



OWASP  
BeNeLux Days  
Conference  
2023

Using WebAssembly to run,  
extend, and secure your  
application

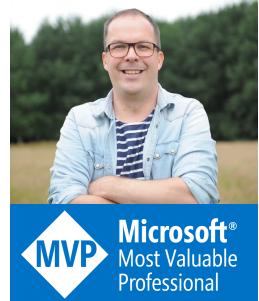
Niels Tanis



## 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**

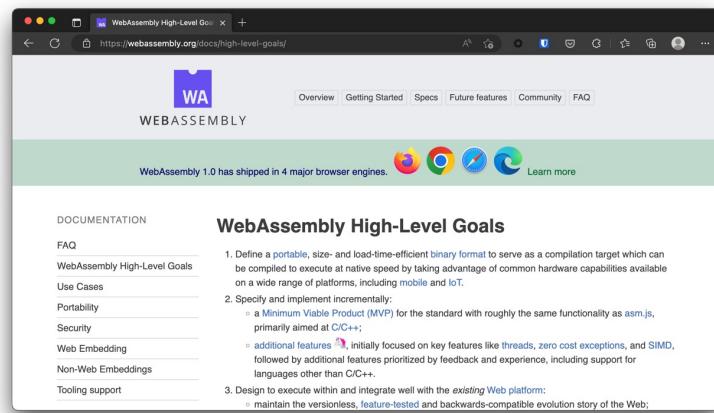


**MVP** Microsoft®  
Most Valuable  
Professional



[@nielstanis@infosec.exchange](mailto:@nielstanis@infosec.exchange)

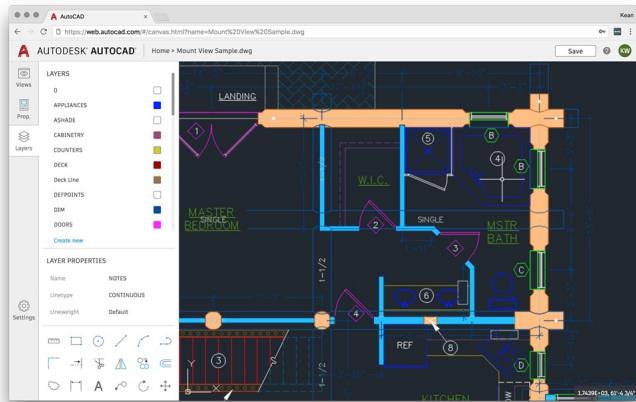
# WebAssembly



@nielstanis@infosec.exchange

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

# WebAssembly - AutoCAD



@nielstanis@infosec.exchange

# WebAssembly - SDK's

The screenshot shows a Medium article page. At the top, there's a navigation bar with a back arrow, forward arrow, and a search icon. Below it, a dark header bar has the title 'Introducing the Disney+ Application Development Kit (ADK)' and the author's name, Mike Hanley. The main content area features a large image of a person's face, followed by the article's title and a brief summary. Below the summary, there's a 'By' section with the author's name and title, Tom Schroeder, Sr. SWE / Technical Lead, Native Client Platform, Living Room Devices. There are also some engagement metrics like 415 likes and 4 comments.

The screenshot shows a blog post from Amazon Science. The header includes the Amazon logo and the category 'CLOUD AND SYSTEMS'. The main title is 'How Prime Video updates its app for more than 8,000 device types'. Below the title, there's a short blurb: 'The switch to WebAssembly increases stability, speed.' The post is attributed to Alexandru Enă and dated January 27, 2022. At the bottom, there's a note: '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
- Using WebAssembly to ...
  - ... run your application
  - ... extend your application
  - ... secure your application
- Conclusion
- Q&A



 [@nielstanis@infosec.exchange](mailto:@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

