



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



WebAssembly

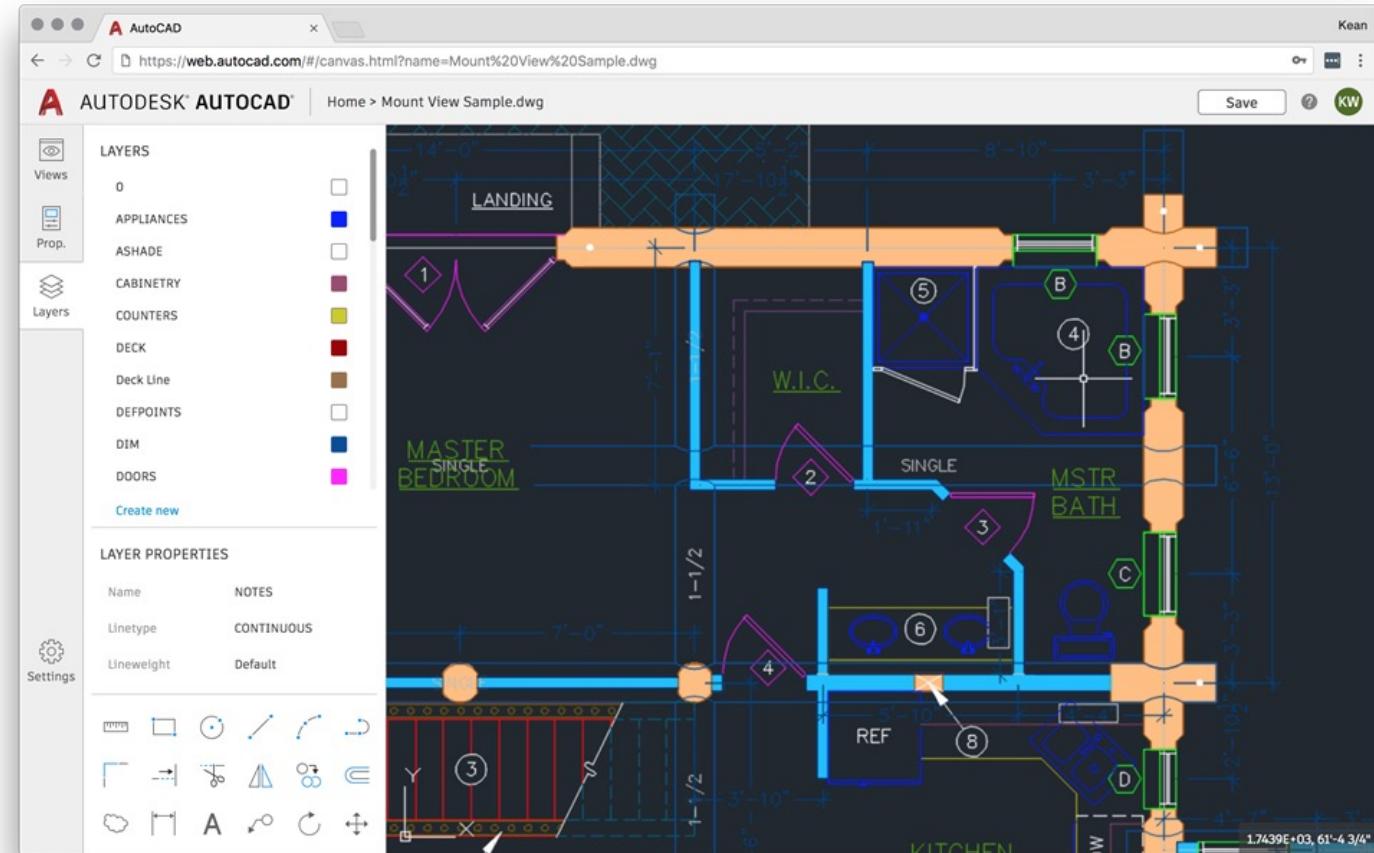
0101
0101

The screenshot shows the official WebAssembly website at <https://webassembly.org>. The page features a large purple 'WA' logo and the word 'WEBASSEMBLY'. A navigation bar includes links for Overview, Getting Started, Specs, Feature Extensions, Community, and FAQ. Below the navigation, a green banner states 'WebAssembly 1.0 has shipped in 4 major browser engines.' followed by icons for Firefox, Chrome, Safari, and Edge, with a 'Learn more' link. A main text block explains that WebAssembly (abbreviated *Wasm*) is a binary instruction format for a stack-based virtual machine, designed as a portable compilation target for programming languages. At the bottom, a yellow box contains developer reference information about Wasm, mentioning MDN's WebAssembly pages, the W3C Community Group, and the W3C Working Group.





WebAssembly - AutoCAD



@nielstanis@infosec.exchange

0101
0101

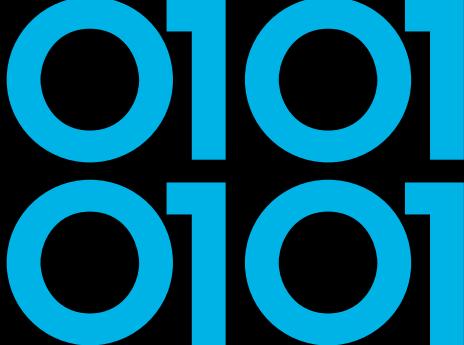
WebAssembly - SDK's

A screenshot of a Medium article page. The title is "Introducing the Disney+ Application Development Kit (ADK)" by Mike Hanley. The article was published on Sep 8, 2021, and has a 10 min read time. It features a profile picture of Mike Hanley, social sharing icons (Twitter, Facebook, LinkedIn), and a "Get started" button. The content discusses the Disney+ Application Development Kit (ADK) and its benefits.

A screenshot of a Medium article page. The title is "How Prime Video updates its app for more than 8,000 device types" by Alexandru Ene. The article is part of the "CLOUD AND SYSTEMS" series. It discusses the switch to WebAssembly and its benefits. The content mentions Prime Video's delivery to millions of customers on various devices.



@nielstanis@infosec.exchange



Agenda

- Introduction
- WebAssembly 101
- Running .NET on WebAssembly
- Extending .NET with WebAssembly
- Securing .NET with WebAssembly
- Conclusion
- Q&A





