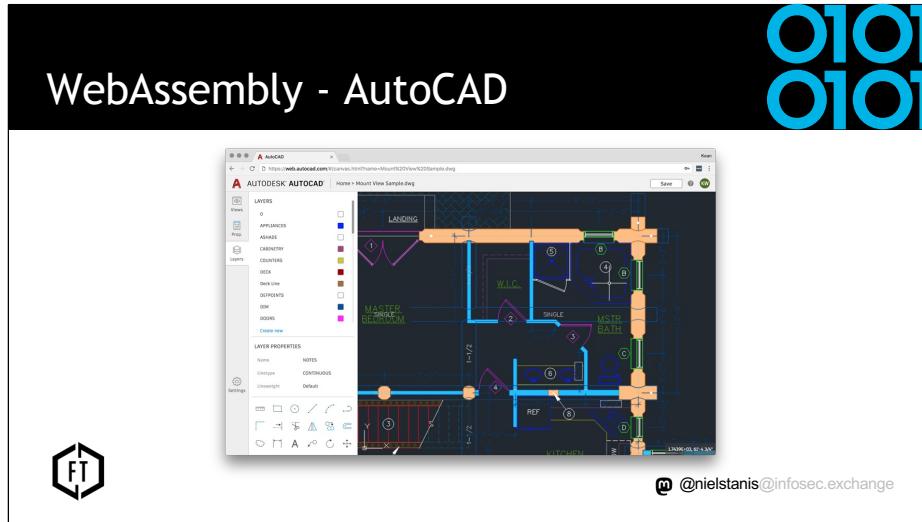




<https://hacks.mozilla.org/files/2019/08/04-01-star-diagram.png>

0101
0101

WebAssembly - AutoCAD



0101 0101

WebAssembly - SDK's

The image contains two side-by-side screenshots of web pages. The left screenshot is from Medium.com, showing a post titled "Introducing the Disney+ Application Development Kit (ADK)" by Mike Harley. The right screenshot is from Amazon's science blog, showing a post titled "How Prime Video updates its app for more than 8,000 device types" by Alessandro Iosu. Both posts discuss the use of WebAssembly in their respective applications.

