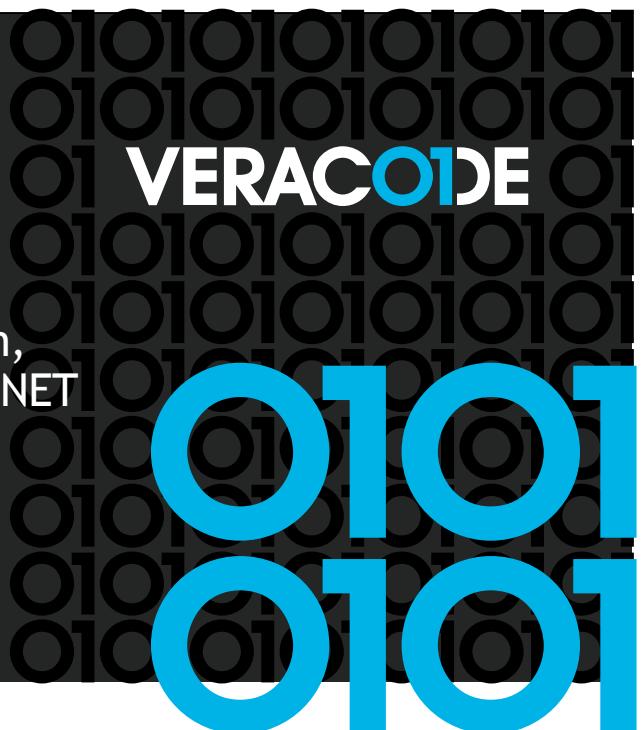




Using WebAssembly to run,
extend, and secure your .NET
application

Niels Tanis



0101
0101

Who am I?

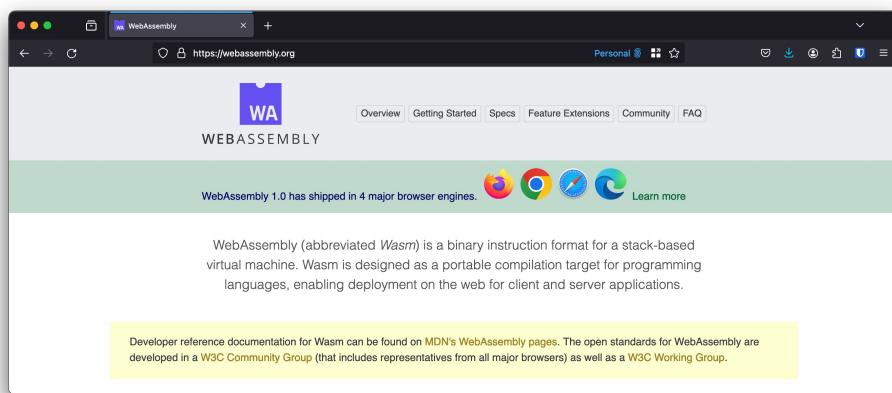
- Niels Tanis
- Sr. Principal Security Researcher
 - Background .NET Development, Pentesting/ethical hacking, and software security consultancy
 - Research on static analysis for .NET apps
 - Enjoying Rust!
- Microsoft MVP - Developer Technologies



