



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

Niels Tanis
Sr. Principal Security Researcher

*SWETUG
GÖTEBORGS*

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

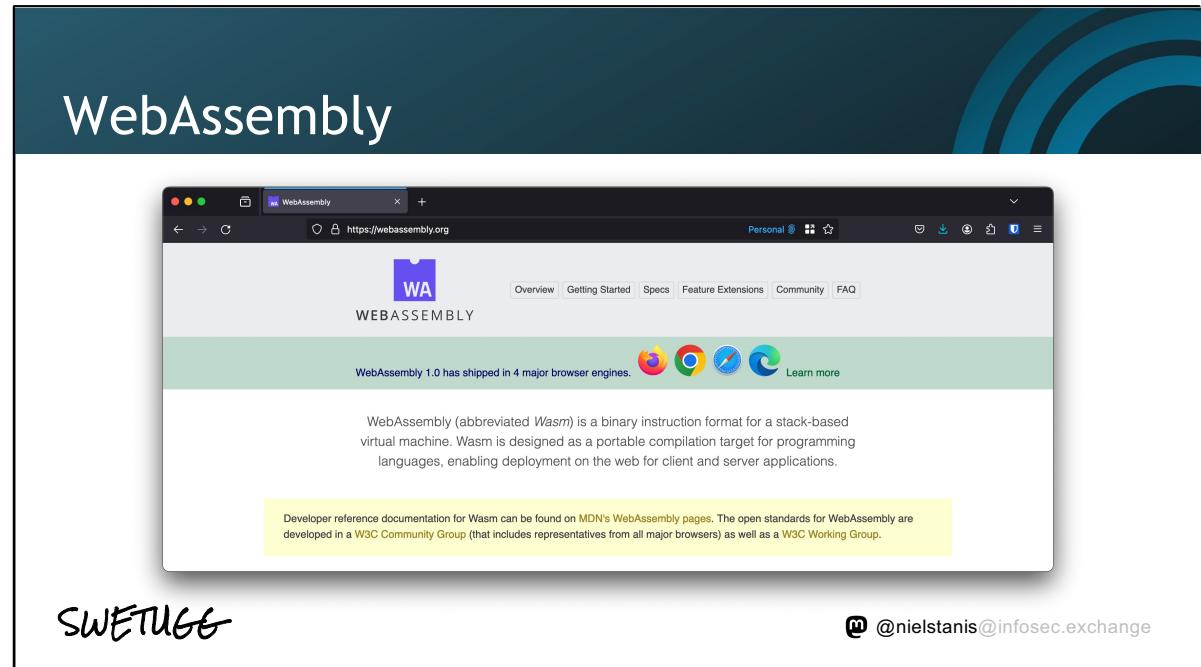
VERACODE



MVP Microsoft®
Most Valuable
Professional

SWETUGS

 @nielstanis@infosec.exchange

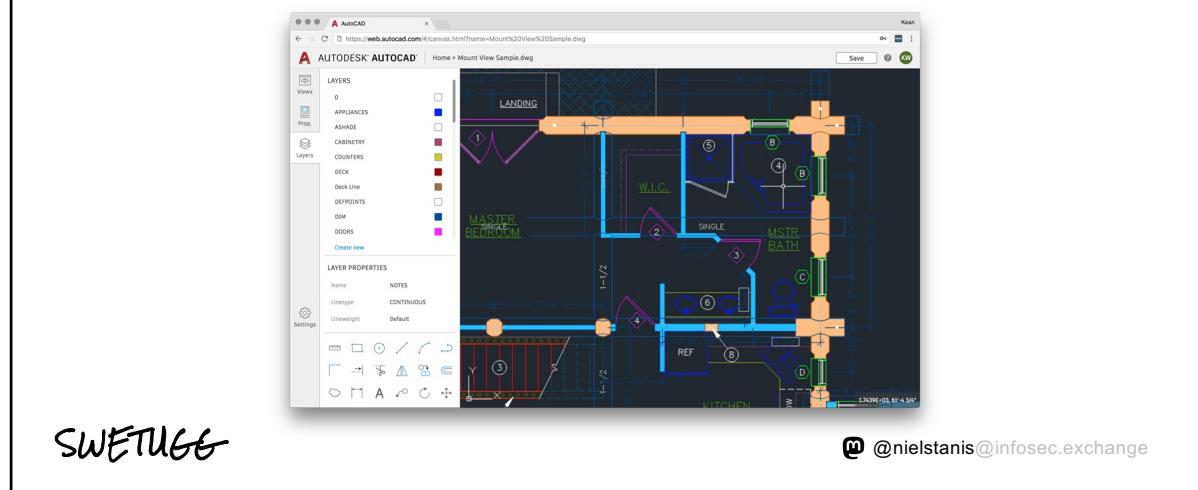


SWETUGS

 @nielstanis@infosec.exchange

<https://webassembly.org/>

WebAssembly - AutoCAD



WebAssembly - SDK's

The screenshot shows a Medium article page. At the top, there's a dark header with the title 'Introducing the Disney+ Application Development Kit (ADK)'. Below the header, the author's profile picture and name 'Mike Hanley' are shown, along with a 'Follow' button. The main content area contains the article text, which starts with 'Published in disney-streaming'. The text is written by Tom Schroeder, Sr. SWE / Technical Lead, Native Client Platform, Living Room Devices. It includes a summary, a 'Get started' button, and social sharing icons.