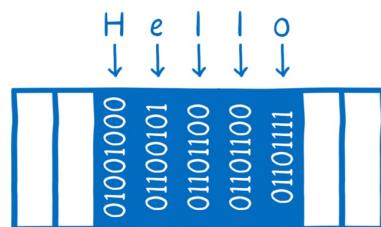


@nielstanis@infosec.exchange

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

# WebAssembly Memory

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



@nielstanis@infosec.exchange

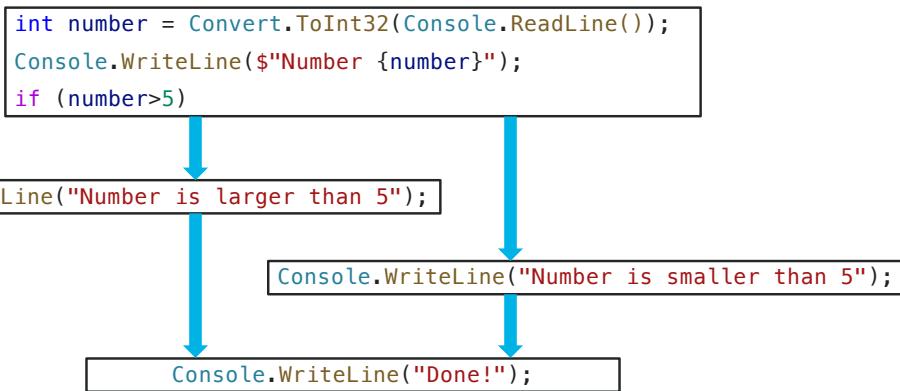
# WebAssembly Control-Flow Integrity

```
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!");
```



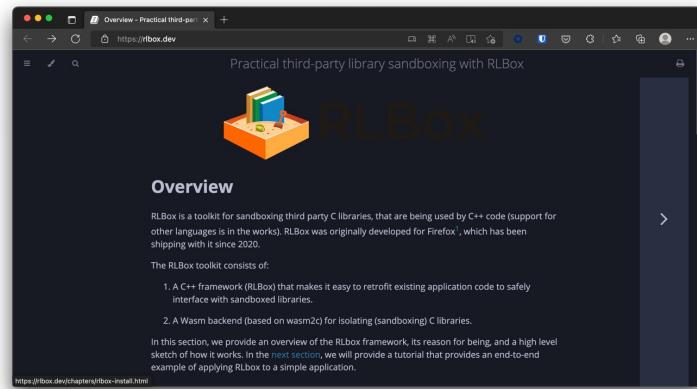
@nielstanis@infosec.exchange

# WebAssembly Control-Flow Integrity



@nielstanis@infosec.exchange

# FireFox RLBox

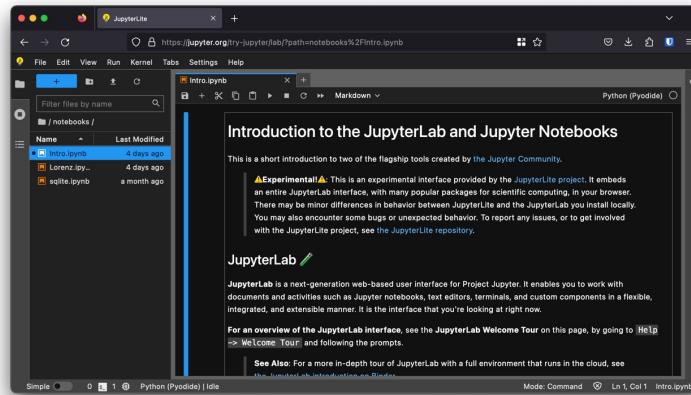


@nielstanis@infosec.exchange

<https://rlbox.dev/>

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

# Python on WASM



@nielstanis@infosec.exchange

# Python on WASM

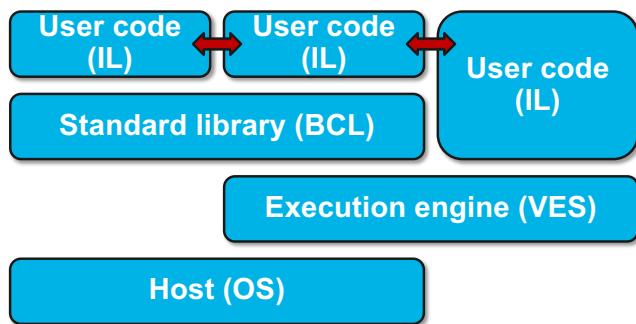
The screenshot shows a browser window with two tabs open. The left tab is a file explorer showing a directory structure with files like 'data' and 'notebooks'. The right tab is a Jupyter Notebook titled 'Inro.ipynb' containing a single cell with the text: 'Introduction to the JupyterLab and Jupyter Notebooks'. Below the browser window is a Network tab from a developer tools interface, showing a list of network requests. The requests include files from 'cdn.jsdelivr.net' and 'cdn.jsdelivr.net/pyodide/v0.24.1/full/pyodide.asm.wasm'. The Network tab has various columns: Status, Met., Domain, File, Initiator, Type, Transferred, Size, Headers, Cookies, Request, Response, Timings, and a Raw button.