 @nielstanis@infosec.exchange

WebAssembly

0101
0101

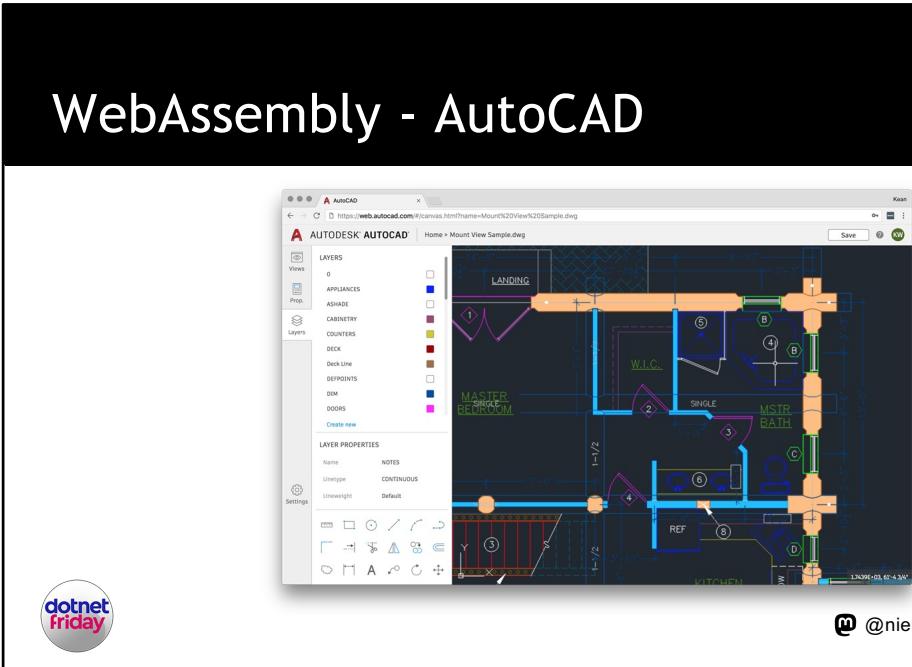


@nielstanis@infosec.exchange

<https://webassembly.org/>

0101
0101

WebAssembly - AutoCAD



@nielstanis@infosec.exchange

0101
0101

WebAssembly - SDK's

The screenshot shows a Medium article page. At the top, there's a navigation bar with a back arrow, a forward arrow, a refresh button, and a search icon. Below the bar, it says "Published in disney-streaming". The main content area features a profile picture of Mike Hanley, a "Follow" button, and the date "Sep 8, 2021 · 10 min read · Listen". The title of the article is "Introducing the Disney+ Application Development Kit (ADK)". Below the title, it says "By: Tom Schroeder, Sr. SWE / Technical Lead, Native Client Platform, Living Room Devices". There are two small icons: a play button with the number 415 and a comment icon with the number 4. At the bottom of the article preview, there are several small links and a search bar.

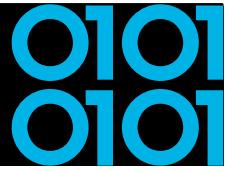
The screenshot shows a blog post from Amazon Science. At the top, it says "amazon | science" and "CLOUD AND SYSTEMS". The title of the post is "How Prime Video updates its app for more than 8,000 device types". Below the title, it says "The switch to WebAssembly increases stability, speed." and "By Alexandru Enă January 27, 2022". There is a "Share" button and a link to "Prime Video". At the bottom, it says "At Prime Video, we're delivering content to millions of customers on".



@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>



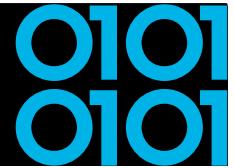
Agenda

- 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

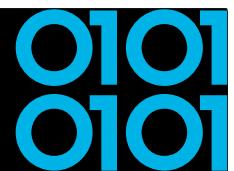


WEBASSEMBLY



@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



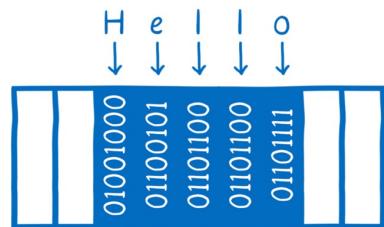
@nielstanis@infosec.exchange

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

0101
0101

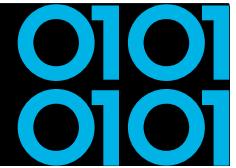
WebAssembly Memory

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



@nielstanis@infosec.exchange



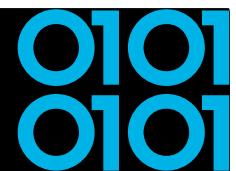


WebAssembly Control-Flow Integrity

```
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!");
```

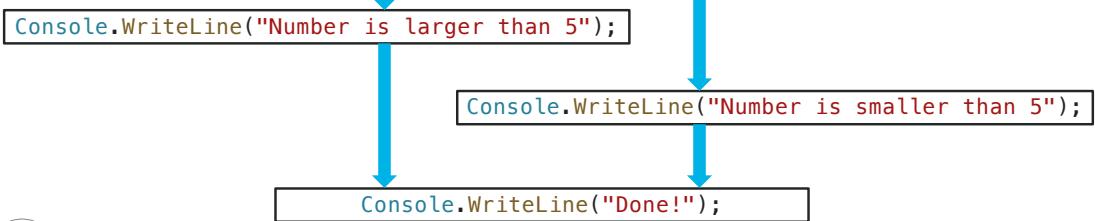


 @nielstanis@infosec.exchange



WebAssembly Control-Flow Integrity

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



@nielstanis@infosec.exchange

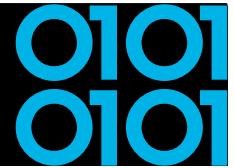
FireFox RLBox

dotnet
Friday

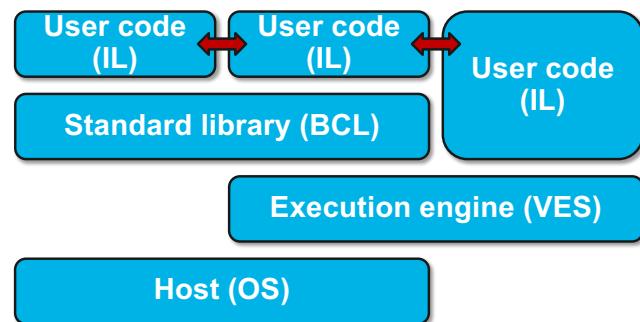
@nielstanis@infosec.exchange

<https://rlbox.dev/>

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



Running .NET on WebAssembly



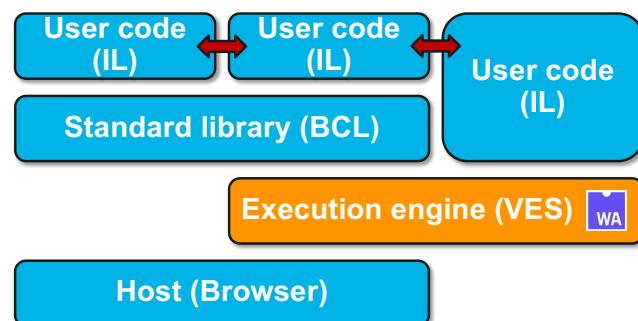
@nielstanis@infosec.exchange

Diagram:

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

Running .NET on WebAssembly

0101
0101



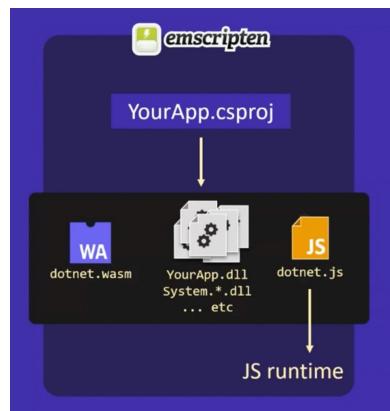
@nielstanis@infosec.exchange

Diagram:

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

0101
0101

Blazor WebAssembly

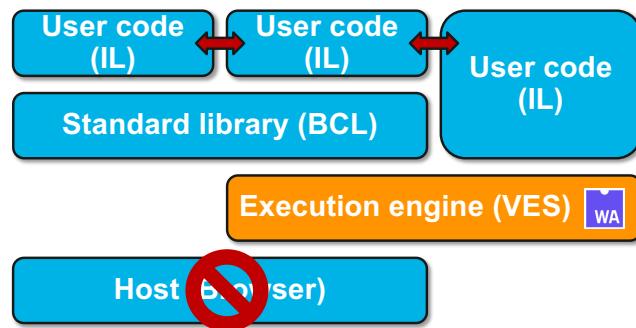


@nielstanis@infosec.exchange





Running .NET on WebAssembly



@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 recall .NET Standard? 😊



 @nielstanis@infosec.exchange

0101
0101

Docker vs WASM & WASI

The screenshot shows a Twitter web client interface. On the left, there's a sidebar with various icons: a blue bird (Twitter), a house, a magnifying glass, a person, a mail icon, and a circular profile picture. The main content area has a dark background. At the top, it says "Collectie". Below that, a tweet from "Solomon Hykes" (@solomonstre) is displayed:

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 the tweet, there's 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".

On the right side of the screen, there's a dark sidebar with a small circular icon containing a white "dotnet" logo and the text "dotnet Friday".

In the bottom right corner of the main content area, there's a small "m" icon followed by the email address "@nielstanis@infosec.exchange".

0101
0101

Docker vs WASM & WASI

The screenshot shows a Twitter post from Solomon Hykes (@solomonstre) dated March 27, 2019. The tweet reads:

"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 :)

Below the tweet, a reply from @linclark states: "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! [twitter.com/linclark/status...](#)".

Engagement statistics at the bottom of the tweet show 56 Retweets, 5 Geciteerde Tweets, and 165 Vind-ik-leuks.

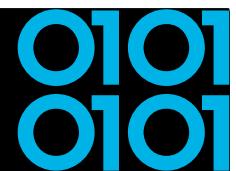
On the left side of the screen, there is a circular badge with the text "dotnet Friday". On the right side, there is a watermark with the letter "m" and the email address "@nielstanis@infosec.exchange".

0101
0101

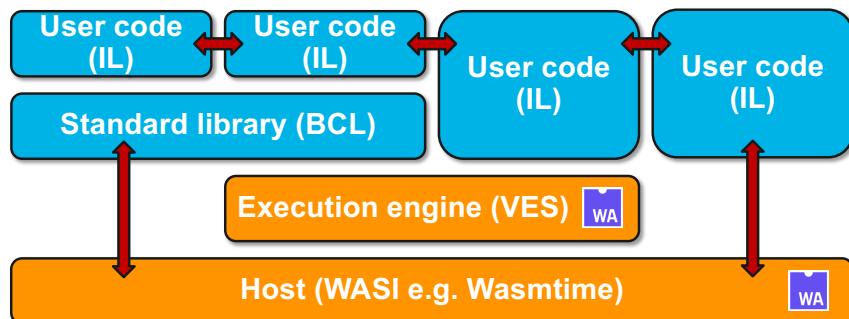
Docker & WASM

The screenshot shows two browser tabs side-by-side. The left tab displays the 'Introducing the Docker+Wasm Technical Preview' page, featuring a header with the Docker+Wasm logo, a title, author information (Michael Irwin, Oct 24 2022), and a note about the availability of the Technical Preview. The right tab shows a detailed architectural diagram of the Docker Engine. It starts with the 'Docker Engine' at the top, which manages a single container identified by 'containerd'. Inside the container, there are three parallel stacks: 'containerd-shim', 'runc', and 'Container process'. A fourth stack, 'containerd-wasm-shim', also interacts with the 'runc' and 'Container process' layers. At the bottom, 'wasmedge' and 'Wasm Module' components are shown. Below the diagram, a callout text reads: 'Let's look at an example! After installing the preview, we can run the following command to start an example Wasm application:' followed by a command-line snippet.