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



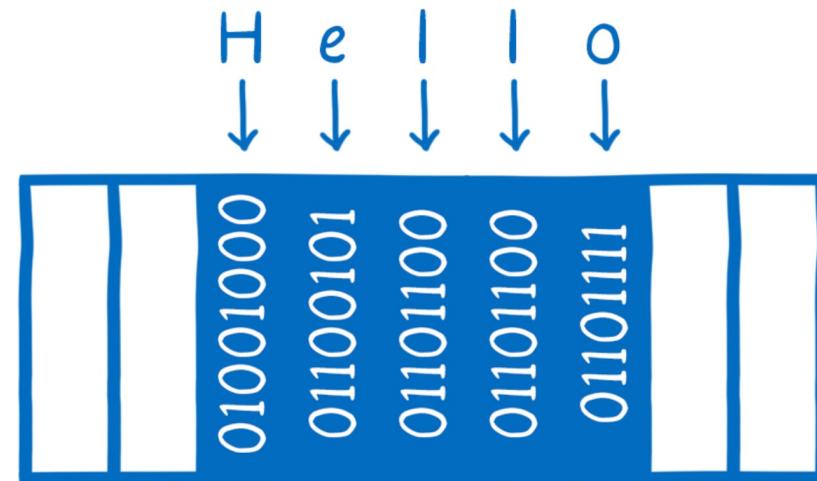
WEBASSEMBLY



0101
0101

WebAssembly Memory

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





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!");
```





WebAssembly Control-Flow Integrity

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

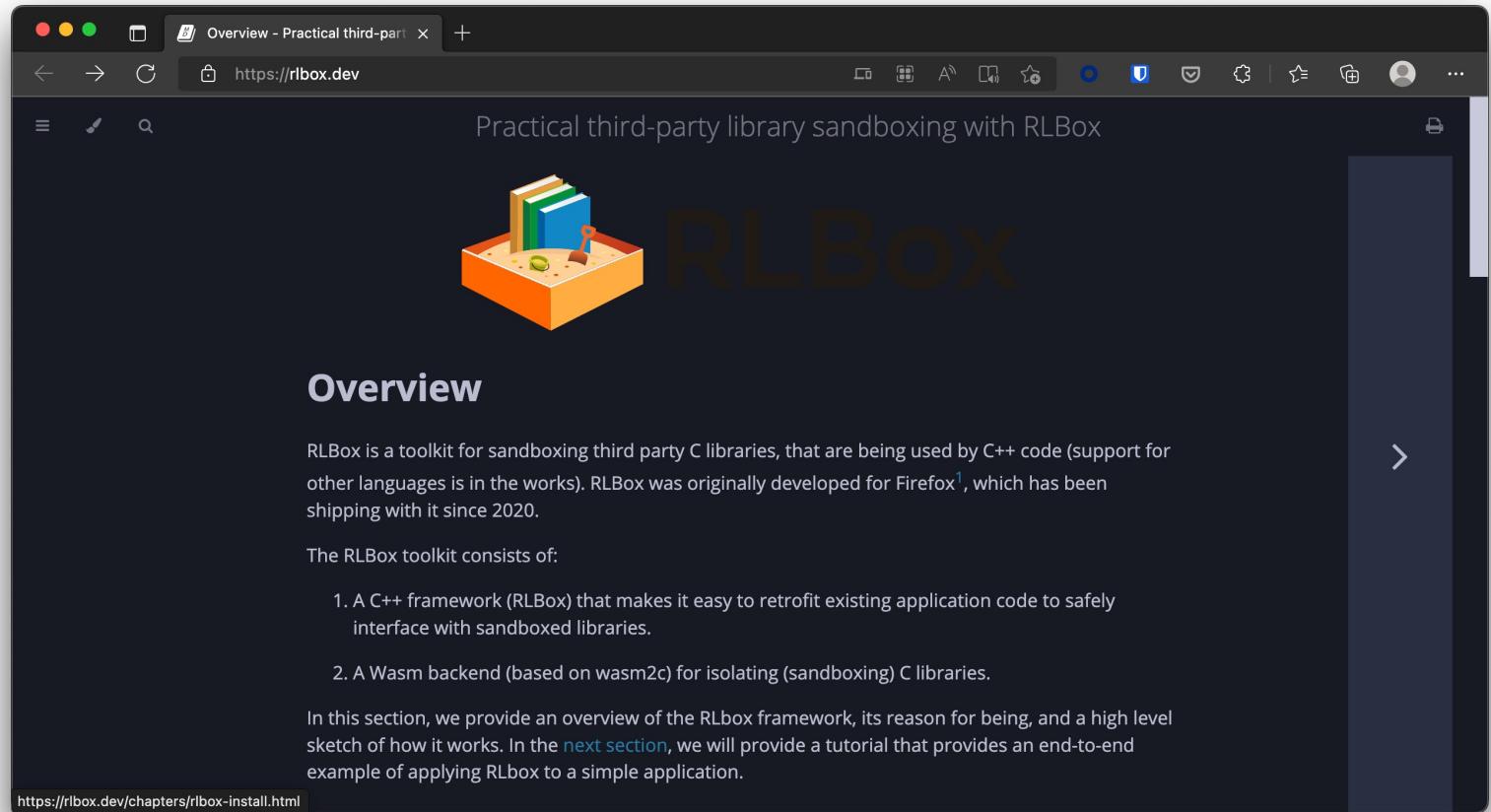
```
Console.WriteLine("Number is larger than 5");
```

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

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

0101
0101

FireFox RLBox



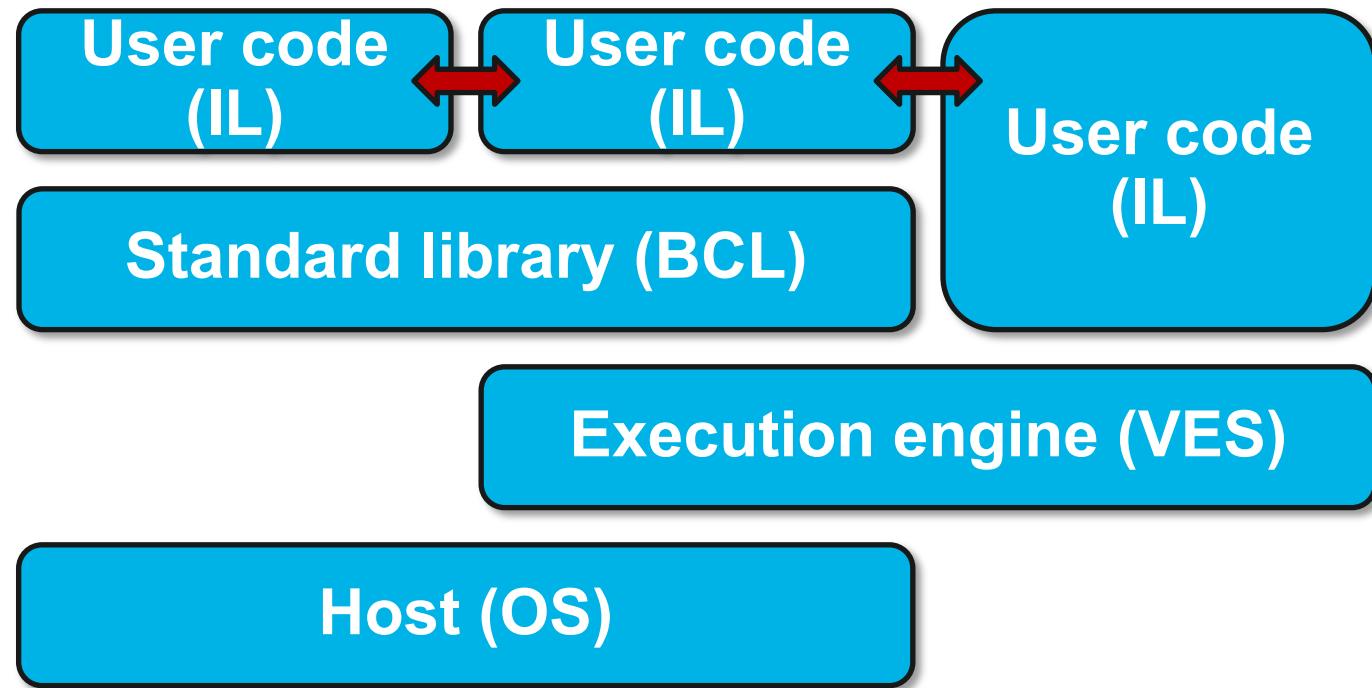
The screenshot shows a Firefox browser window with a dark theme. The title bar says "Overview - Practical third-part" and the address bar shows "https://rlbox.dev". The main content area has a dark background with a large orange sandcastle icon containing three books and a shovel. To the right of the icon, the word "RLBox" is written in a large, bold, dark font. Below the icon, the word "Overview" is centered in a white font. The text below "Overview" describes RLBox as a toolkit for sandboxing third-party C libraries. It mentions support for C++ and other languages, its development for Firefox since 2020, and its components: a C++ framework and a Wasm backend. A link at the bottom leads to "rlbox.dev/chapters/rlbox-install.html".