The screenshot shows a blog post from 'amazon | science' under the 'CLOUD AND SYSTEMS' category. The title is 'How Prime Video updates its app for more than 8,000 device types'. The post discusses how the switch to WebAssembly increases stability and speed. It's authored by Alexandru Enă and dated January 27, 2022. The post includes a 'Share' button and a note about delivering content to millions of customers.

SWETUGG

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

SWETUGG

 @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

SWETUGS

@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



WEBASSEMBLY

SWETUGS

@nielstanis@infosec.exchange

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

WebAssembly

- The security model of WebAssembly:
 - Protect users from buggy or malicious modules
 - Provide developers with useful primitives and mitigations for developing safe applications



WEBASSEMBLY

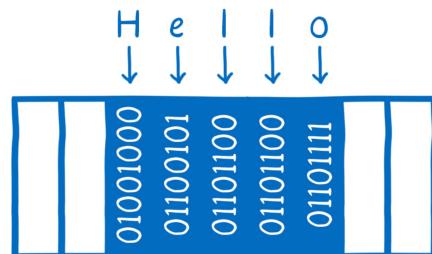
SWETUGS

@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



SWETUGG

 @nielstanis@infosec.exchange

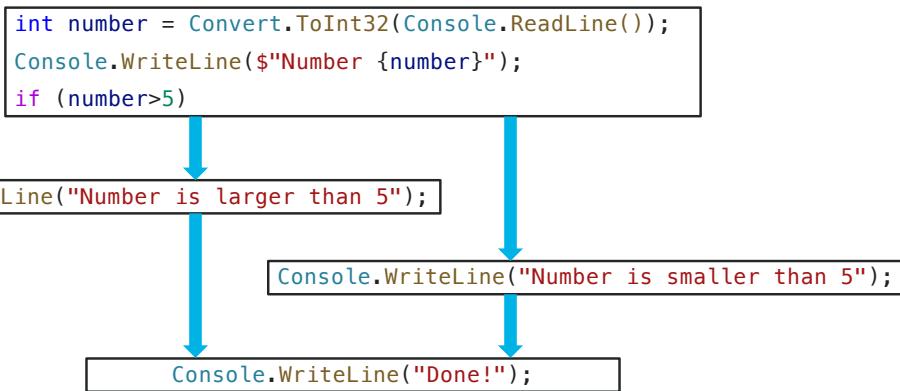
WebAssembly Control-Flow Integrity

```
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!");
```

SWETUGG

 @nielstanis@infosec.exchange

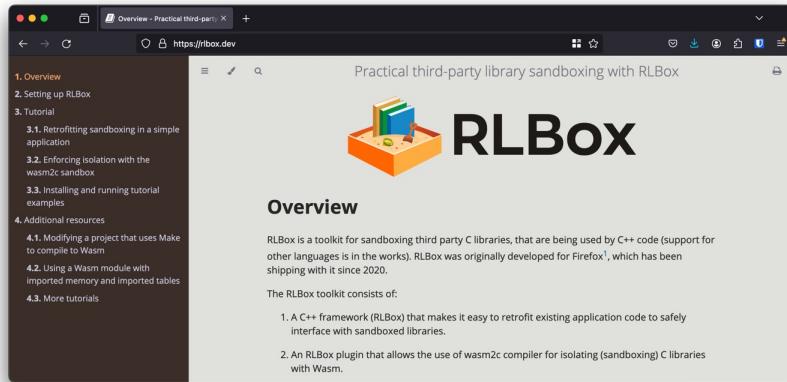
WebAssembly Control-Flow Integrity



SWETUGS

@nielstanis@infosec.exchange

FireFox RLBox



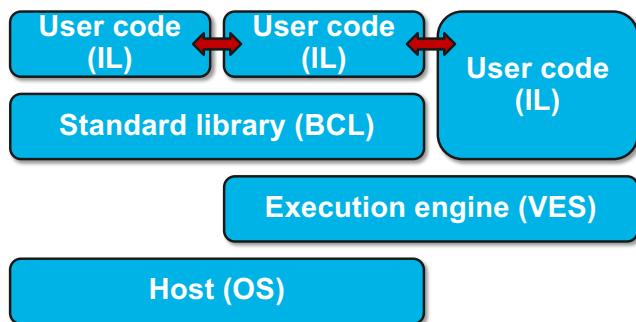
SWETUGS

 @nielstanis@infosec.exchange

<https://rlbox.dev/>

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

Running .NET on WebAssembly



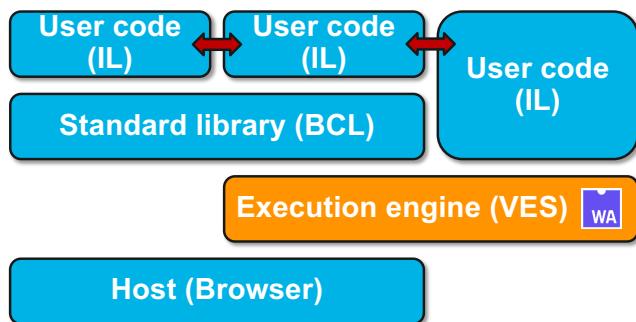
SWETUGS

@nielstanis@infosec.exchange

Diagram:

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

Running .NET on WebAssembly



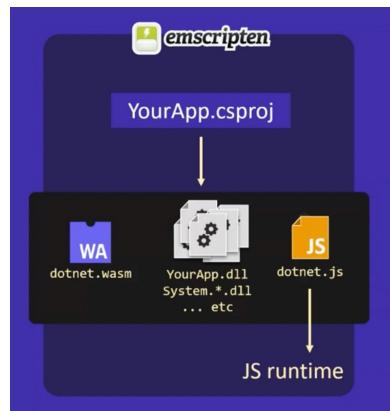
SWETUGS

@nielstanis@infosec.exchange

Diagram:

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

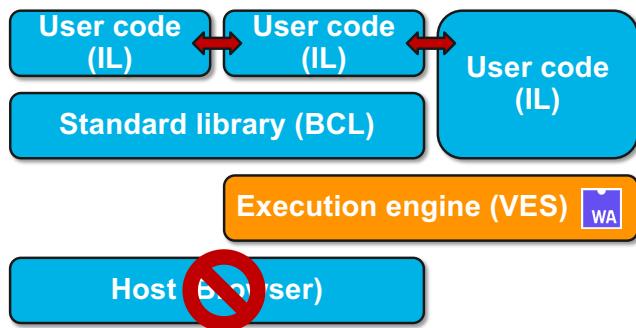
Blazor WebAssembly



SWETUGS

@nielstanis@infosec.exchange

Running .NET on WebAssembly



SWETUGS

@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



SWETUGS

@nielstanis@infosec.exchange

WebAssembly System Interface WASI

- Strong sandbox with Capability Based Security
- Preview1, supports e.g. FileSystem actions
- Preview2 supports a lot more!
- Anyone recall .NET Standard? ☺



SWETUGS

@nielstanis@infosec.exchange

Docker vs WASM & WASI

A screenshot of a Twitter web client window. The main tweet is from Solomon Hykes (@solomonstre) and reads:
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 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...

The reply includes a link: hacks.mozilla.org/2019/03/standa...
Dede collectie weergeven

At the bottom left of the window, there is handwritten text that appears to read "SWETUGG".

@nielstanis@infosec.exchange

Docker vs WASM & WASI

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 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/110900000000000000

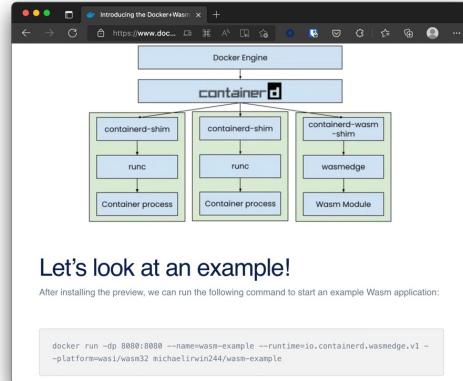
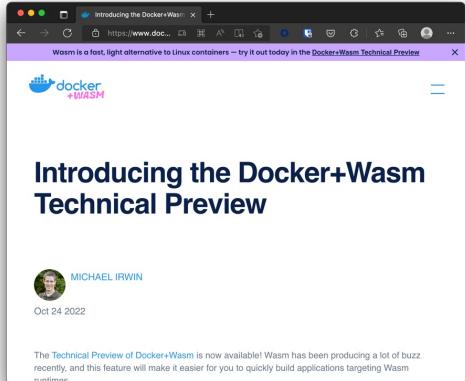
4:50 a.m. - 28 mrt. 2019 - Twitter Web App

56 Retweets 5 Geciteerde Tweets 165 Vind-ik-leuks

SWETUGG

M @nielstanis@infosec.exchange

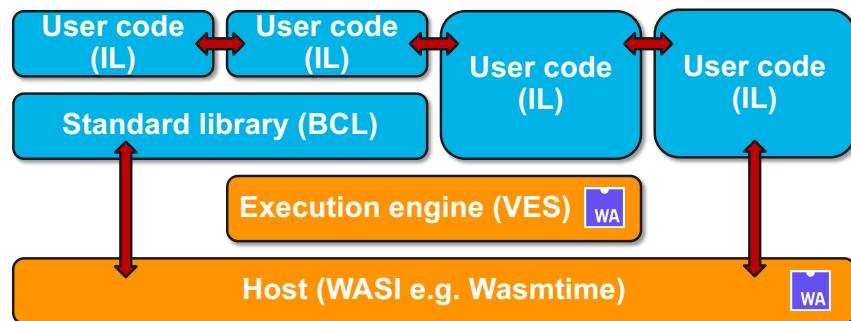
Docker & WASM



SWETUGG

@nielstanis@infosec.exchange

WebAssembly System Interface WASI



SWETUGS

👤 @nielstanis@infosec.exchange

Experimental WASI SDK for .NET



SWETUGS

 @nielstanis@infosec.exchange

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

.NET 8 WASI-Experimental

wasi-experimental workload

.NET 8 includes a new workload called `wasi-experimental`. It builds on top of the Wasm functionality used by Blazor, extending it to run in `wasmtime` and invoke WASI interfaces. It is far from done, but already enables useful functionality.