WebAssembly System Interface WASI



@nielstanis@infosec.exchange

0101
0101

Experimental WASI SDK for .NET

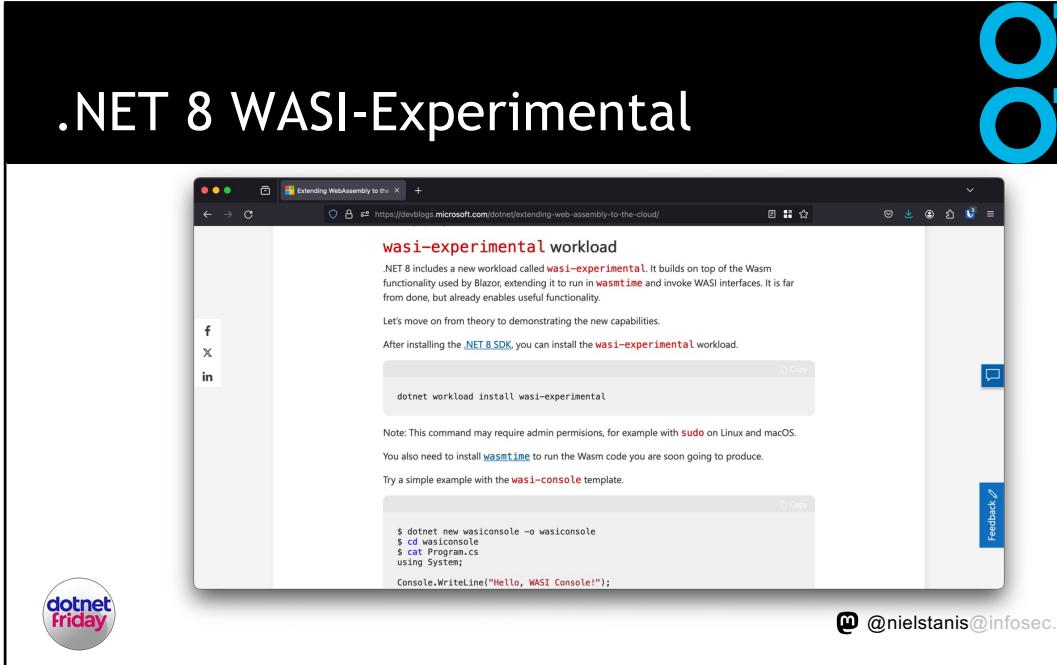


@nielstanis@infosec.exchange

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

0101
0101

.NET 8 WASI-Experimental

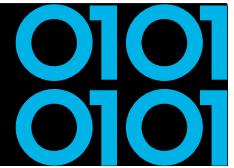


@nielstanis@infosec.exchange

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

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

<https://devblogs.microsoft.com/dotnet/extending-web-assembly-to-the-cloud/>



Extending .NET with WASM

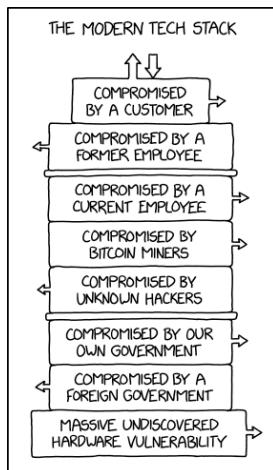
- WasmTime.NET NuGet package
- Can run WASM inside of any .NET application
- Extend with Rust based WASM module
- Limit capabilities
- Demo time!