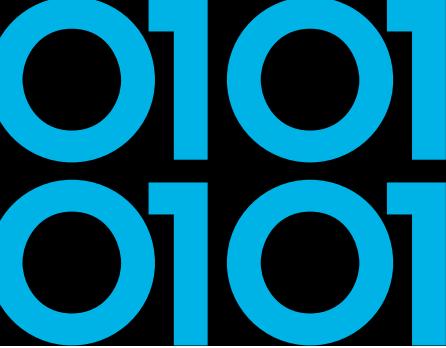


 @nielstanis@infosec.exchange

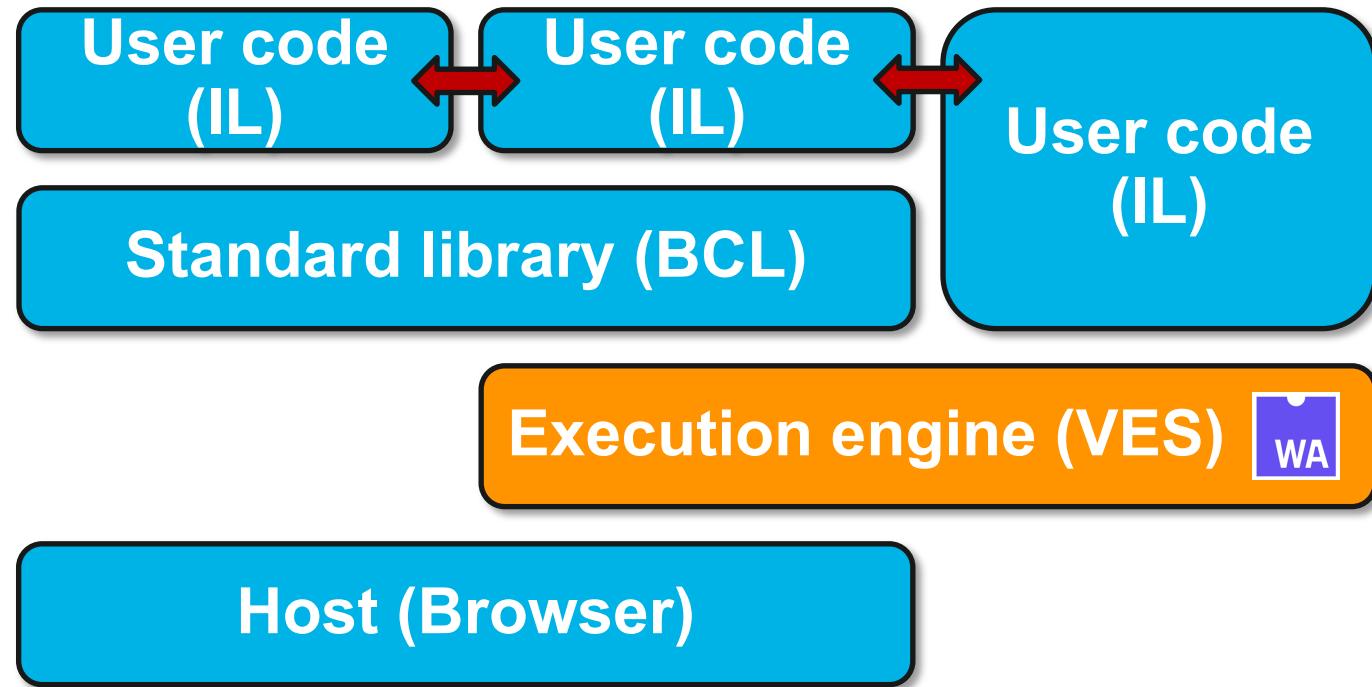


Running .NET on WebAssembly



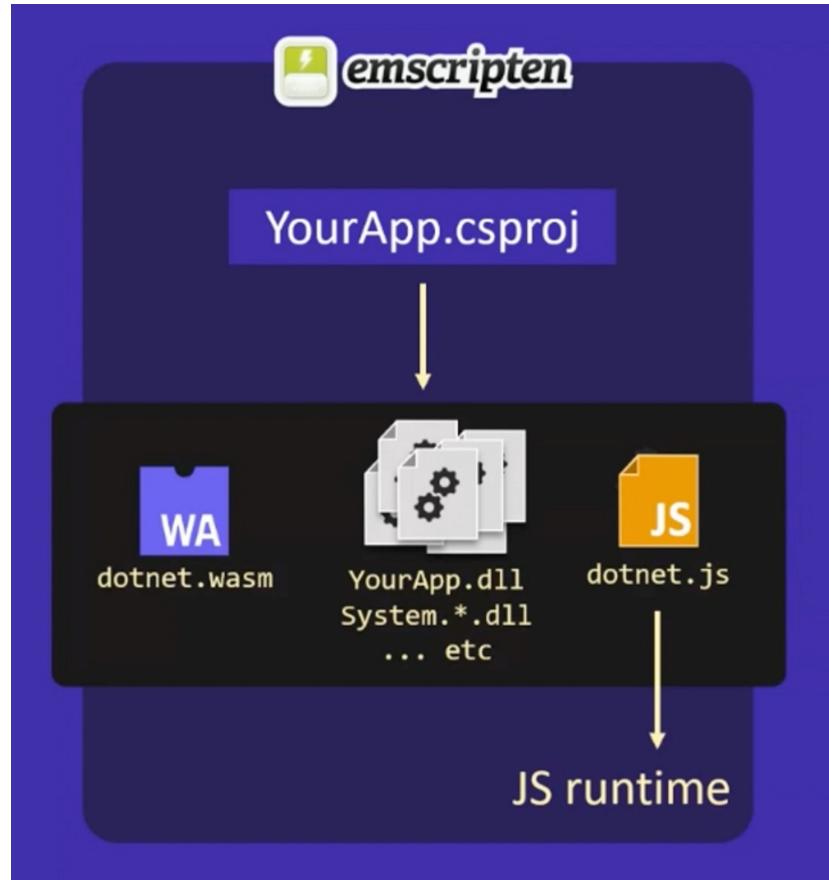


Running .NET on WebAssembly



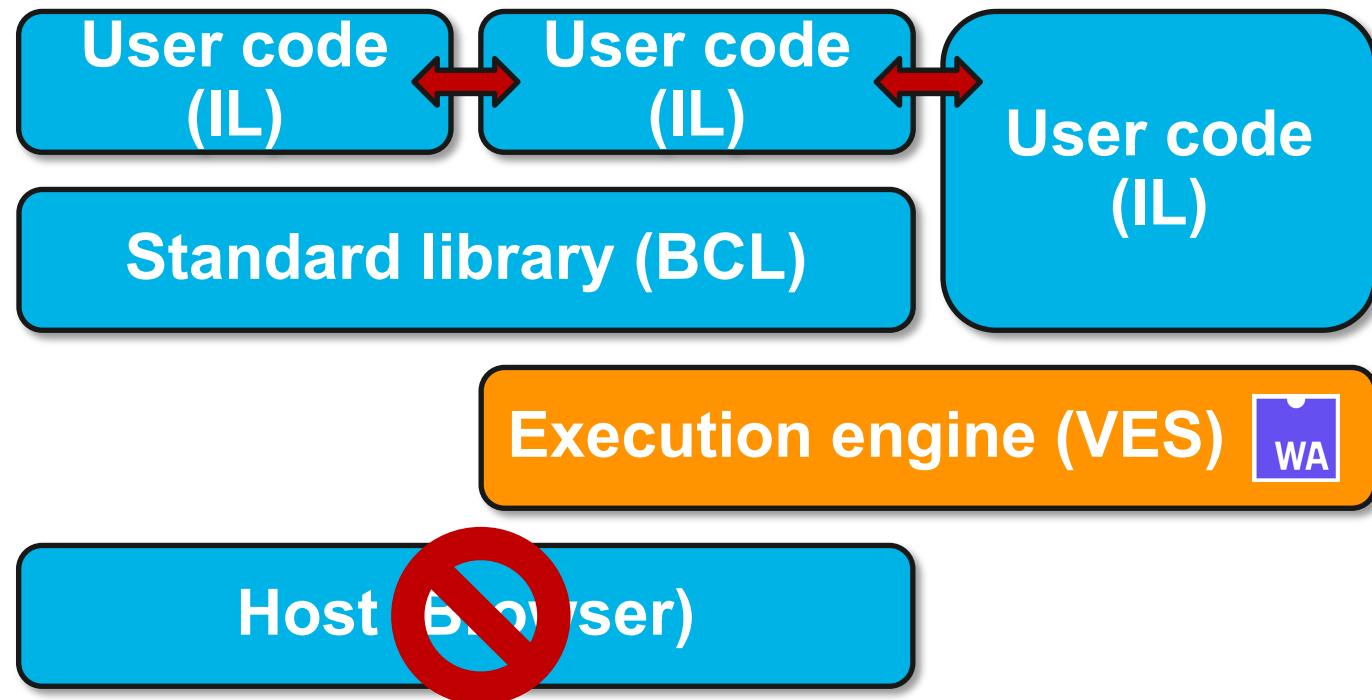
0101
0101

Blazor WebAssembly





Running .NET on WebAssembly



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



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? ☺



0101
0101

Docker vs WASM & WASI

A screenshot of a Twitter web client window. The URL in the address bar is <https://twitter.com/s...>. The 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...

Announcing WASI: A system interface for running WebAssembly outside the web (and inside it too)

hacks.mozilla.org/2019/03/standa...

[Deze collectie weergeven](#)

At the bottom of the tweet card, it says 9:39 p.m. · 27 mrt. 2019 · Twitter Web Client.



0101
0101

Docker vs WASM & WASI