Status	Met.	Domain	File	Initiator	Type	Transferred	Size	Headers	Cookies	Request	Response	Timings	Raw
200	GET	cdn.jsdelivr.net	Safe.js?v=2.7.5	MathJax.js!..js	js	service worker	6.2...						
200	GET	cdn.jsdelivr.net	568a1ff7157e4d8ab5f6e.js?v=def1ff	870.0f0ceb..js	js	service worker	6.0...						
200	GET	cdn.jsdelivr.net	fonddata.js?v=2.7.5	MathJax.js!..js	js	service worker	39.1...						
200	GET	cdn.jsdelivr.net	9511415b4c2fffe9eaaan0c.js?v=1415b4c2fffe9eaaan0c.js	568a1ff7157e4d8ab5f6e.js?v=def1ff	js	service worker	166...						
200	GET	cdn.jsdelivr.net	caret-down.svg	img	svg	service worker	257 B						
200	GET	cdn.jsdelivr.net	pyodide.js	9511415b4c2fffe9eaaan0c.js?v=1415b4c2fffe9eaaan0c.js	js	service worker	17.4...						
200	GET	cdn.jsdelivr.net	pyodide.asm.wasm	fetch	wasm	service worker	9 MB						
200	GET	cdn.jsdelivr.net	python_stubs.zip	fetch	wasm	service worker	8.8...						



@nielstanis@infosec.exchange

# Running .NET on WebAssembly

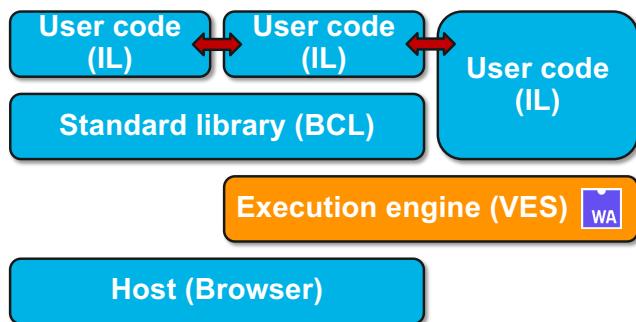


@nielstanis@infosec.exchange

Diagram:

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

# Running .NET on WebAssembly

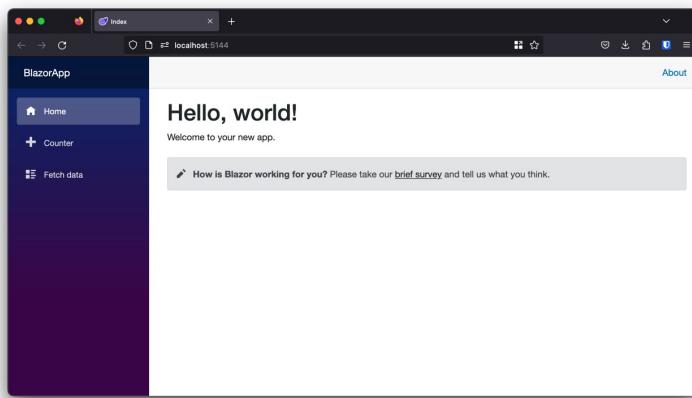


@nielstanis@infosec.exchange

Diagram:

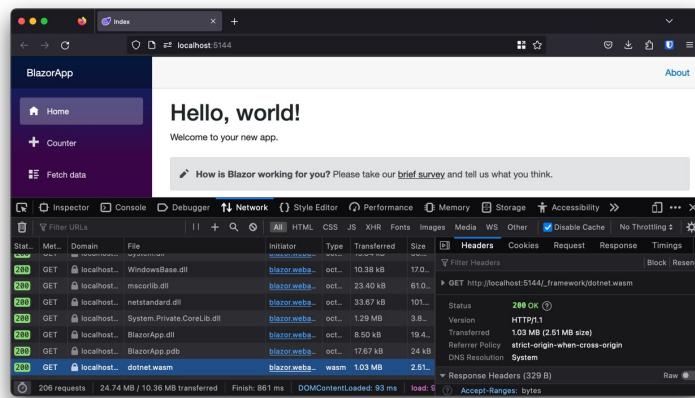
<https://github.com/itowlson/wasmday22/blob/main/slides/Wasm%20Interfaces%20and%20.NET.pptx>

# Blazor WebAssembly



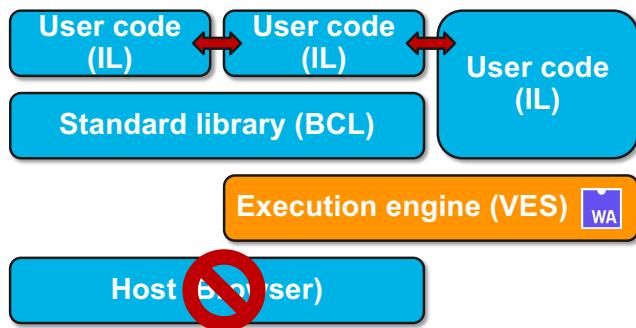
 @nielstanis@infosec.exchange

# 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](mailto:@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 familiar with .NET Standard? 😊



👤 @nielstanis@infosec.exchange

# 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) dated March 27, 2019. The tweet 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 March 27, 2019. The reply reads:

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/standa...  
D deze collectie weergeven

At the bottom of the tweet card, it says "9:39 p.m. - 27 mrt. 2019 · Twitter Web Client". To the right of the tweet card, there is a small profile icon and the handle "@nielstanis@infosec.exchange".