FT

@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



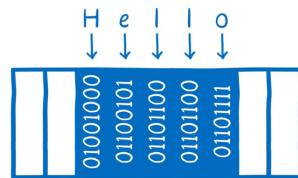
@nielstanis@infosec.exchange

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

0101
0101

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)
```

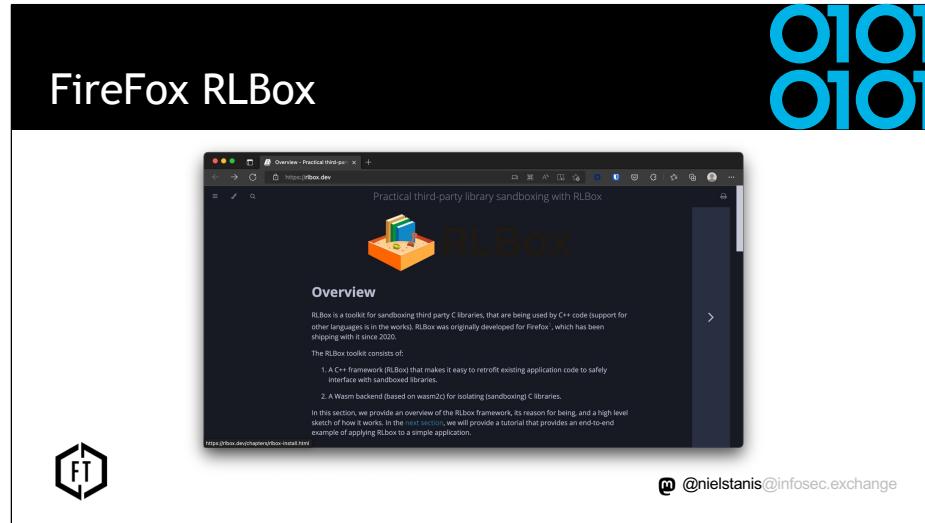
```
Console.WriteLine("Number is larger than 5");
```

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

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



✉ @nielstanis@infosec.exchange

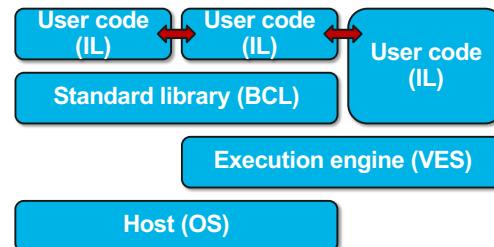


<https://rlbox.dev/>

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



Running .NET on WebAssembly



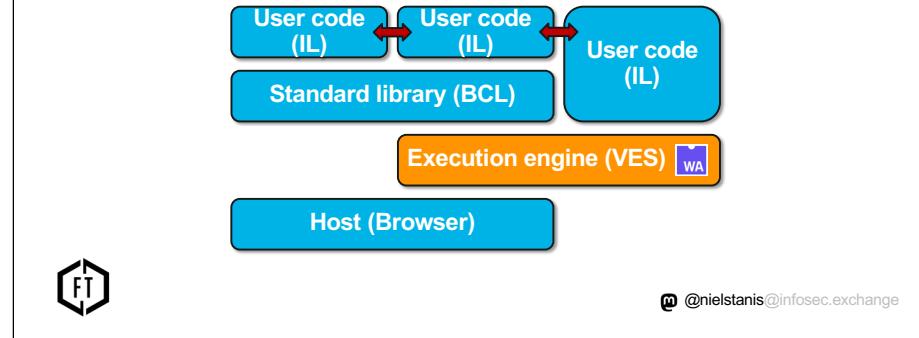
✉ @nielstanis@infosec.exchange

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>

0101
0101

Blazor WebAssembly



✉️ @nielstanis@infosec.exchange





Running .NET on WebAssembly

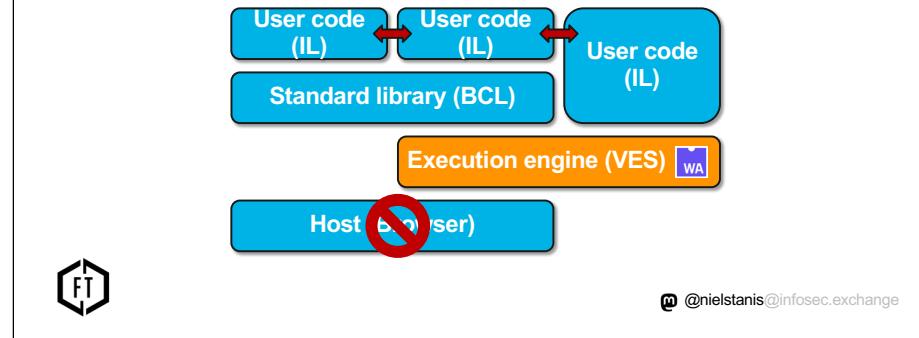


Diagram:

<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



0101
0101

Docker vs WASM & WASI

The screenshot shows a Twitter interface with a dark theme. A tweet from Solomon Hykes (@solomonhykes) is displayed. The tweet content is as follows:

"So will wasm replace Docker?" No, but imagine a future where Docker runs linux containers, windows containers and wasm containers side by side. Over time wasm might become the most popular container type. Docker will love them all equally, and run it all :)