A screenshot of a Twitter web application window. The tweet is from Solomon Hykes (@solomonstre) dated March 27, 2019. The tweet text is: "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 includes a reply from Solomon Hykes (@solomonstre) dated March 28, 2019, stating: "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...](https://twitter.com/linclark/status/110881411111111111)".

At the bottom of the tweet card, there is a link "Deze collectie weergeven".

Below the tweet card, the timestamp is 4:50 a.m. · 28 mrt. 2019 · Twitter Web App. The engagement metrics are 56 Retweets, 5 Geciteerde Tweets, and 165 Vind-ik-leuks.



0101
0101

Docker & WASM

The screenshot shows a web browser window with the title "Introducing the Docker+Wasm Technical Preview". The page features a purple header bar with the text "Wasm is a fast, light alternative to Linux containers — try it out today in the Docker+Wasm Technical Preview". Below this is the Docker+Wasm logo. The main content area has a dark blue background with the heading "Introducing the Docker+Wasm Technical Preview" in large white font. At the bottom left is a circular profile picture of Michael Irwin, followed by his name "MICHAEL IRWIN" and the date "Oct 24 2022". A text block 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."

The screenshot shows a web browser window with the same title as the first screenshot. It displays a diagram illustrating the Docker+Wasm architecture. The diagram shows the "Docker Engine" at the top, which interacts with a "containerd" component. The "containerd" component oversees three separate runtime environments: "containerd-shim", "containerd-shim", and "containerd-wasm-shim". Each "containerd-shim" environment runs a "runc" process, which in turn manages a "Container process". The "containerd-wasm-shim" environment runs a "wasmedge" process, which manages a "Wasm Module". Arrows indicate the flow of control and communication between these components.

Let's look at an example!

After installing the preview, we can run the following command to start an example Wasm application:

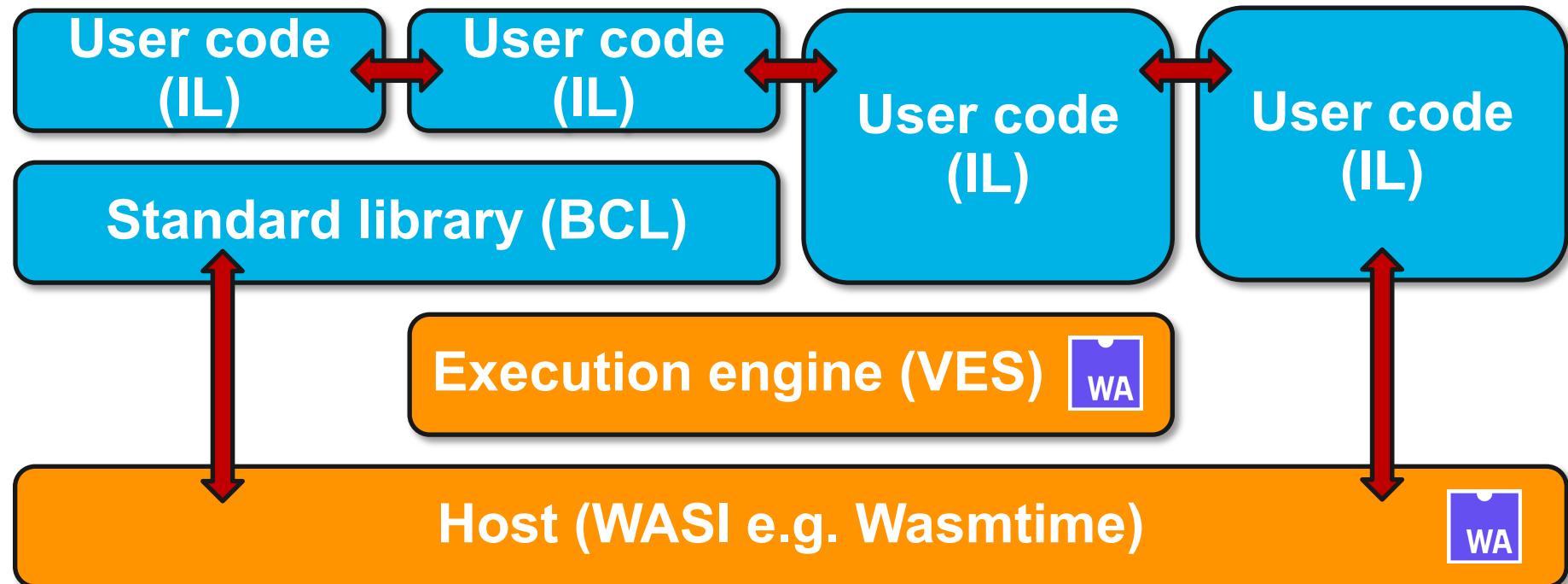
```
docker run -dp 8080:8080 --name=wasm-example --runtime=io.containerd.wasmedge.v1 --platform=wasi/wasm32 michaelirwin244/wasm-example
```



@nielstanis@infosec.exchange



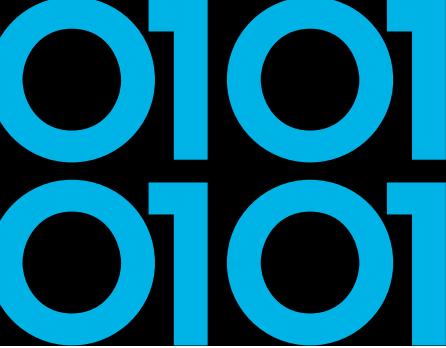
WebAssembly System Interface WASI





Experimental WASI SDK for .NET





.NET 8 WASI-Experimental