 @nielstanis@infosec.exchange

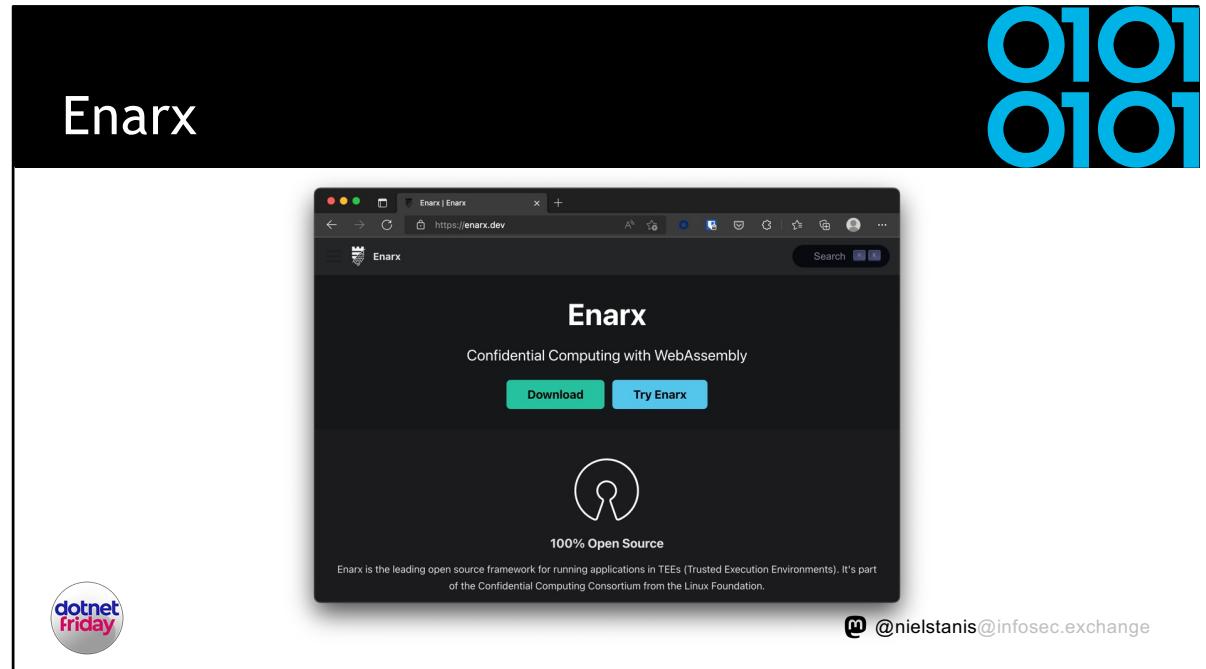
0101
0101

Trusted Computing - XKCD 2166



@nielstanis@infosec.exchange

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

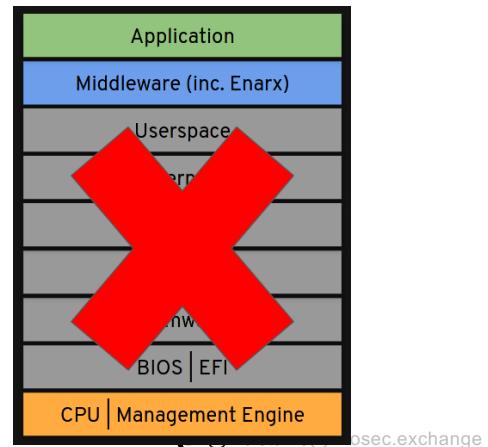


<https://enarx.dev/>

0101
0101

Enarx Threat Model

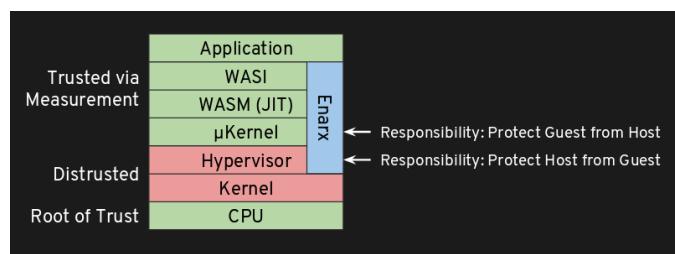
- Don't trust the host
- Don't trust the host owner
- Don't trust the host operator
- Hardware cryptographically verified
- Software audited and cryptographically verified





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

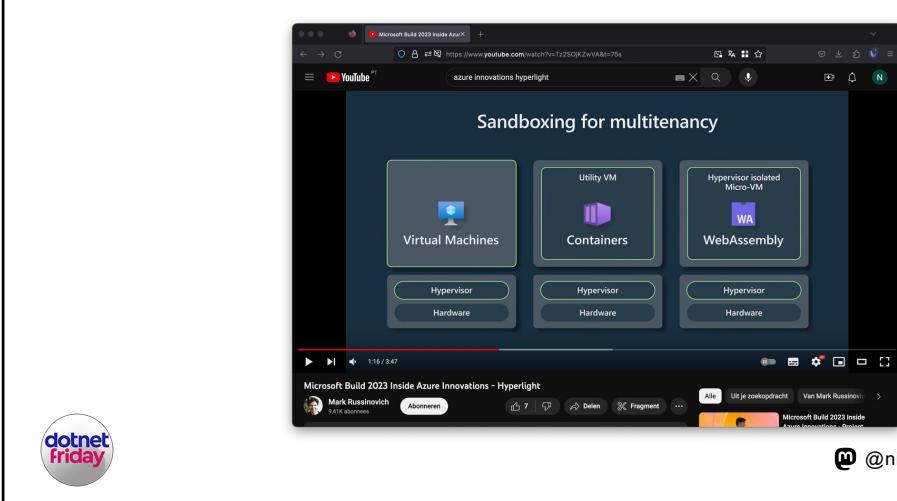


@nielstanis@infosec.exchange



0101
0101

Project Hyperlight

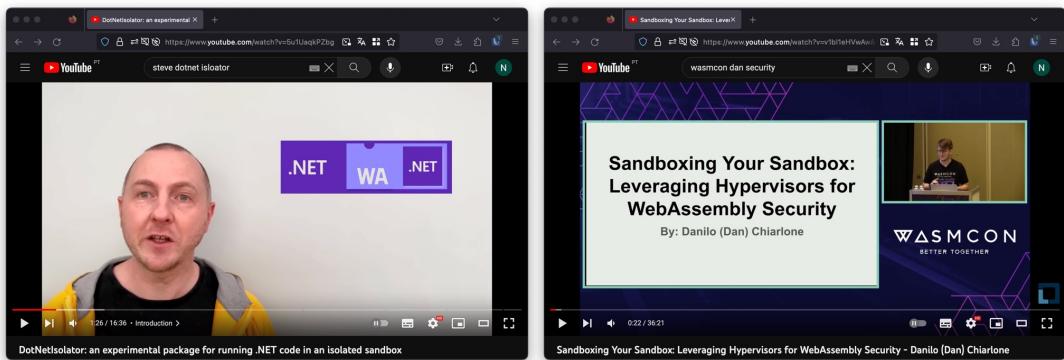


@nielstanis@infosec.exchange



DotNetIsolator & Project Hyperlight

0101
0101



@nielstanis@infosec.exchange

0101
0101

WASM - What's next?