# Docker vs WASM & WASI

A screenshot of a Twitter browser window. The tweet is from Solomon Hykes (@solomonstre) dated March 27, 2019, at 4:50 a.m. 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 :)

The tweet has 56 Retweets, 5 Geciteerde Tweets, and 165 Vind-ik-leuks.

At the bottom right of the screenshot, there is a watermark with the OWASP logo and the text "@nielstanis@infosec.exchange".

# Docker & WASM

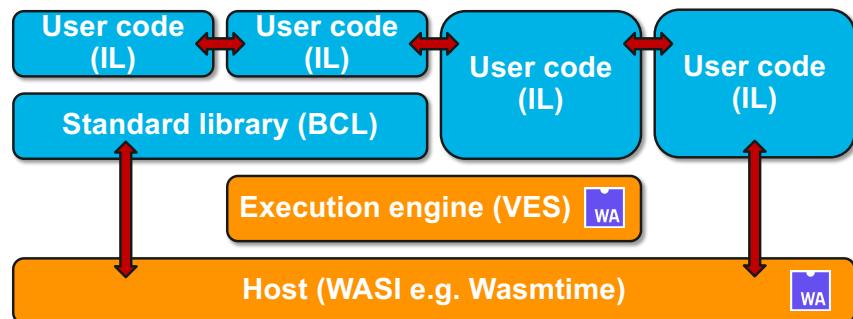
The image contains two side-by-side screenshots of a web browser displaying the Docker+Wasm Technical Preview page.

**Left Screenshot:** The title "Introducing the Docker+Wasm Technical Preview" is displayed prominently. Below it, there is a photo of Michael Irwin, his name, and the date "Oct 24 2022". A note at the bottom 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."

**Right Screenshot:** This screenshot shows a detailed architectural diagram. At the top is the "Docker Engine". Below it is a box labeled "containerd". Inside "containerd", there are three separate boxes, each containing a "runc" box and a "Container process" box. To the right of "containerd" is another box labeled "containerd-wasm shim", which contains a "wasm-edge" box and a "Wasm Module" box. Arrows indicate the flow from the Docker Engine down to containerd, and from containerd to each of the three runc boxes. Arrows also point from each runc box to its corresponding Container process box, and from the Container process box to the "wasm-edge" box within the containerd-wasm shim box.

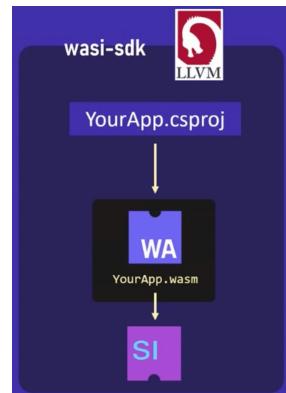
OWASP  @nielstanis@infosec.exchange

# WebAssembly System Interface WASI



@nielstanis@infosec.exchange

# Experimental WASI SDK for .NET



@nielstanis@infosec.exchange

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

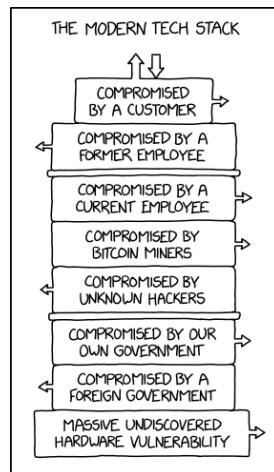
## Extending .NET with WASM

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



[@nielstanis@infosec.exchange](mailto:@nielstanis@infosec.exchange)

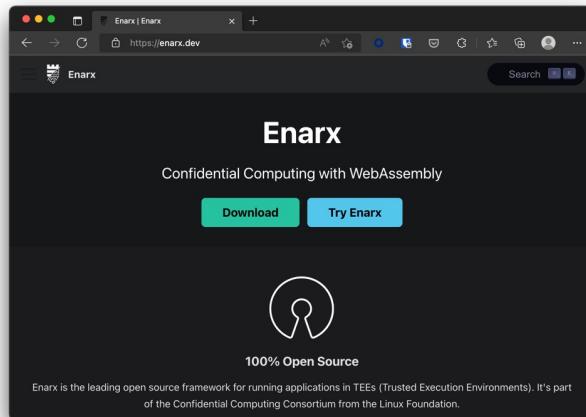
# Trusted Computing - XKCD 2166



@nielstanis@infosec.exchange

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

# Enarx

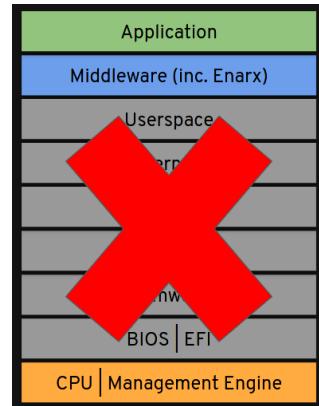


[@nielstanis@infosec.exchange](mailto:@nielstanis@infosec.exchange)