Let's move on from theory to demonstrating the new capabilities.

After installing the [.NET 8 SDK](#), you can install the `wasi-experimental` workload.

```
dotnet workload install wasi-experimental
```

Note: This command may require admin permissions, for example with `sudo` on Linux and macOS.

You also need to install `wasmtime` to run the Wasm code you are soon going to produce.

Try a simple example with the `wasi-console` template.

```
$ dotnet new wasiconsole -o wasiconsole
$ cd wasiconsole
$ cat Program.cs
using System;
Console.WriteLine("Hello, WASI Console!");
```

SWETUGS

[@nielstanis@infosec.exchange](https://github.com/dotnet/runtime/issues/65895)

<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

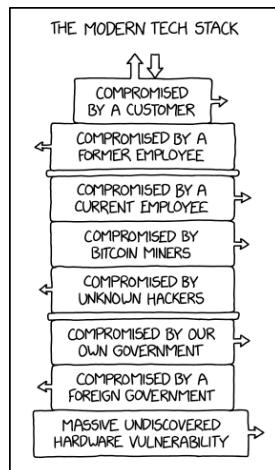
- WasmTime.NET NuGet package
- Can run WASM inside of any .NET application
- Extend with Rust based WASM module
- Limit capabilities
- Demo time!

SWETUGS

 @nielstanis@infosec.exchange

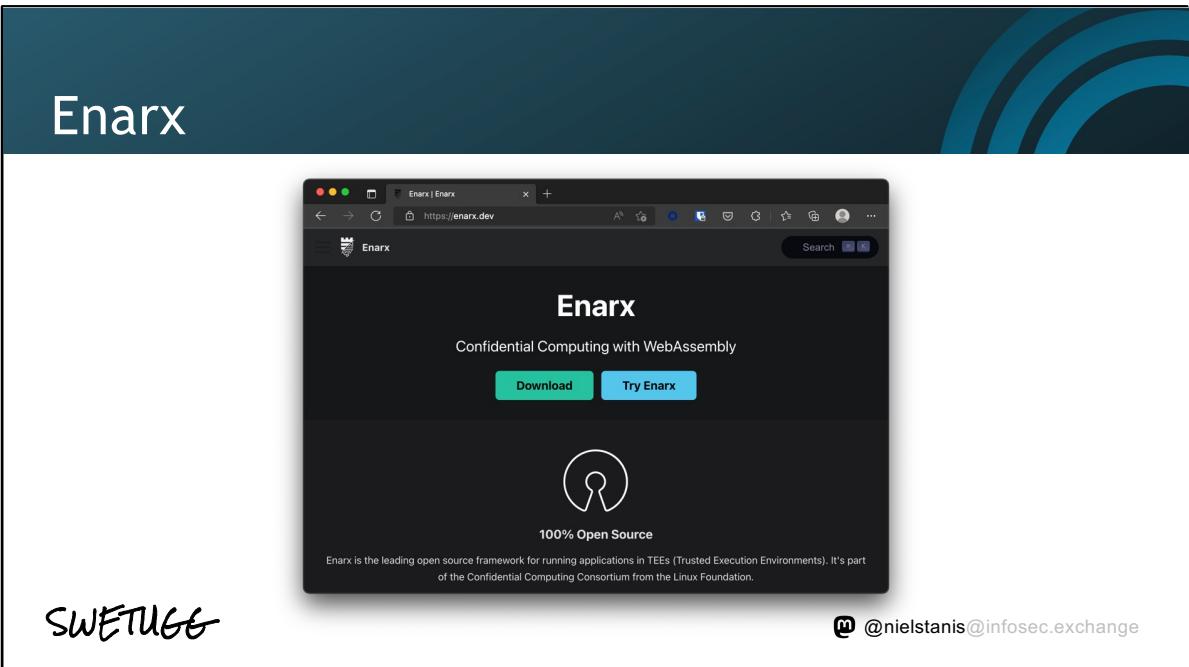
Trusted Computing - XKCD 2166

SWETUGS



@nielstanis@infosec.exchange

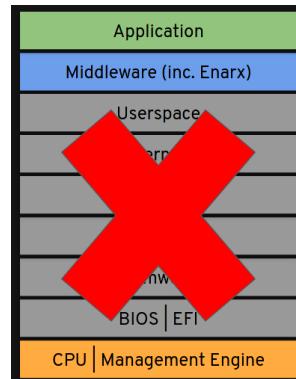
<https://xkcd.com/2166/>



<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

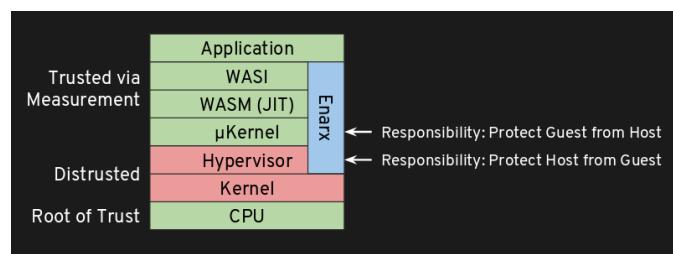


 @nielstanis@infosec.exchange

SWETUGS

Enarx

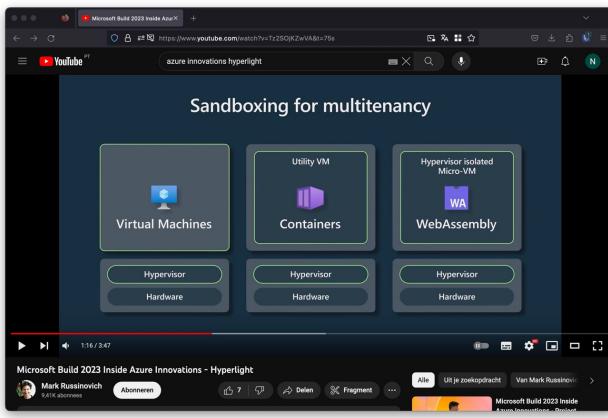
- Leverages Trusted Execution Environment (TEE) direct on processor
 - AMD's SEV, Intel's SGX and IBM's PEF
- Attestation of hardware and Enarx runtime



SWETUGS

@nielstanis@infosec.exchange

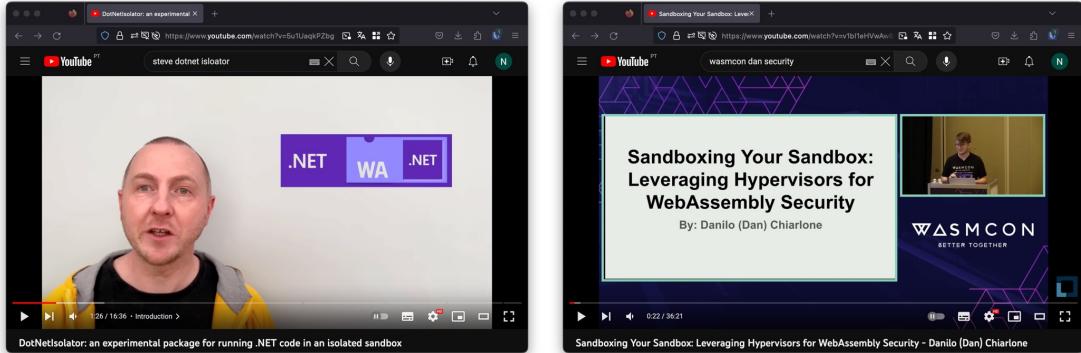
Project Hyperlight



SWETUGG

@nielstanis@infosec.exchange

DotNetIsolator & Project Hyperlight

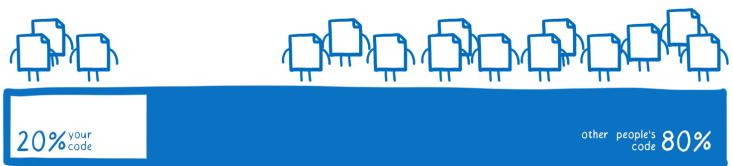


SWETUGG

 @nielstanis@infosec.exchange

WASM - What's next?