If WASM+WASI existed in 2008, we wouldn't have needed to create 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 WAS is up to the task! <https://twitter.com/i/lnk/r/1statu...>

The tweet was posted on March 27, 2019, at 4:50 a.m. It has 56 Retweets, 5 Geciteerde Tweets, and 165 Vind-ik-leuks.

At the bottom right of the screenshot, there is a small watermark or icon that says "@nielstanis@infosec.exchange".



O1O1
O1O1

Docker & WASM

The screenshot shows a web browser displaying the "Introducing the Docker+Wasm Technical Preview" page. The page features 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, and a note about the availability of the preview. A small "FT" logo is visible at the bottom left of the page.

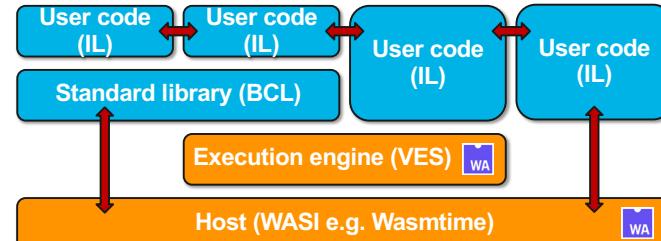
On the right side of the browser, there is a diagram illustrating the Docker+Wasm architecture. It shows a "Docker Engine" at the top, which oversees a "container" (represented by a blue square). Inside the container, there are three "Container process" boxes, each containing a "runc" box. To the right of the container, there is a "Wasm module" box labeled "wasmEdge".

Below the diagram, a section titled "Let's look at an example!" contains a command-line snippet:

```
docker run -d -p 8080:8080 --openwasm-exe=ls --runfile=/io/containerd_wasmedge_v1 --glslangver=LWebGL32 MichaelIrwin24/wasm-example
```

At the bottom right of the browser window, there is a social media handle: [@niestanis@infosec.exchange](https://twitter.com/niestanis).

WebAssembly System Interface WASI



✉ @nielstanis@infosec.exchange



0101
0101

Experimental WASI SDK for .NET



<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' and has the identifier '#65895'. The issue was opened by 'SteveSandersonMS' on February 25, 2022, and has 14 tasks assigned. 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 separable WASI-ready release.' Below the description, there is a list of tasks:

- [] Add [wasi] Add new RIO for WASI #7780
- [] [API-Proposal]: [serializesystem-Timers] #76309 API change
- [] Build runtime using WASI SDK instead of using the Emscripten-built binary.
- [] MSBUILDFixes for WASI
- [] Enable所有权 to all APIs exposed by System.NatLab. Currently there's a small hardcoded list.
- [] [wasi] consult with libnative #79048



@nielstanis@infosec.exchange

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

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



Extending .NET with WASM

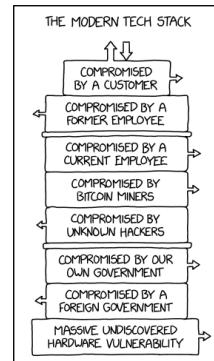
- WasmTime.NET NuGet package
- Can run WASM inside of any .NET application
- Extend with Rust based WASM module
- Demo time!



@nielstanis@infosec.exchange

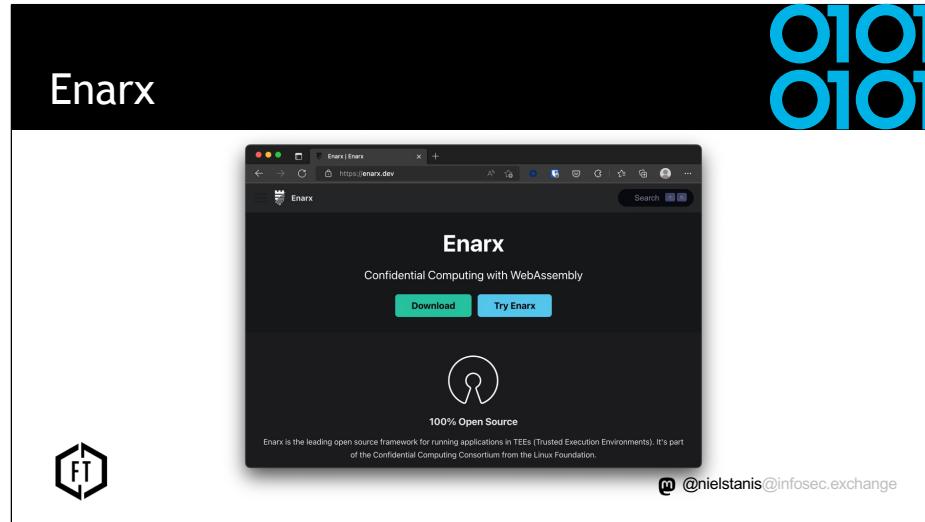
Trusted Computing - XKCD 2166

0101
0101



@nielstanis@infosec.exchange

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

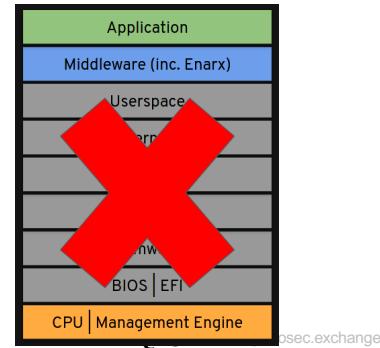


<https://enarx.dev/>



Enarx Threat Model

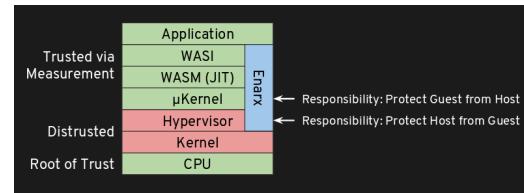
- Don't trust the host
- Don't trust the host owner
- Don't trust the host operator
- Hardware cryptographically verified
- Software audited and cryptographically verified





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

WASM - What's next?

0101
0101

composition of an
average code base



@nielstanis@infosec.exchange

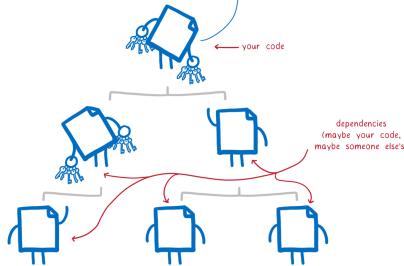
Dependencies

0101
0101



Here are the keys
to the castle.

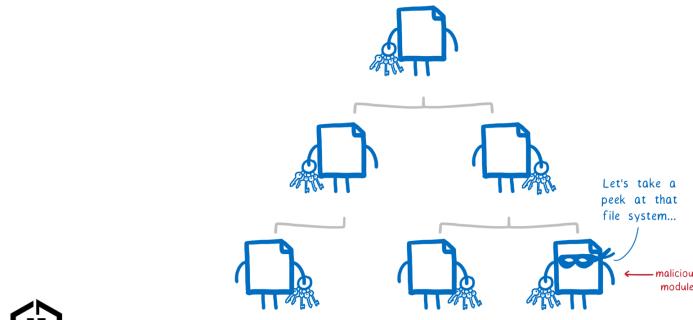
Make copies and
pass them down to
your dependencies.



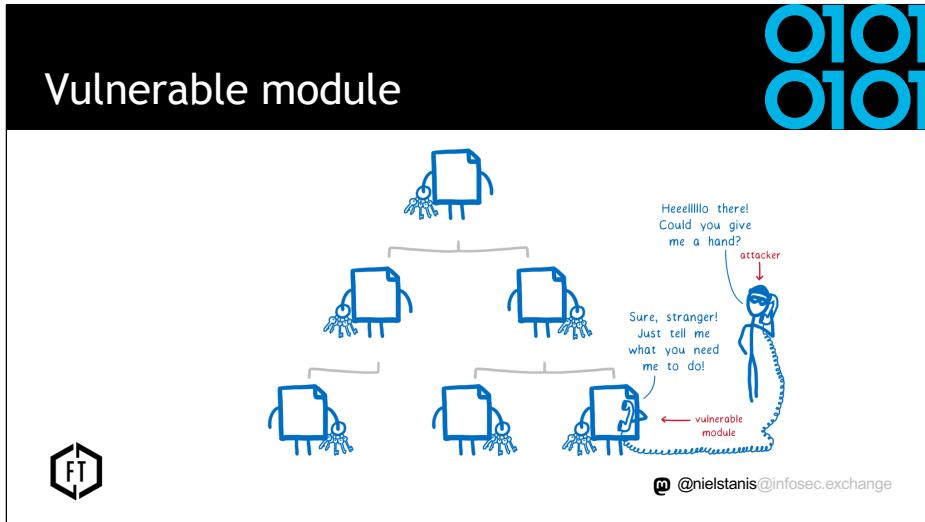
iis@infosec.exchange

Malicious module

0101
0101

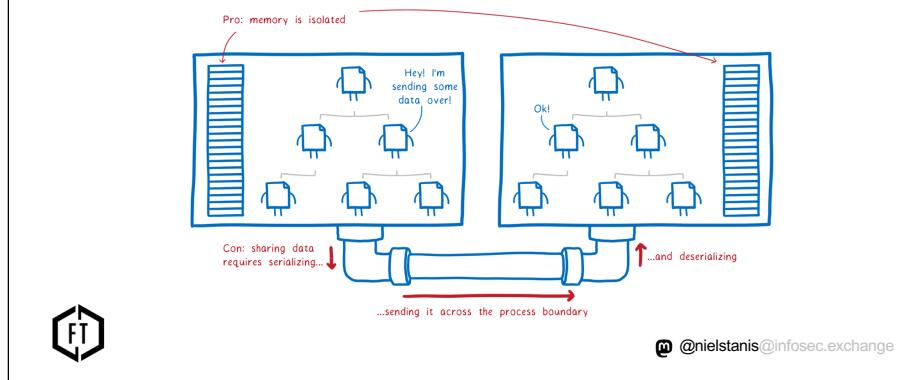


@nielstanis@infosec.exchange



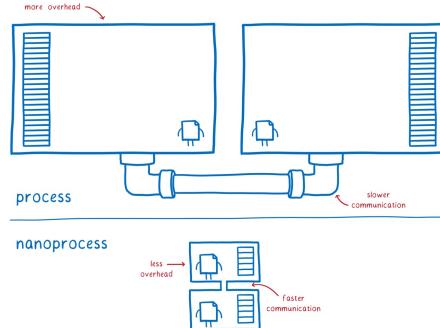