<https://enarx.dev/>

## Enarx Threat Model

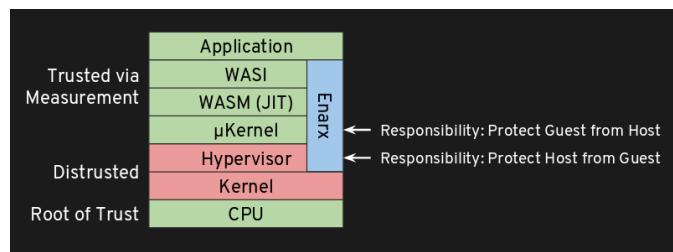
- Don't trust the host
- Don't trust the host owner
- Don't trust the host operator
- Hardware cryptographically verified
- Software audited and cryptographically verified



@nielstanis@infosec.exchange

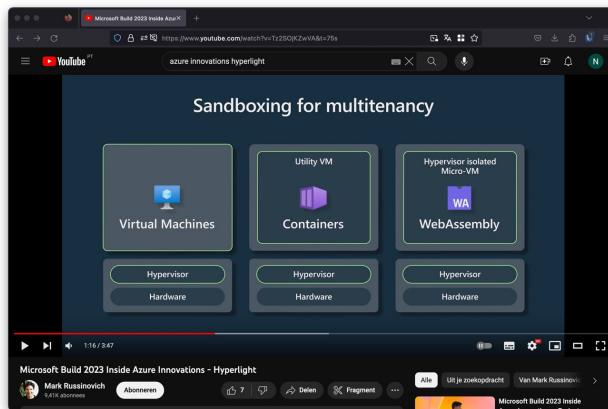
# Enarx

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



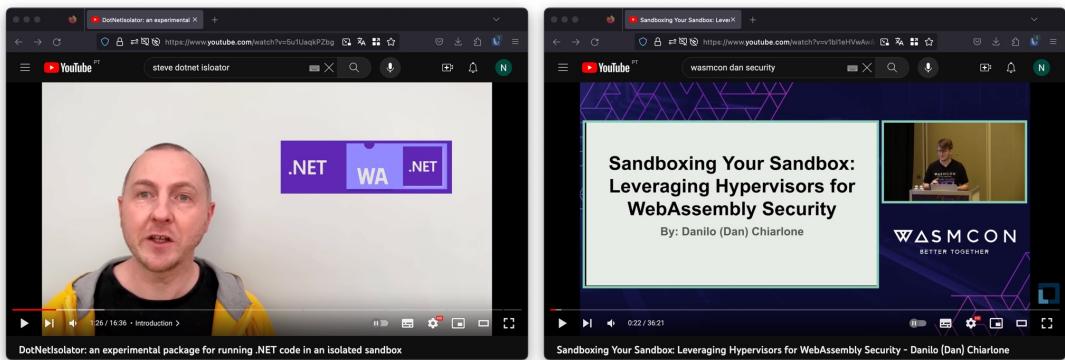
@nielstanis@infosec.exchange

# Project Hyperlight



@nielstanis@infosec.exchange

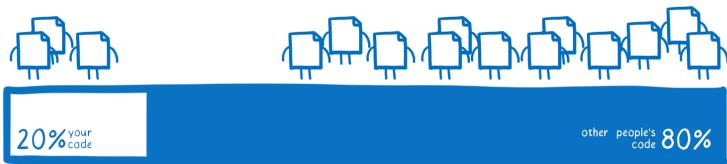
# DotNetIsolator & Project Hyperlight



@nielstanis@infosec.exchange

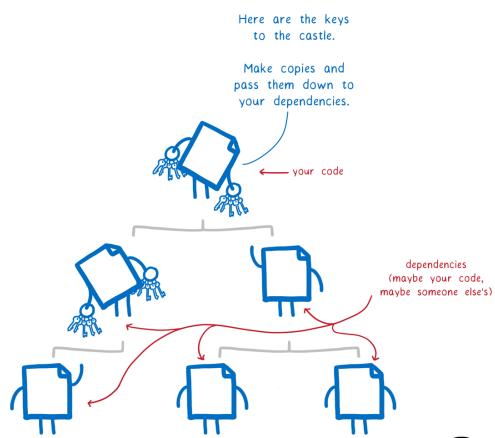
## WASM - What's next?