composition of an
average code base

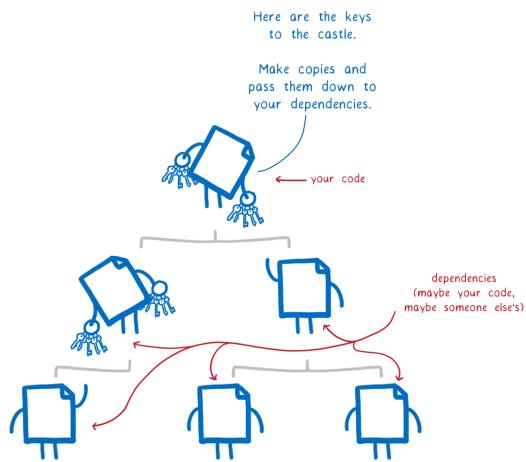


SWETUGG

 @nielstanis@infosec.exchange

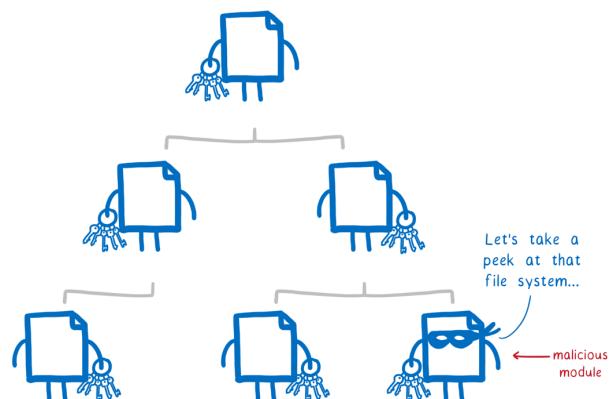
Dependencies

SWETUGG



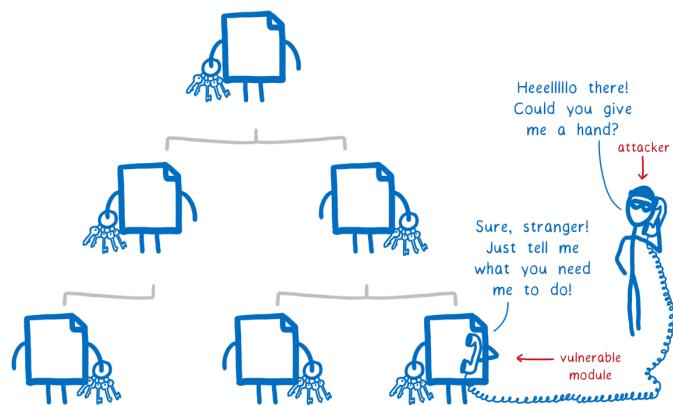
iis@infosec.exchange

Malicious module



@nielstanis@infosec.exchange

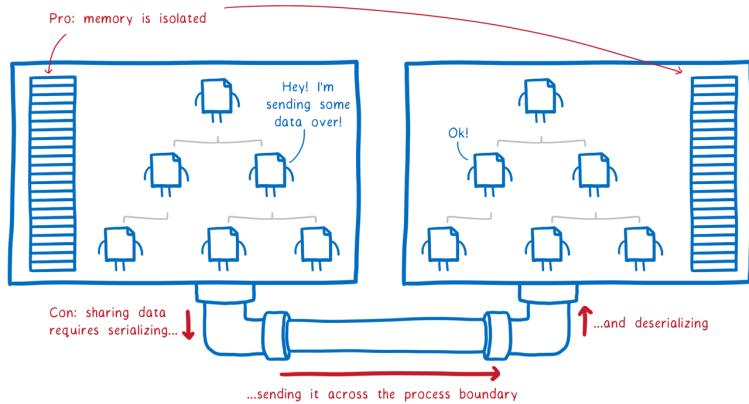
Vulnerable module



SWETUGG

@nielstanis@infosec.exchange

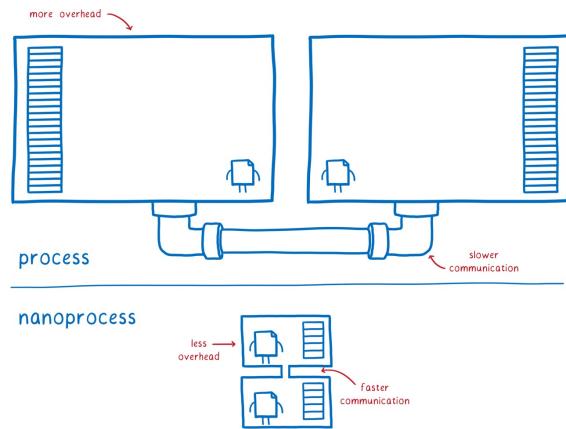
Process Isolation



SWETUGS

@nielstanis@infosec.exchange

WebAssembly Nano-Process

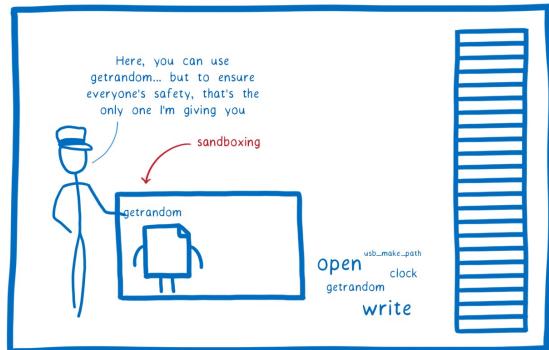


* not drawn to scale
@nielstanis@infosec.exchange

SWETUGS

WebAssembly Nano-Process

1. Sandboxing

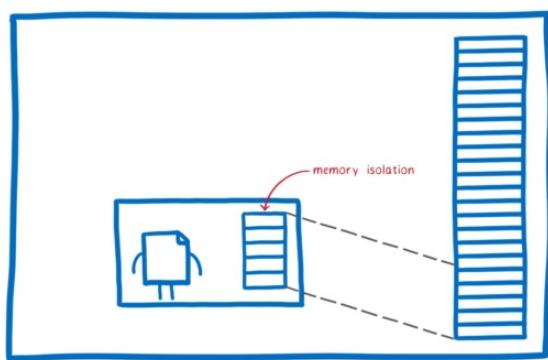


SWETUGS

@nielstanis@infosec.exchange

WebAssembly Nano-Process

2. Memory model

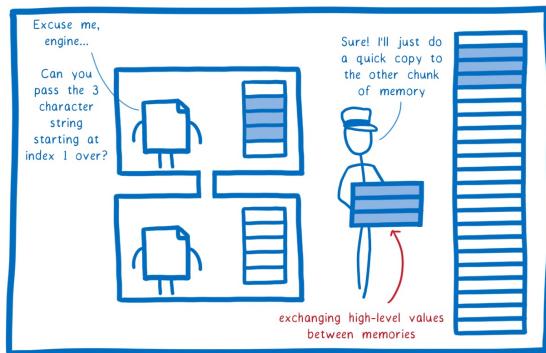


@nielstanis@infosec.exchange

SWETUGS

WebAssembly Nano-Process

3. Interface Types

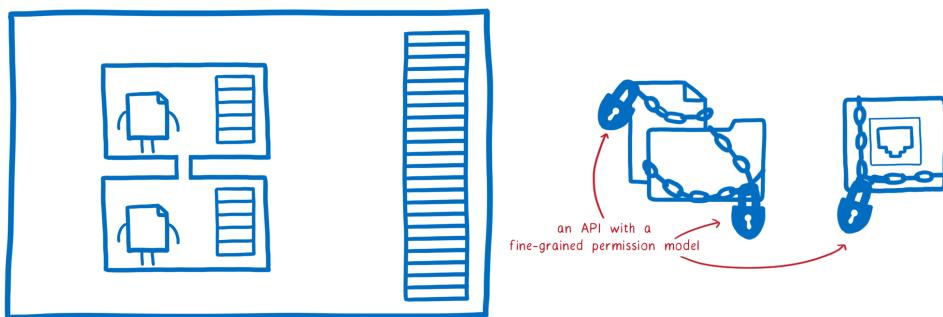


 @nielstanis@infosec.exchange

SWETUGS

WebAssembly Nano-Process

4. WebAssembly System Interface