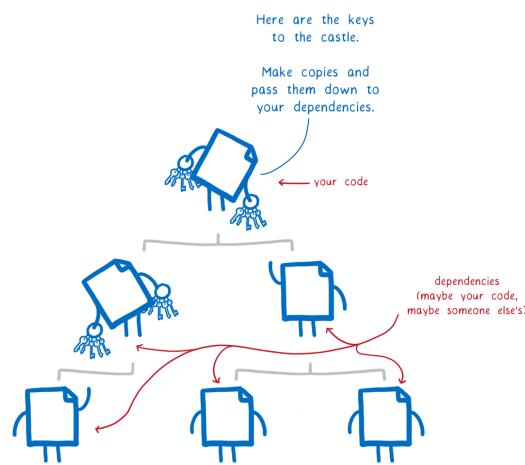
composition of an
average code base



 @nielstanis@infosec.exchange

0101
0101

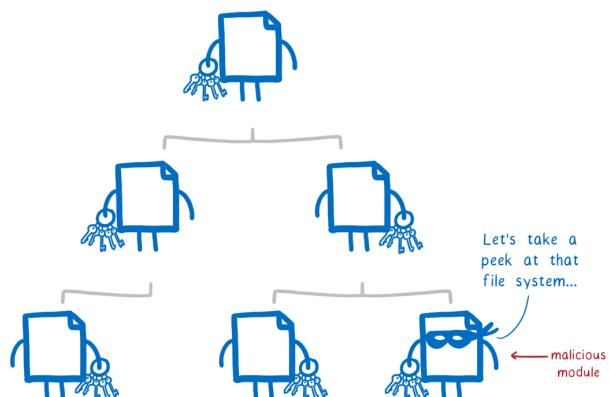
Dependencies



iis@infosec.exchange

0101
0101

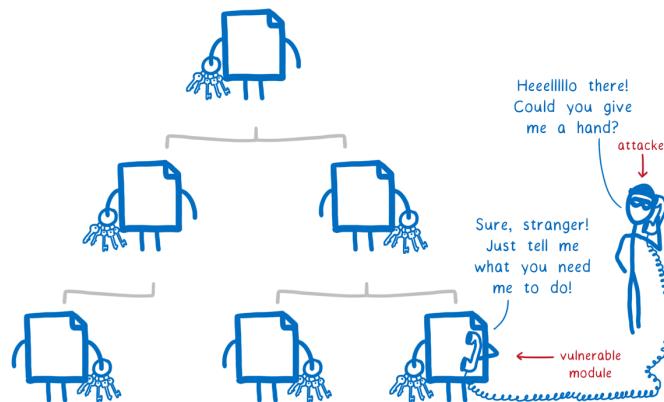
Malicious module



@nielstanis@infosec.exchange

0101
0101

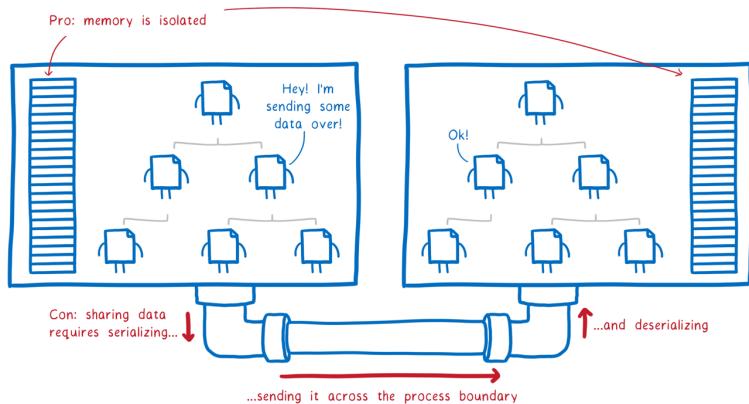
Vulnerable module



@nielstanis@infosec.exchange

0101
0101

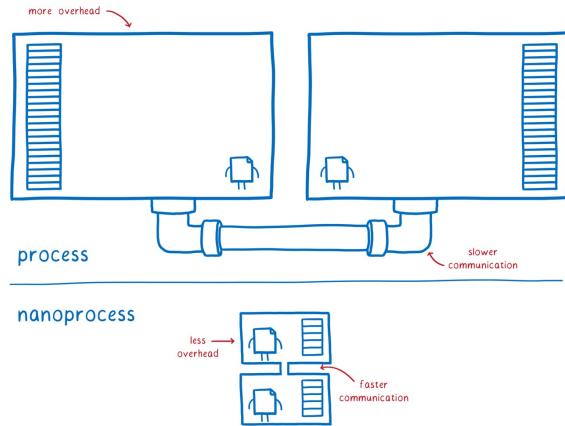
Process Isolation



@nielstanis@infosec.exchange

0101
0101

WebAssembly Nano-Process



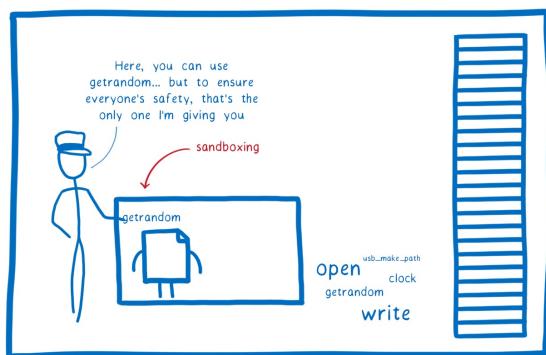
* not drawn to scale
m @nielstanis@infosec.exchange