composition of an  
average code base



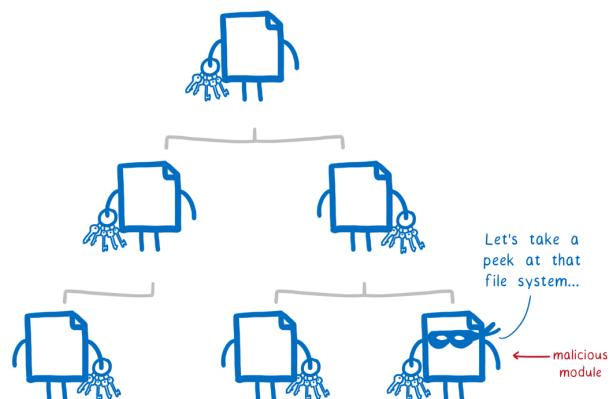
@nielstanis@infosec.exchange

# Dependencies



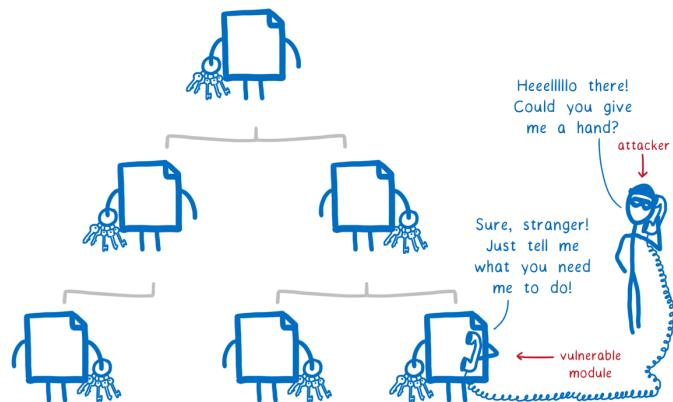
@nielstanis@infosec.exchange

## Malicious module



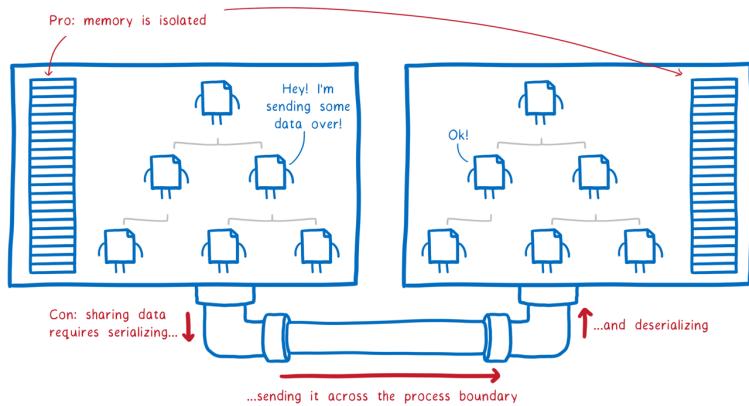
@nielstanis@infosec.exchange

# Vulnerable module



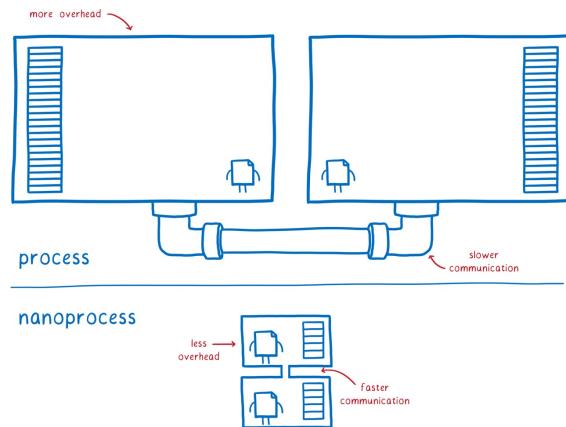
@nielstanis@infosec.exchange

# Process Isolation



@nielstanis@infosec.exchange

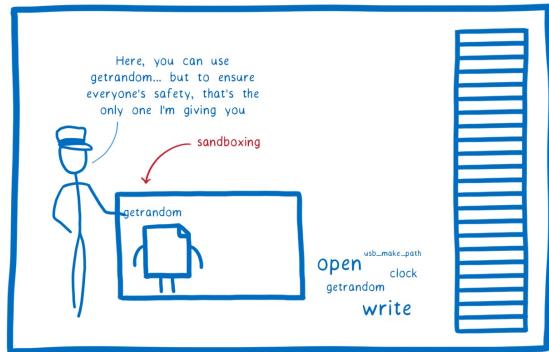
# WebAssembly Nano-Process



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

# WebAssembly Nano-Process

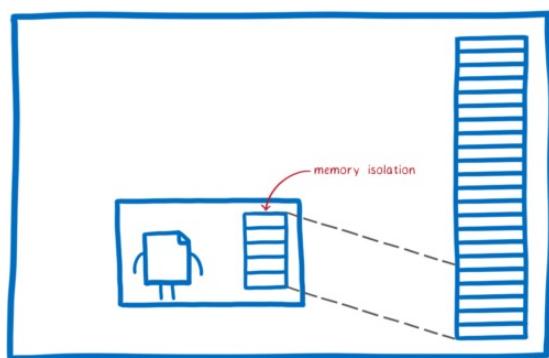
## 1. Sandboxing



@nielstanis@infosec.exchange

# WebAssembly Nano-Process

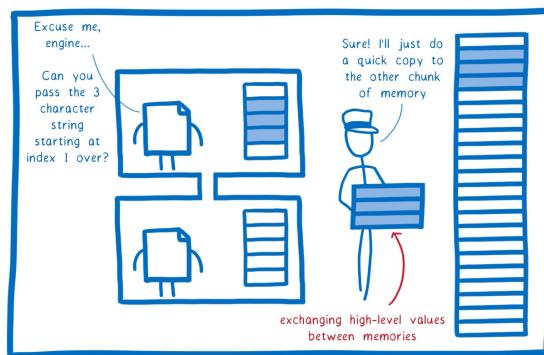
## 2. Memory model



@nielstanis@infosec.exchange

# WebAssembly Nano-Process

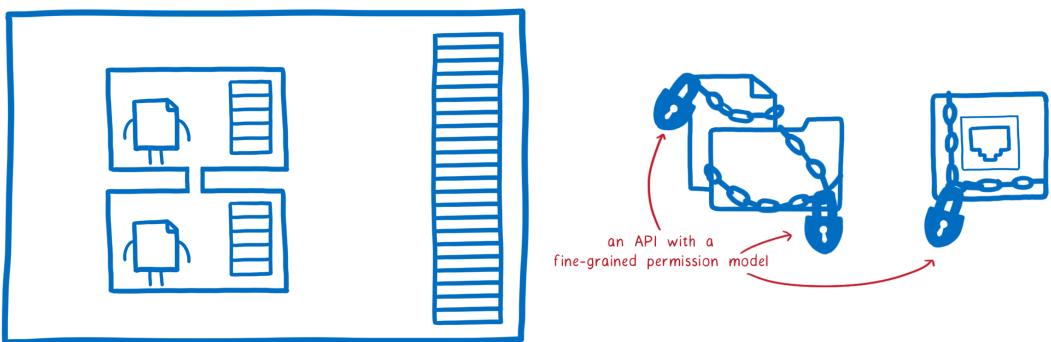
## 3. Interface Types



@nielstanis@infosec.exchange