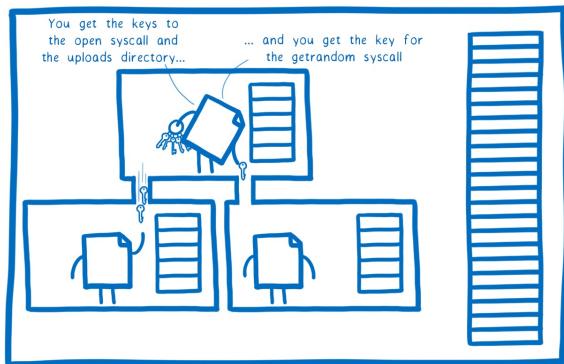


SWETUGG

 @nielstanis@infosec.exchange

WebAssembly Nano-Process

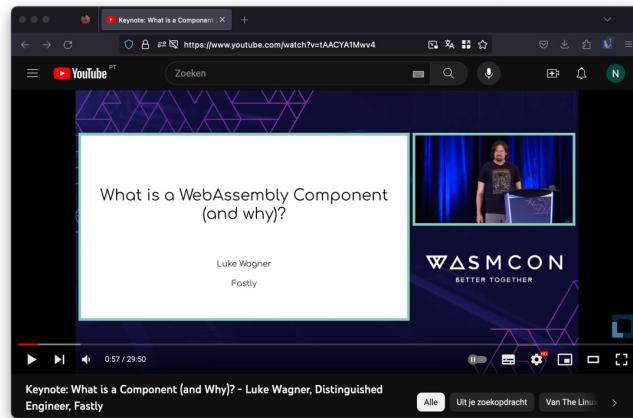
5. The missing link



@nielstanis@infosec.exchange

SWETUGG

WebAssembly Component Model



SWETUGG

@nielstanis@infosec.exchange

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

WASI Preview 2

The screenshot shows a web browser window with the title "WASI Preview 2 Launched - sunfishcode". The page content is from the blog "sunfishcode's blog" by sunfishcode. The main heading is "WASI Preview 2 Launched", followed by a timestamp "Posted on January 25, 2024". The text discusses the launch of WASI Preview 2 and its significance. A handwritten note "SWETUGG" is written vertically on the left side of the screenshot.

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!

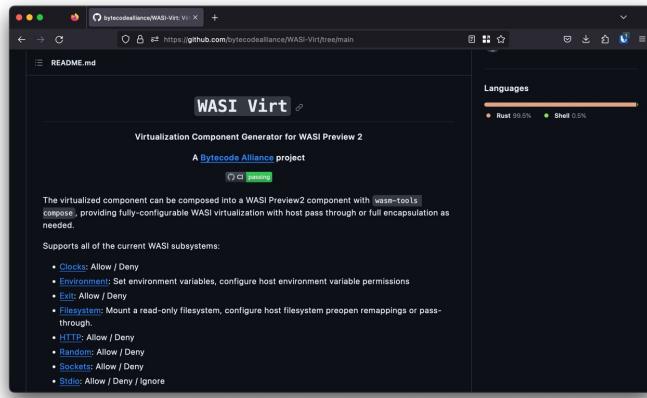
This is a major milestone! We made it! At the same time, the journey is only just beginning. But let's talk this moment to step back and look at what this means.