The screenshot shows a web browser window displaying a Microsoft DevBlog post titled "Extending WebAssembly to the Cloud". The main heading is "wasi-experimental workload". The text explains that .NET 8 includes a new workload called "wasi-experimental" which builds on Wasm functionality used by Blazor, extending it to run in "wasmtime" and invoke WASI interfaces. It notes that while not fully developed, it already provides useful functionality.

Below the heading, there's a section titled "Let's move on from theory to demonstrating the new capabilities." It instructs users to install the ".NET 8 SDK" and then the "wasi-experimental" workload. A command-line snippet is shown:

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

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

It also mentions the need to install "wasmtime" to run the generated Wasm code.

Finally, it suggests trying a simple example with the "wasi-console" template, showing the following command-line steps:

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





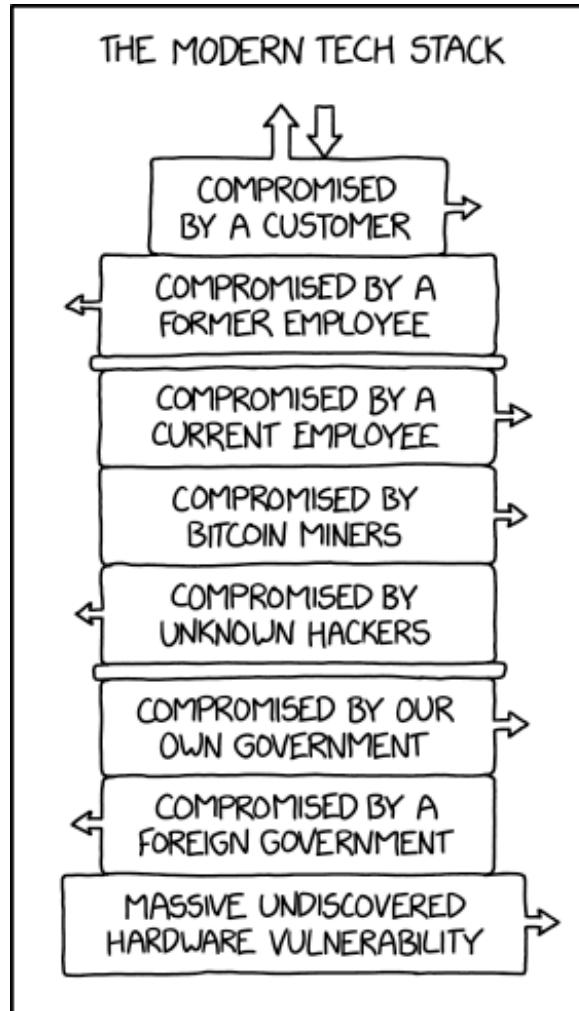
Extending .NET with WASM

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



0101
0101

Trusted Computing - XKCD 2166



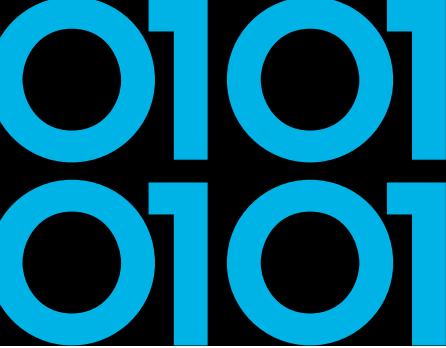
0101
0101

Enarx