# WebAssembly Nano-Process

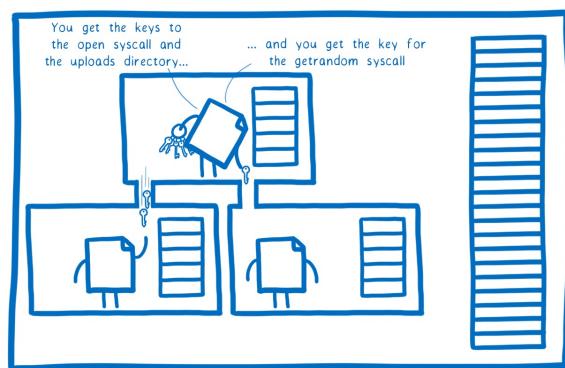
## 4. WebAssembly System Interface



@nielstanis@infosec.exchange

# WebAssembly Nano-Process

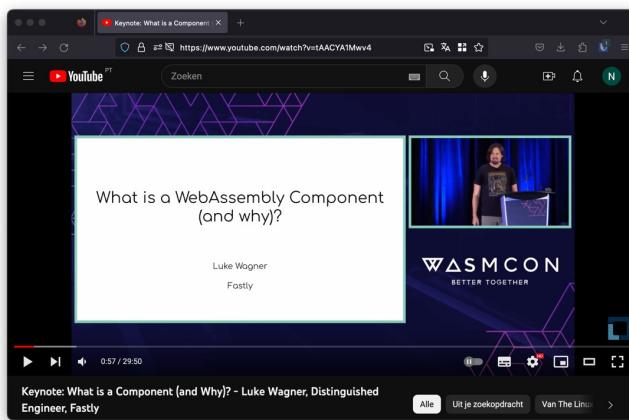
## 5. The missing link



 @nielstanis@infosec.exchange



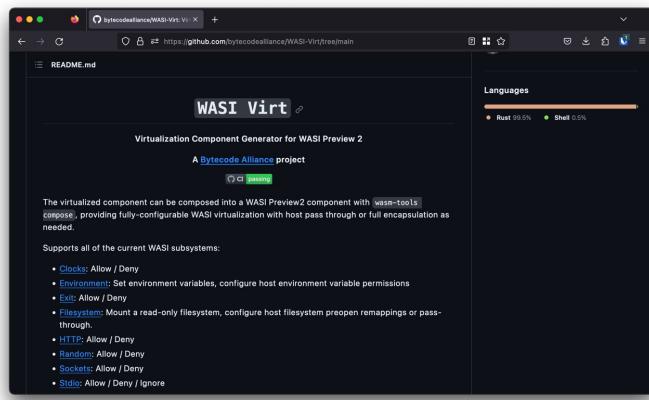
# WebAssembly Component Model



@nielstanis@infosec.exchange

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

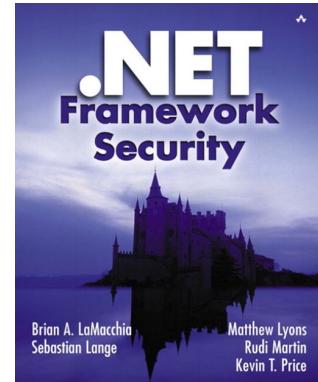
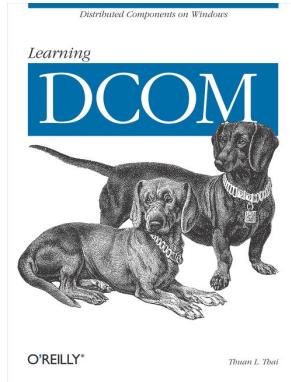
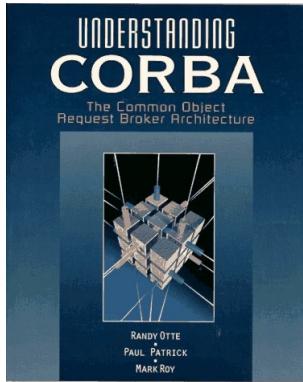
# WebAssembly Component Model



 @nielstanis@infosec.exchange

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

# Have we seen this before?



 [@nielstanis@infosec.exchange](mailto:@nielstanis@infosec.exchange)

## Runtimes and Security

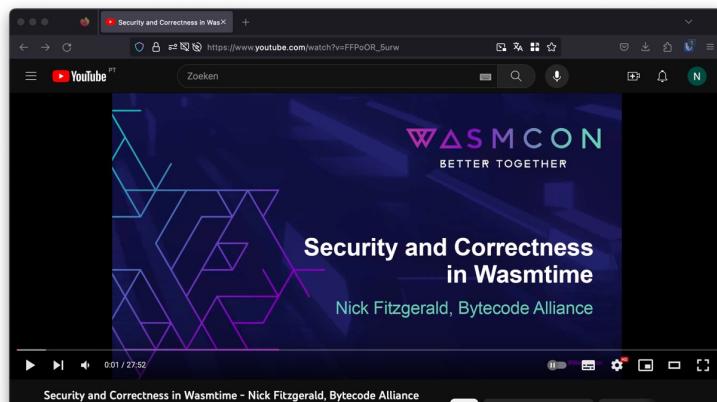
- Most security research published focusses on correctness of WASM runtimes/VM's based on e.g. fuzzing
- Bytecode Alliance Blogpost September 2022 by Nick Fitzgerald:
  - “Security and Correctness in Wasmtime”
  - Written in Rust → Using all it's LangSec features
  - Continues Fuzzing & formal verification
  - Security process & vulnerability disclosure



 @nielstanis@infosec.exchange

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

# Runtimes and Security



@nielstanis@infosec.exchange

[https://www.youtube.com/watch?v=FFPoOR\\_5urw](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!
- I like top-down approach Bytecode Alliance is taking in moving forward, feedback will change/influence



 @nielstanis@infosec.exchange

## Questions?

- <https://github.com/nielstanis/owaspbenelux2023>
- ntanis at Veracode.com
- @nielstanis@infosec.exchange
- <https://blog.fennec.dev>
- Merci! Bedankt!



 [@nielstanis@infosec.exchange](mailto:nielstanis@infosec.exchange)