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

WASI Virt

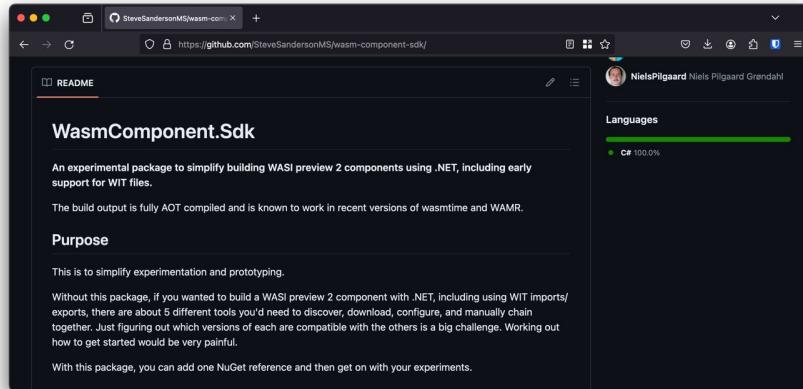


SWETUGG

@nielstanis@infosec.exchange

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

WasmComponent.SDK

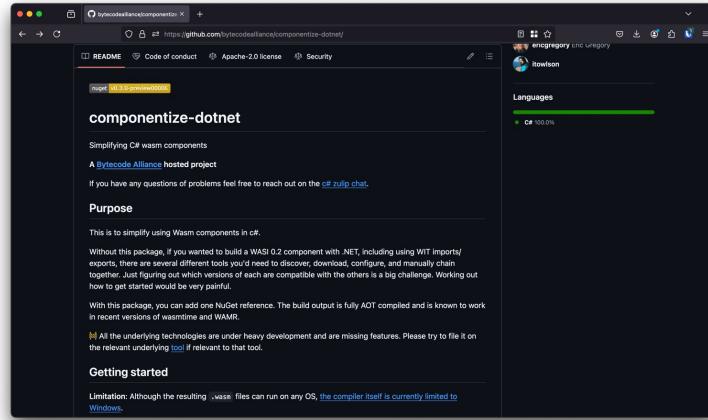


SWETUGG

@nielstanis@infosec.exchange

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

Componentize-dotnet

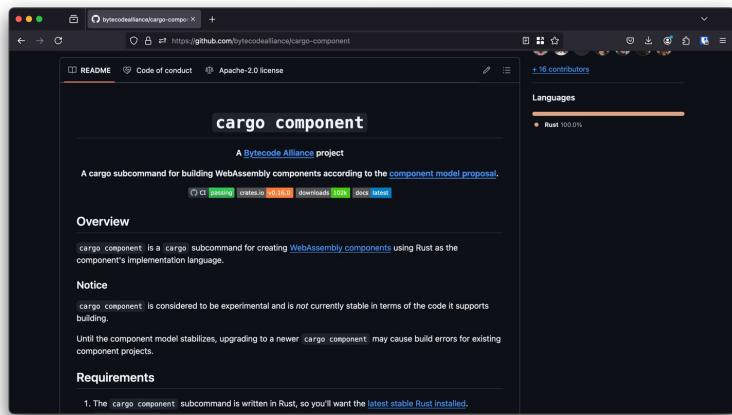


SWETUGS

@nielstanis@infosec.exchange

<https://github.com/bytocodealliance/componentize-dotnet/>

Demo WASI Preview 2 in Rust



SWETUGS

 @nielstanis@infosec.exchange

<https://github.com/bytocodealliance/cargo-component>

Runtimes and Security

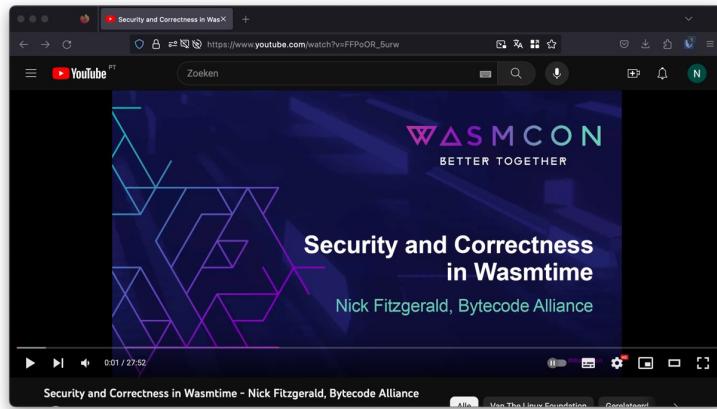
- Most security research published focusses on correctness of WASM runtimes/VM's
- Bytecode Alliance Blogpost September 2022:
 - "Security and Correctness in Wasmtime"
 - Written in Rust → Using all its LangSec features
 - Continues Fuzzing & formal verification
 - Security process & vulnerability disclosure

SWETUGS

 @nielstanis@infosec.exchange

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

Runtimes and Security



SWETUGS

 @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! componentize-dotnet good step forward for .NET ecosystem

SWETUGG

 @nielstanis@infosec.exchange

Questions?

- <https://github.com/nielstanis/swetugg2024wasm/>
- ntanis at Veracode.com
- @nielstanis@infosec.exchange
- <https://blog.fennec.dev>
- Tack! Thank you!

SWETUGG

 @nielstanis@infosec.exchange