The screenshot shows a dark-themed web browser window displaying the Enarx homepage at <https://enarx.dev>. The page features a large "Enarx" logo at the top center, followed by the tagline "Confidential Computing with WebAssembly". Below the tagline are two prominent buttons: a green "Download" button and a blue "Try Enarx" button. In the center of the page is a large, stylized white icon resembling a keyhole or a circular lock. At the bottom of the main content area, the text "100% Open Source" is displayed. A footer note at the very bottom states: "Enarx is the leading open source framework for running applications in TEEs (Trusted Execution Environments). It's part of the Confidential Computing Consortium from the Linux Foundation."

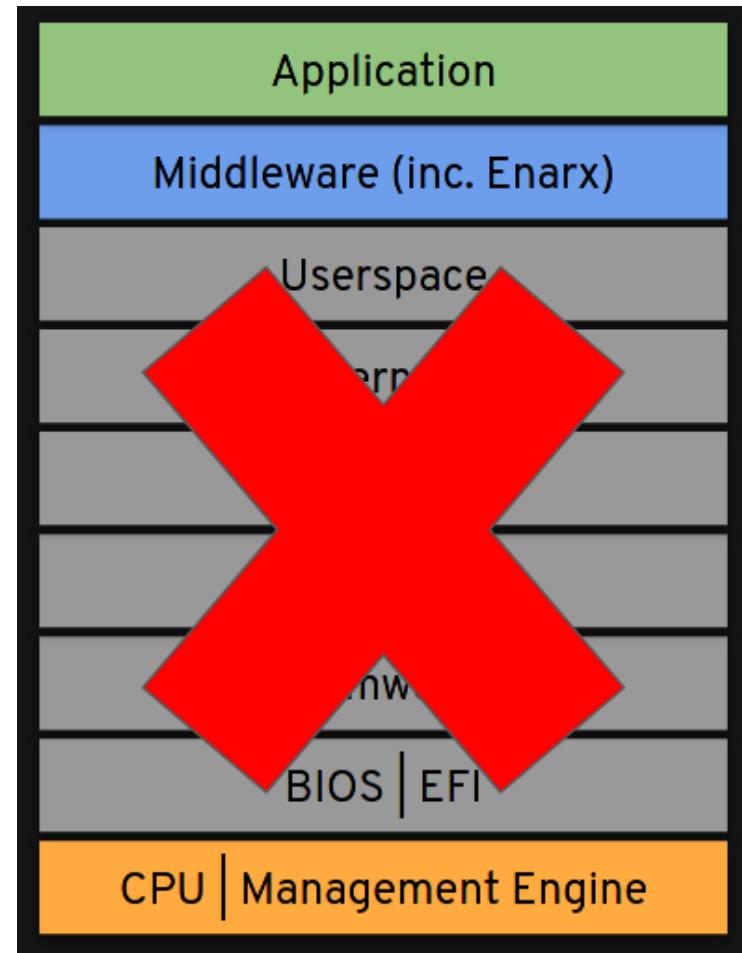


 @nielstanis@infosec.exchange



Enarx Threat Model

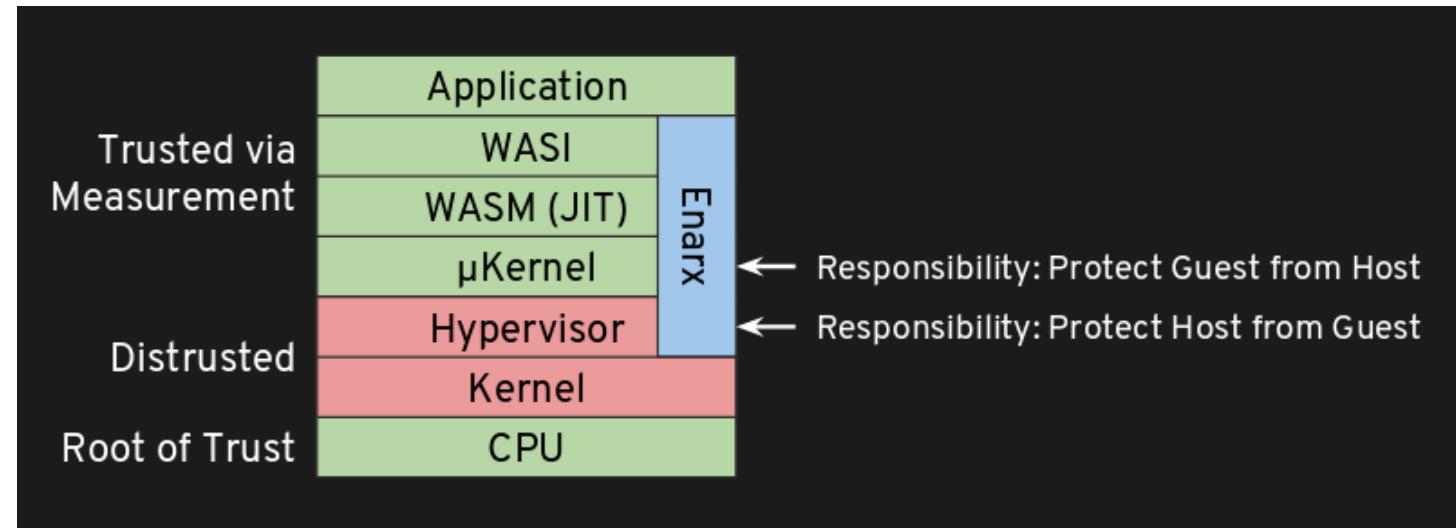
- Don't trust the host
- Don't trust the host owner
- Don't trust the host operator
- Hardware cryptographically verified
- Software audited and cryptographically verified





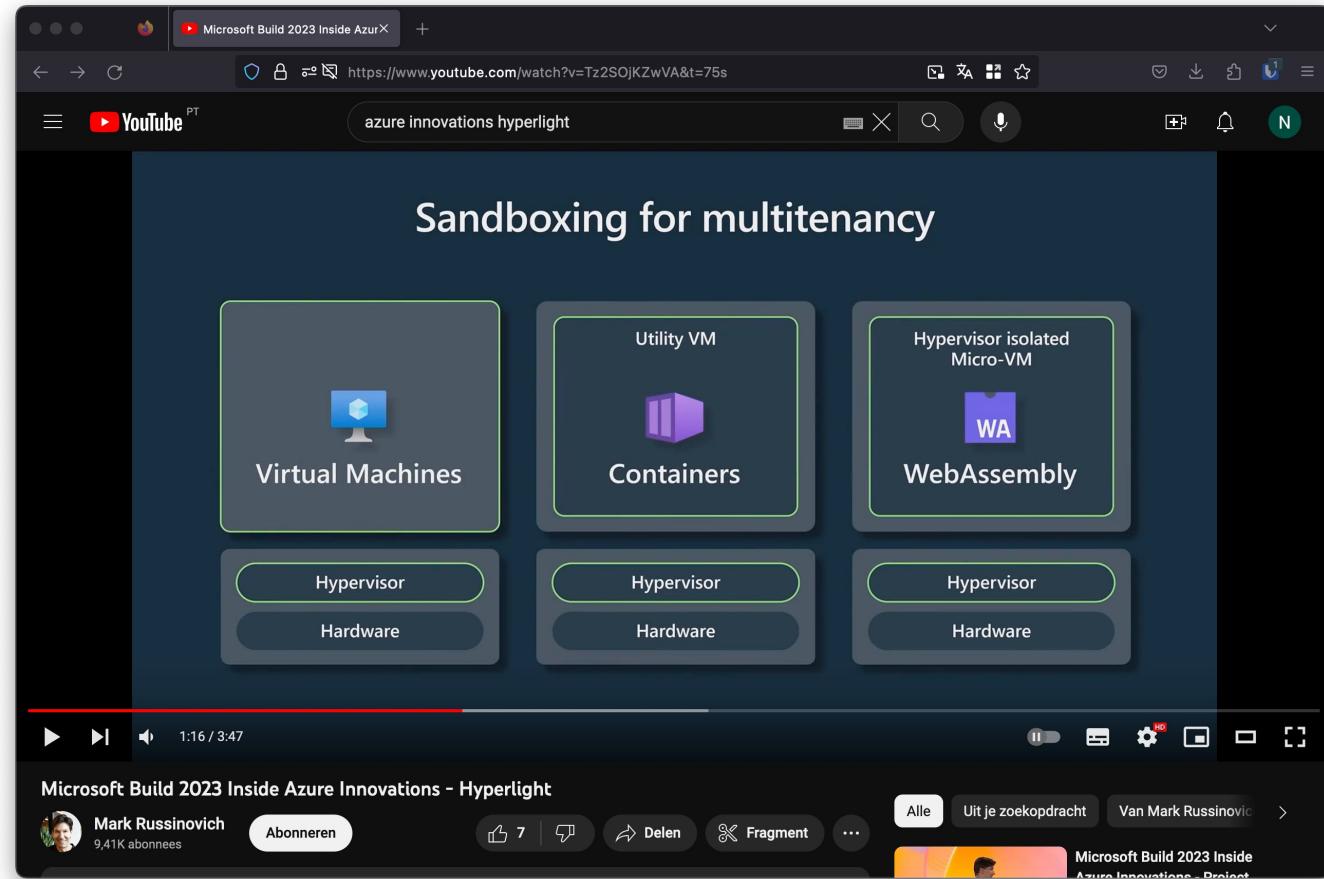
Enarx

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



0101
0101

Project Hyperlight



@nielstanis@infosec.exchange

DotNetIsolator & Project Hyperlight



DotNetIsolator: an experimental package for running .NET code in an isolated sandbox