0101
0101

Process Isolation



WebAssembly Nano-Process

0101
0101



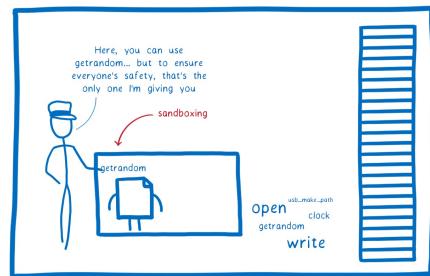
* not drawn to scale @nielstanis@infosec.exchange



0101
0101

WebAssembly Nano-Process

1. Sandboxing



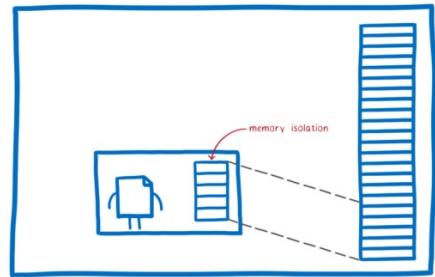
@nielstanis@infosec.exchange



WebAssembly Nano-Process

0101
0101

2. Memory model



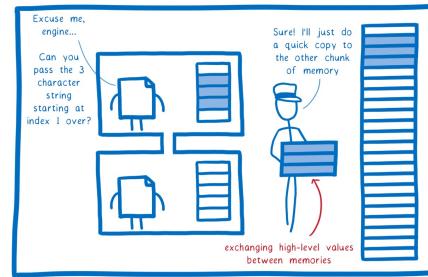
✉ @nielstanis@infosec.exchange



0101
0101

WebAssembly Nano-Process

3. Interface Types



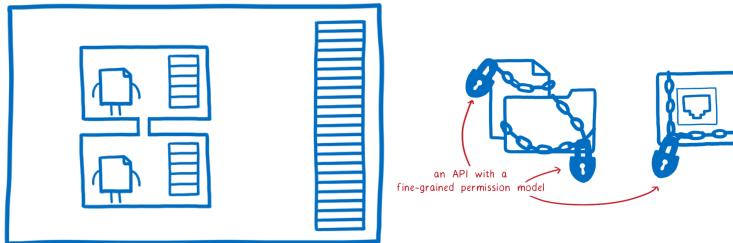
@nielstanis@infosec.exchange



0101
0101

WebAssembly Nano-Process

4. WebAssembly System Interface

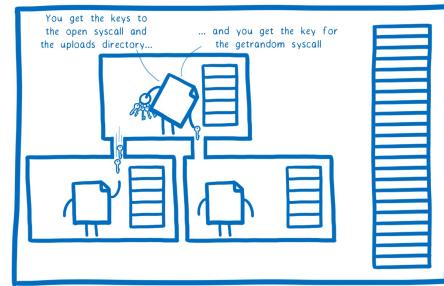


@nielstanis@infosec.exchange

WebAssembly Nano-Process

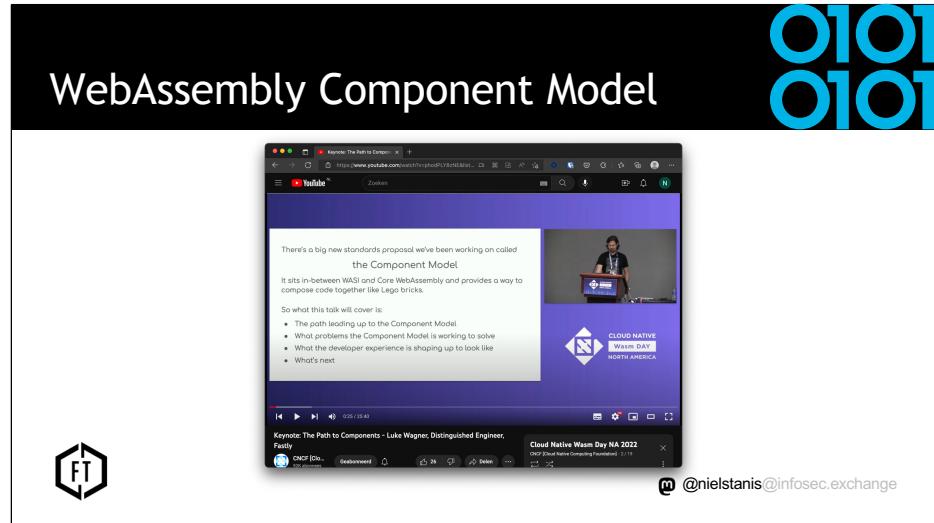
0101
0101

5. The missing link



@nielstanis@infosec.exchange

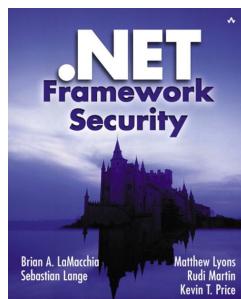
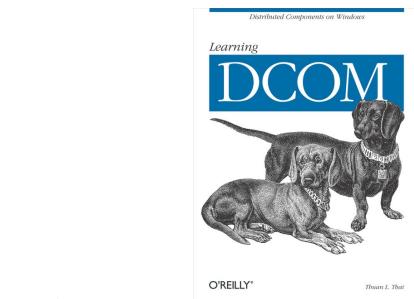




<https://www.youtube.com/watch?v=JotItWTHD5s>

0101
0101

Have we seen this before?



@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/futuretech2023>
- ntanis at Veracode.com
- @nielstanis@infosec.exchange
- <https://blog.fennec.dev>
- Bedankt! Thank you!



 @nielstanis@infosec.exchange