0101
0101

WebAssembly Nano-Process

1. Sandboxing



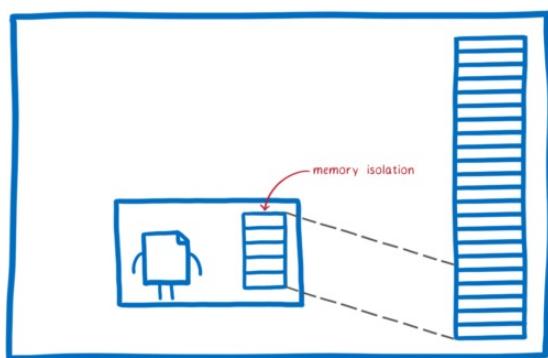
@nielstanis@infosec.exchange



0101
0101

WebAssembly Nano-Process

2. Memory model



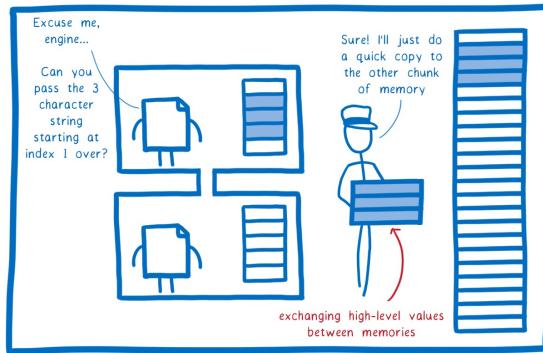
 @nielstanis@infosec.exchange



0101
0101

WebAssembly Nano-Process

3. Interface Types



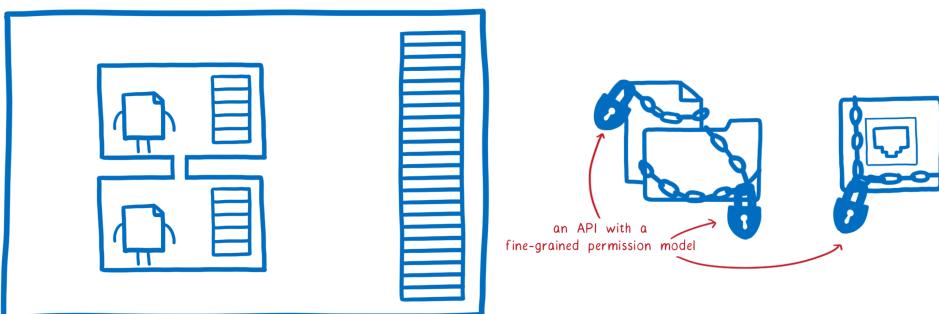
@nielstanis@infosec.exchange



0101
0101

WebAssembly Nano-Process

4. WebAssembly System Interface

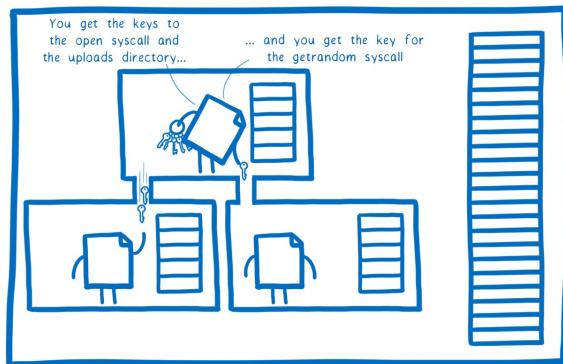


@nielstanis@infosec.exchange

0101
0101

WebAssembly Nano-Process

5. The missing link

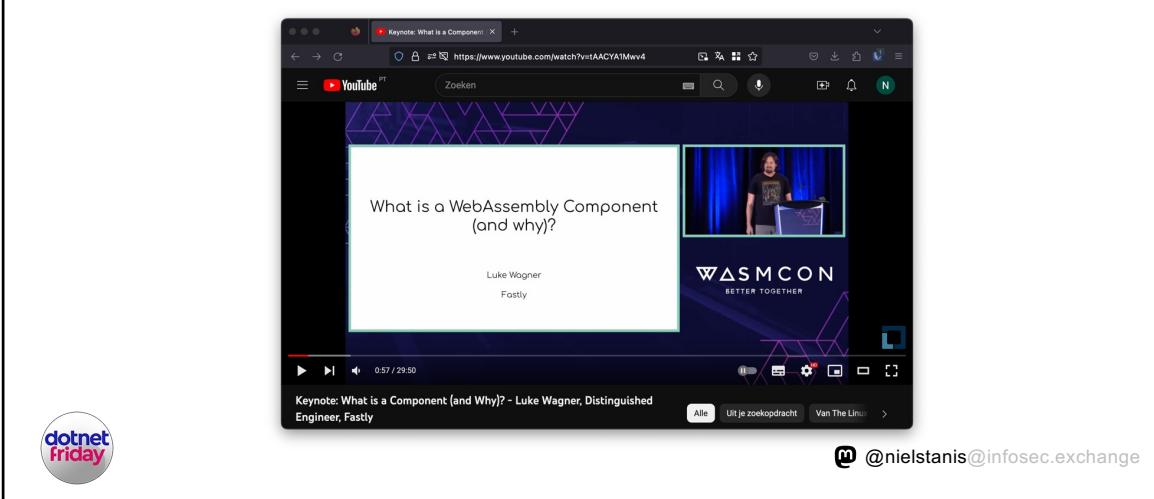


 @nielstanis@infosec.exchange



0101
0101

WebAssembly Component Model



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

0101
0101

WASI Preview 2

The screenshot shows a web browser window with the title "WASI Preview 2 Launched - sunfishcode". The URL is <https://blog.sunfishcode.online/wasi-preview2/>. The page content is as follows:

sunfishcode's blog
A blog by sunfishcode

WASI Preview 2 Launched

Posted on January 25, 2024

The WASI Subgroup has just voted to launch WASI Preview 2! This blog post is a brief look at the present, past, and future of WASI.

The present

The Subgroup voted to launch Preview 2!

This is a major milestone! We made it! At the same time, the journey is only just beginning. But let's talk this moment to step back and look at what this means.

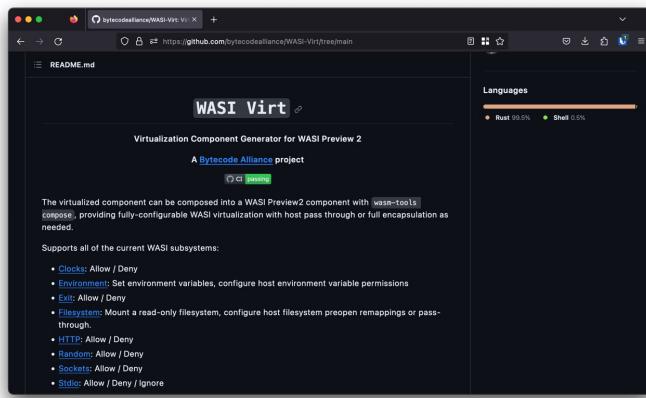
Most immediately, what this means is that the WASI Subgroup officially says that the Preview 2 APIs are stable. There is still a lot more to do, in writing more documentation, more tests, more toolchains, more implementations, and there are a lot more features that we all want to add. This vote today is a milestone along the way, rather than a destination in itself.

It also means that WASI is now officially based on the Wasm component model, which makes it cross-language and virtualizable. Figuring out what a component model even is, designing it, implementing it, and building APIs using it has been a

@nielstanis@infosec.exchange

0101
0101

WASI Virt



@nielstanis@infosec.exchange

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

WasmComponent.SDK

0101
0101

The screenshot shows a GitHub repository page for 'WasmComponent.SDK'. The main content is the README file, which includes sections for 'Purpose' and 'Without this package...'. To the right of the README, there is a sidebar showing the repository's statistics: 'Languages' (C# 100.0%) and a profile picture for 'NielsPilgaard'.

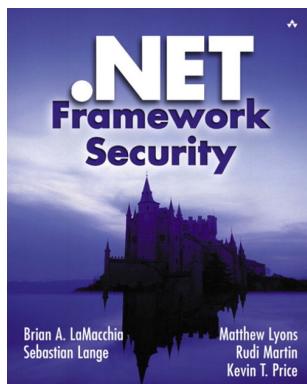
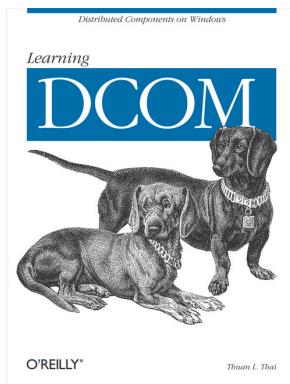


@nielstanis@infosec.exchange

<https://github.com/SteveSandersonMS/wasm-component-sdk/>

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

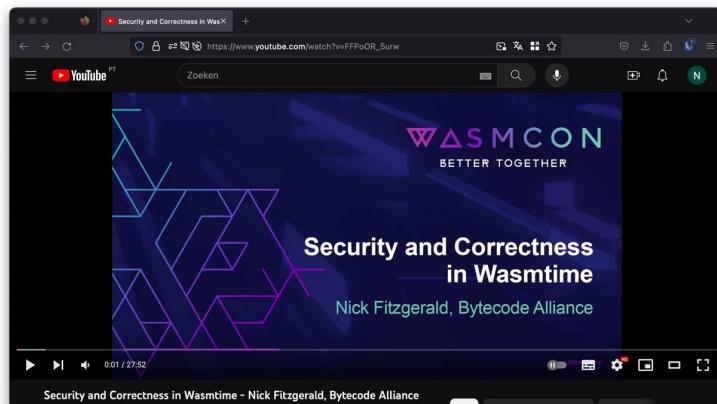


 @nielstanis@infosec.exchange

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

Runtimes and Security

0101
0101



@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!
- WASI Preview 2 big milestone; now tooling can be implemented!



@nielstanis@infosec.exchange

0101
0101

Questions?

- <https://github.com/nielstanis/dotnetfriday2024>
- ntanis at Veracode.com
- @nielstanis@infosec.exchange
- <https://blog.fennec.dev>
- Dank je wel! Thank you!