Sandboxing Your Sandbox:
Leveraging Hypervisors for
WebAssembly Security

By: Danilo (Dan) Chiarlone

WASMCON
BETTER TOGETHER

Sandboxing Your Sandbox: Leveraging Hypervisors for WebAssembly Security - Danilo (Dan) Chiarlone

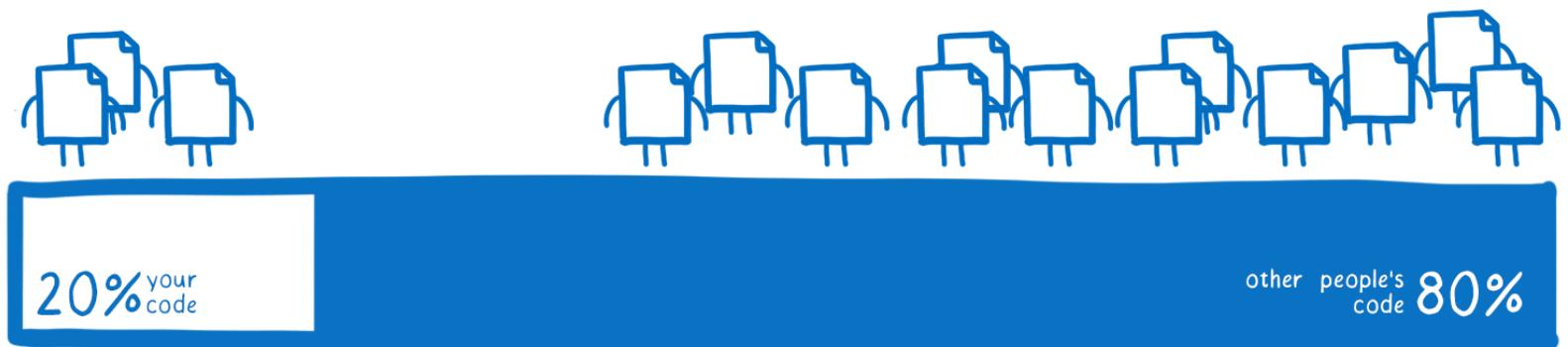


@nielstanis@infosec.exchange

0101
0101

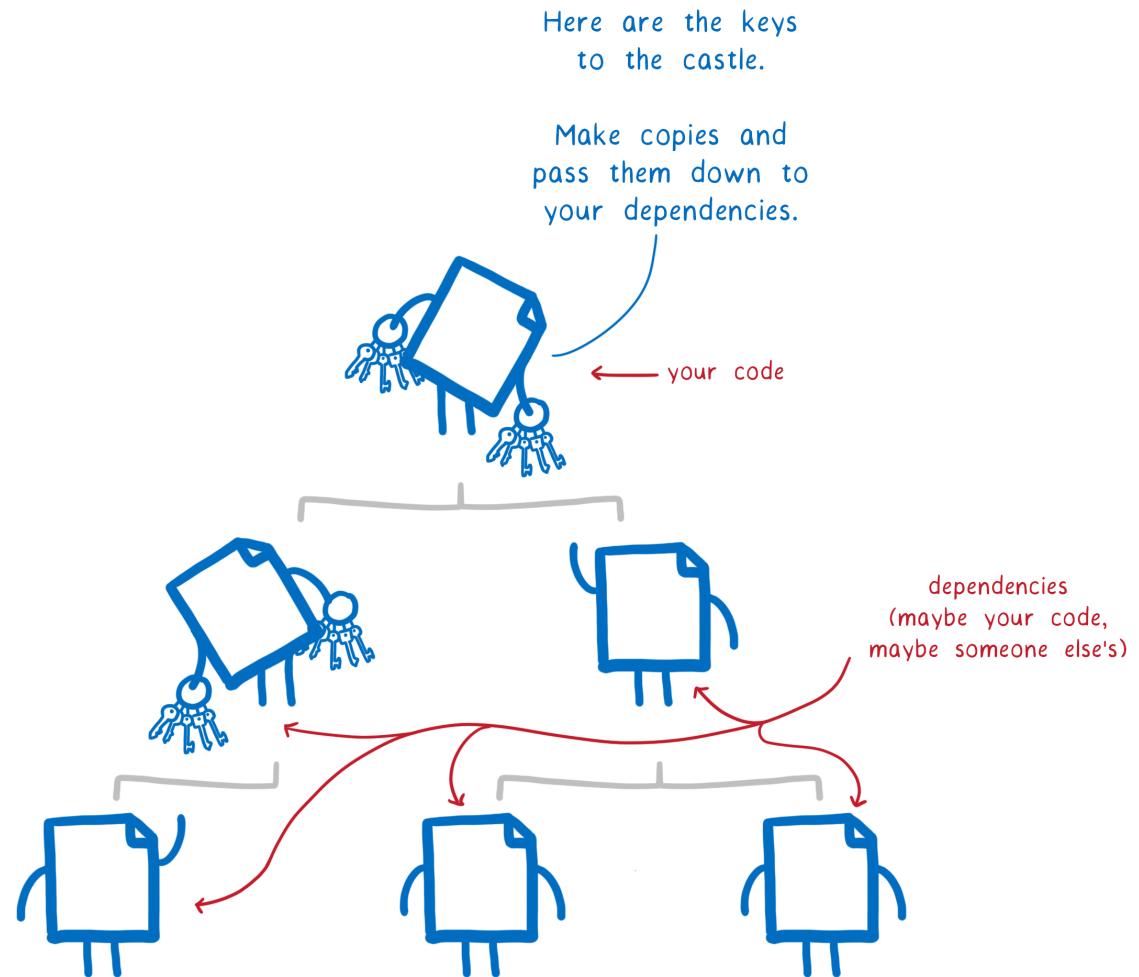
WASM - What's next?

composition of an
average code base



0101
0101

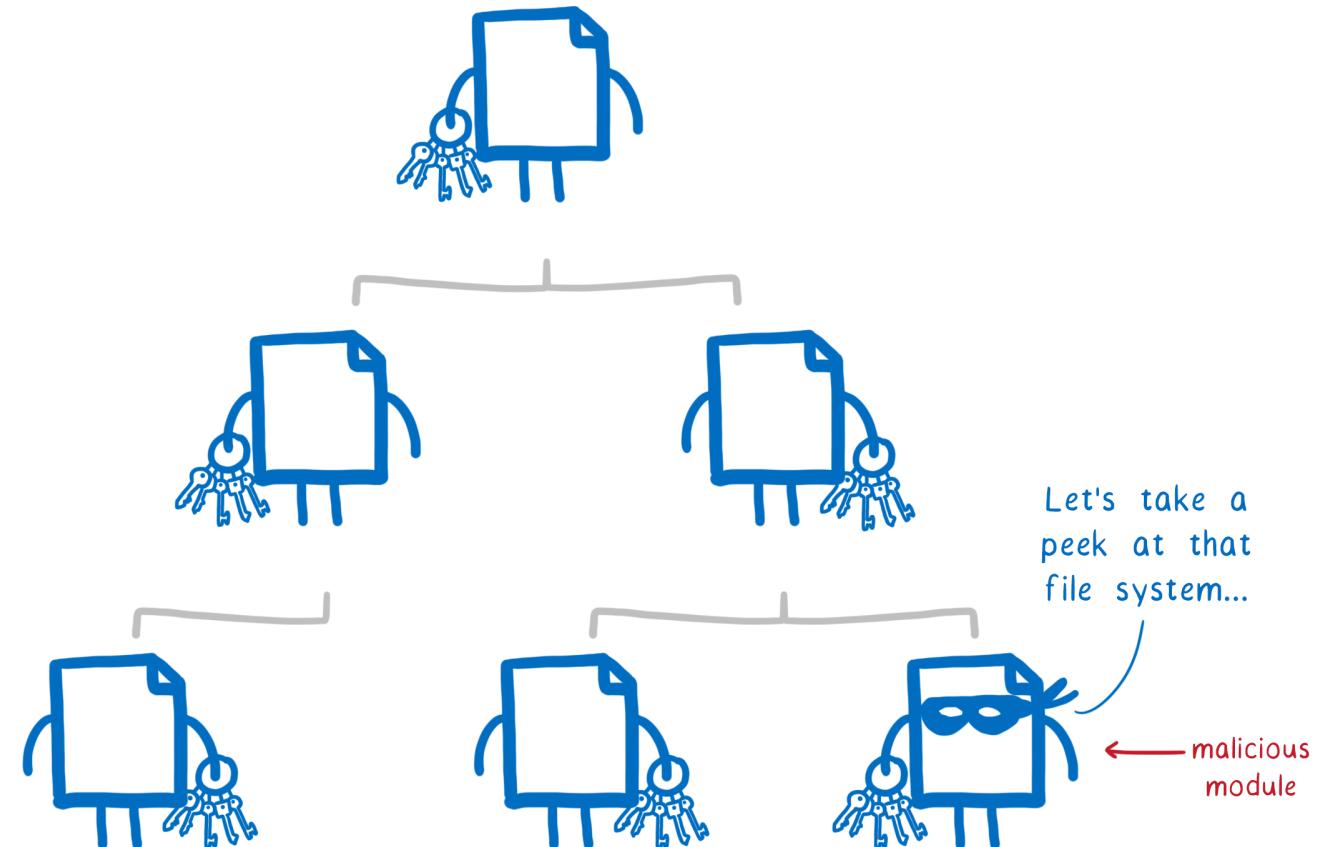
Dependencies



nis@infosec.exchange

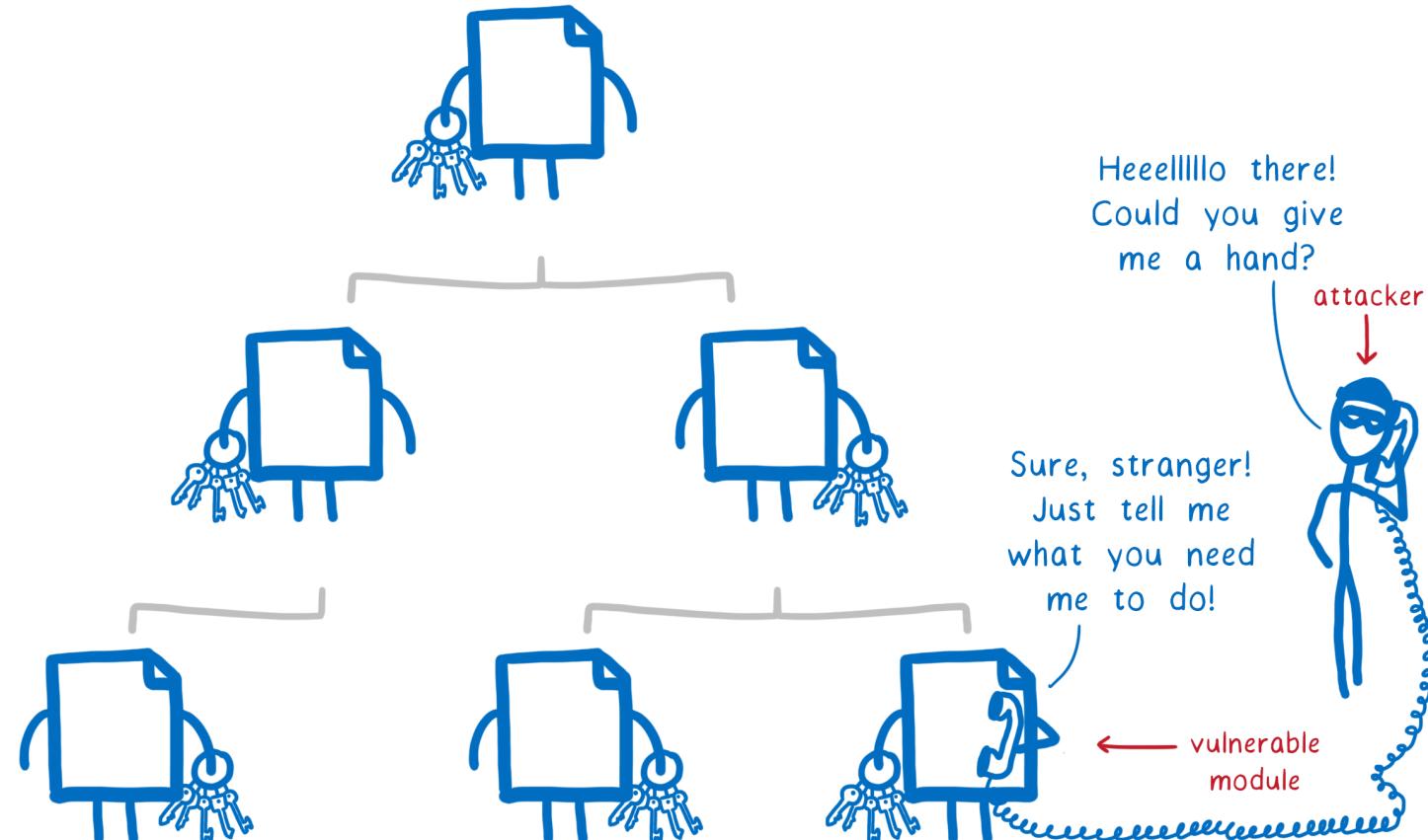
0101
0101

Malicious module



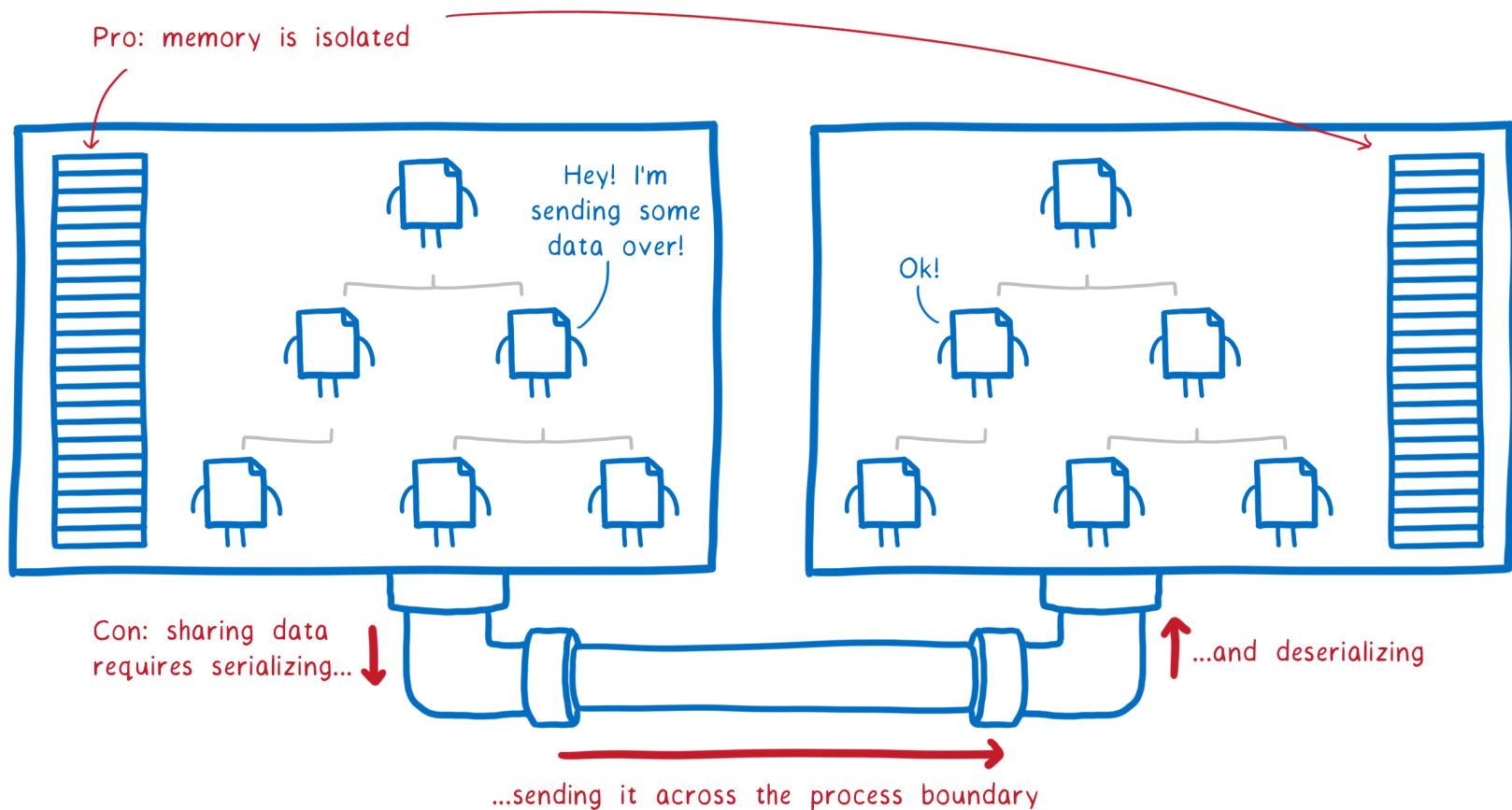
0101
0101

Vulnerable module



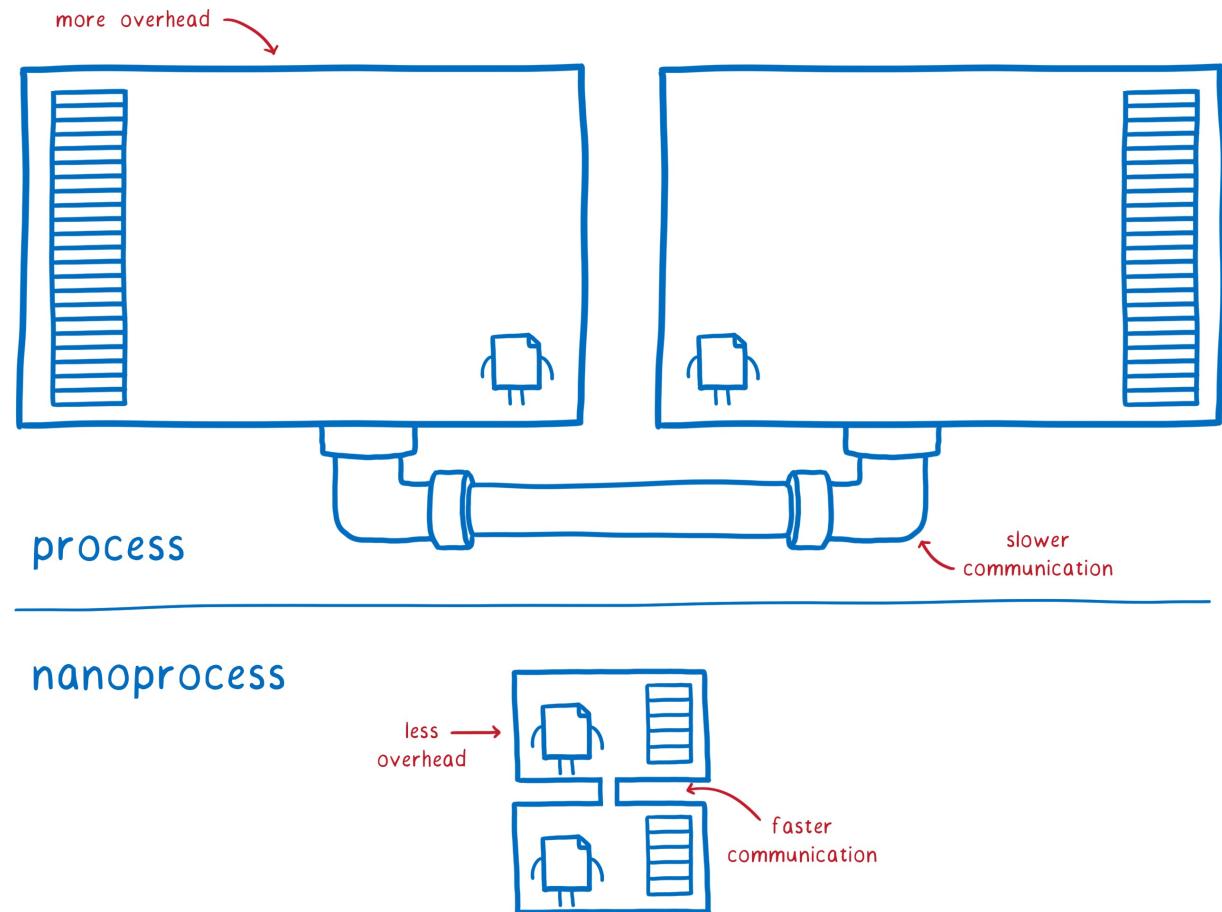
0101
0101

Process Isolation



0101
0101

WebAssembly Nano-Process



* not drawn to scale

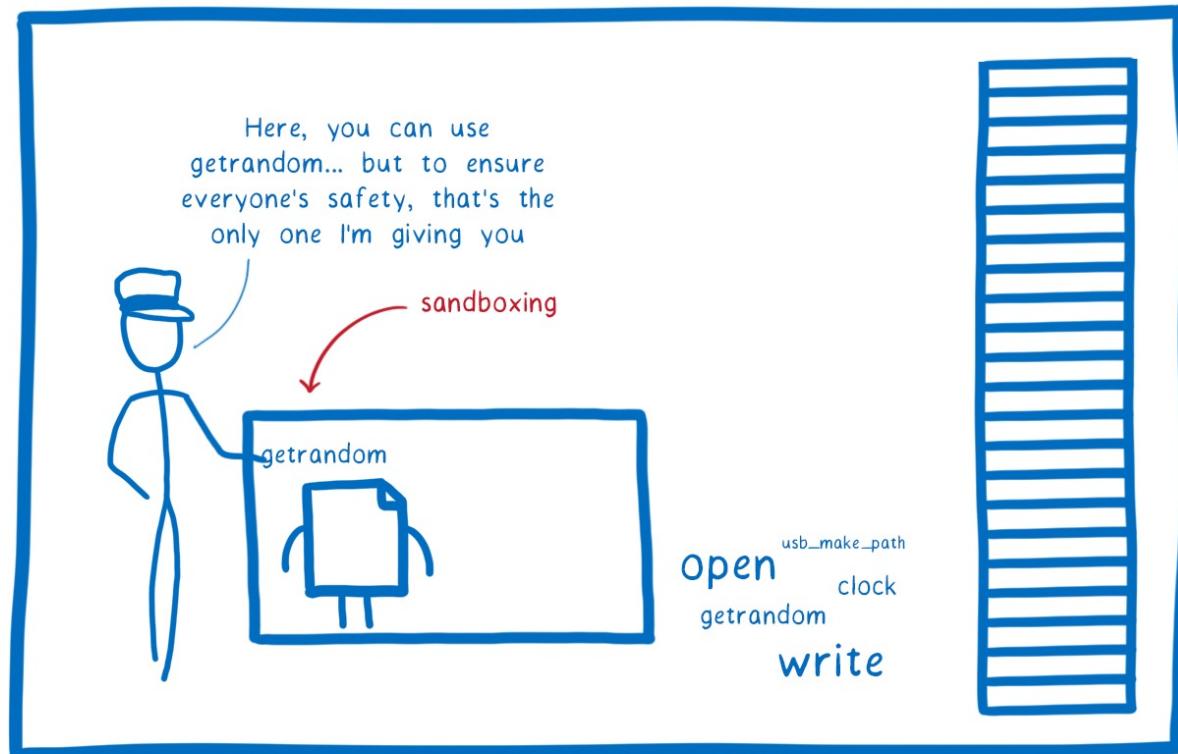


@nielstanis@infosec.exchange



WebAssembly Nano-Process

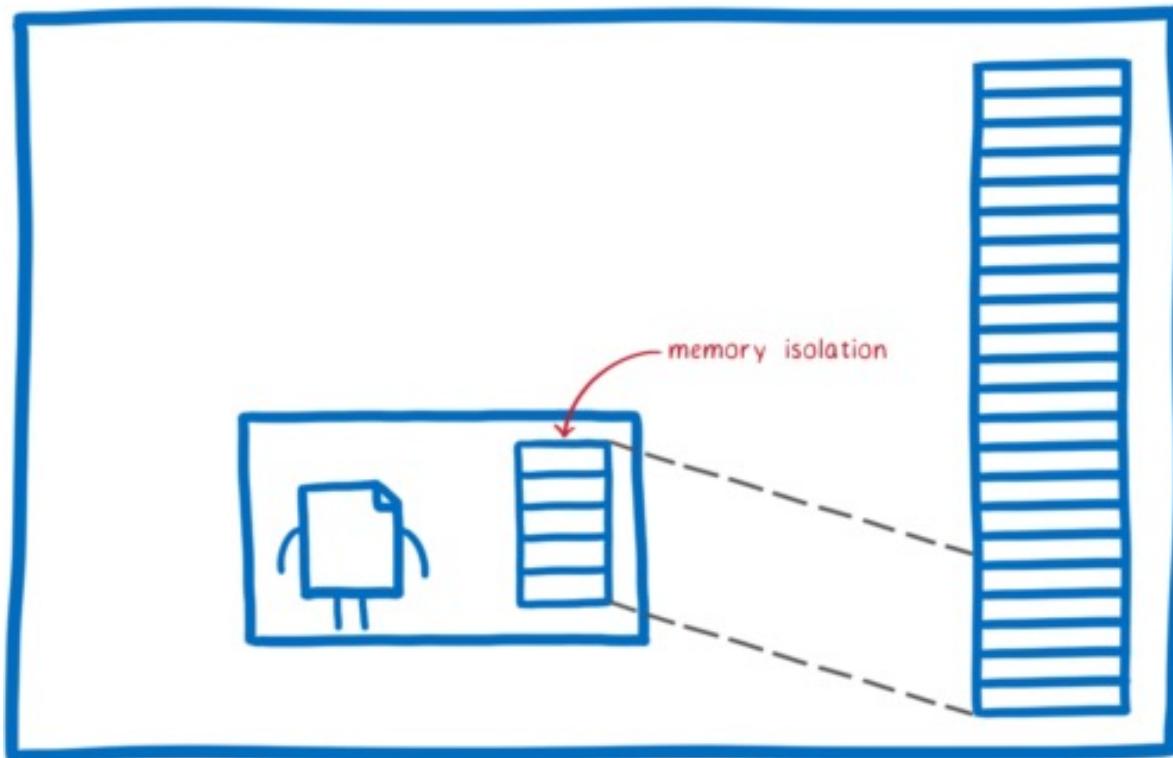
1. Sandboxing





WebAssembly Nano-Process

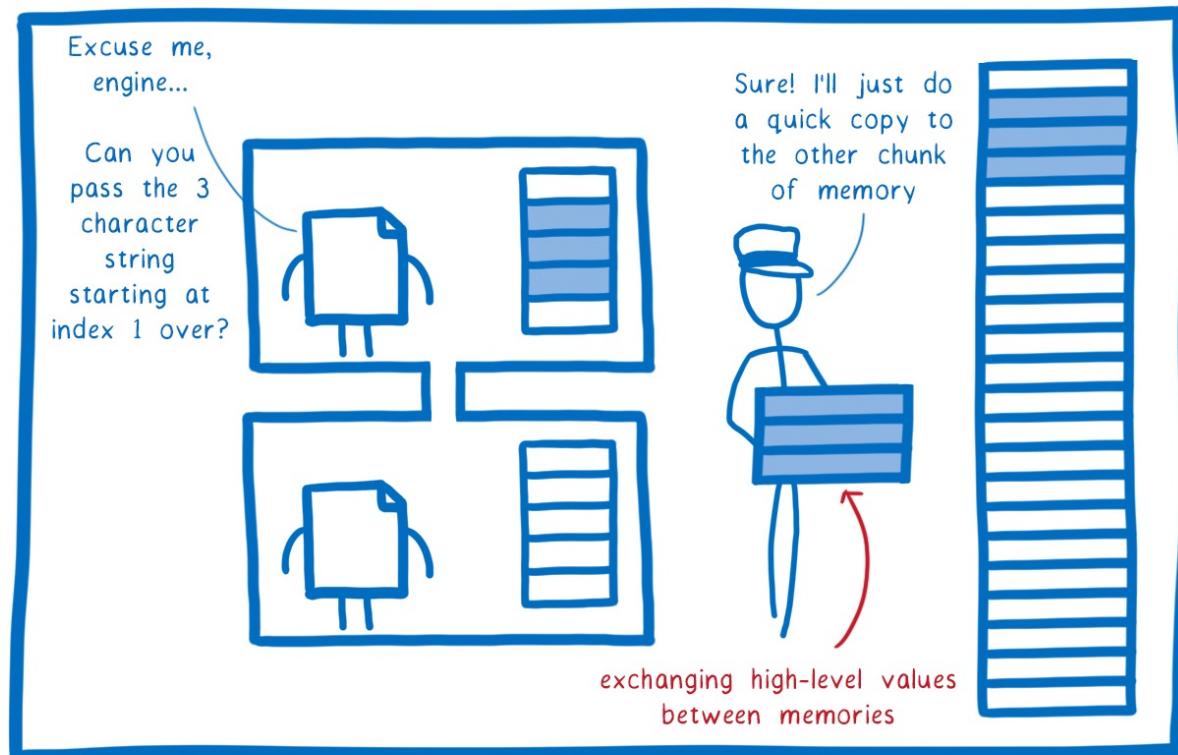
2. Memory model





WebAssembly Nano-Process

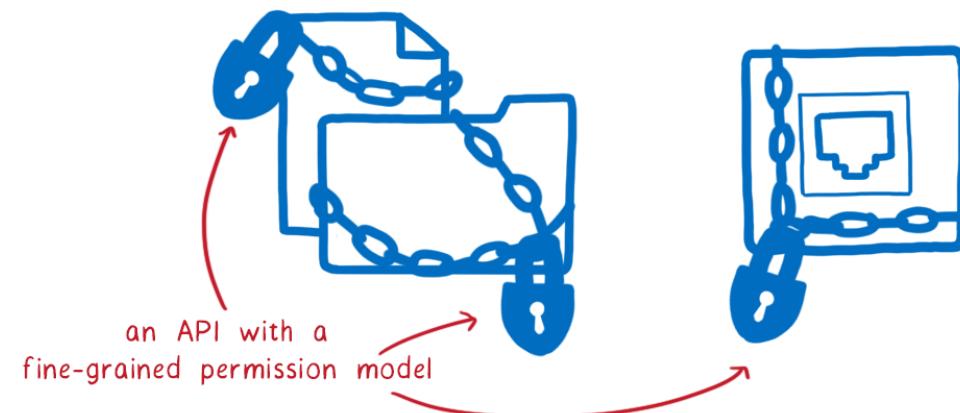
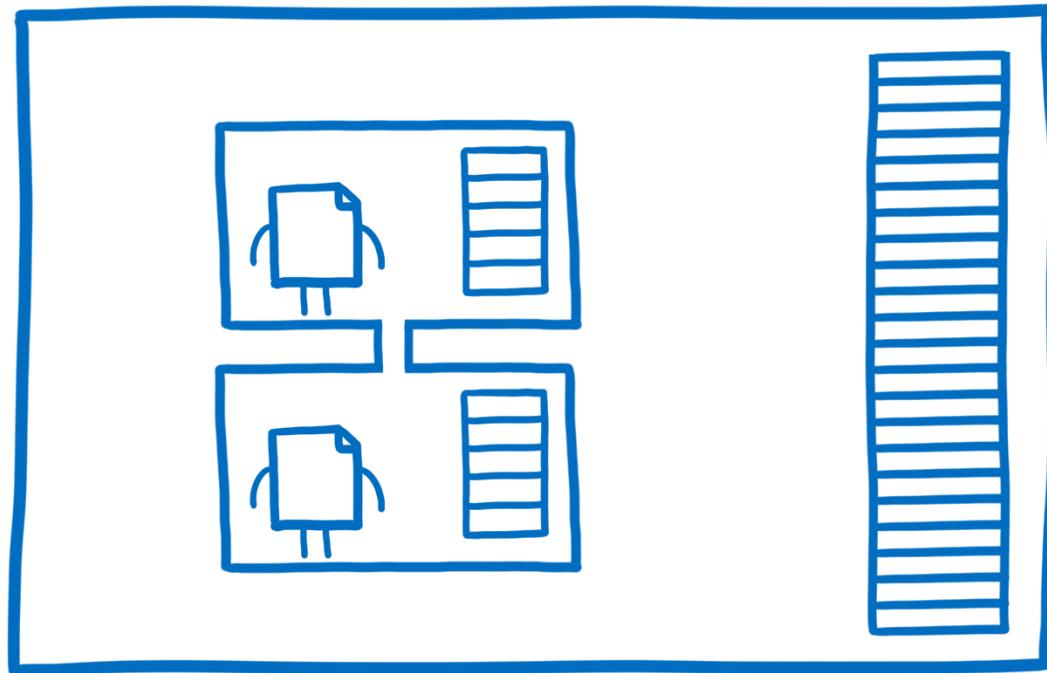
3. Interface Types





WebAssembly Nano-Process

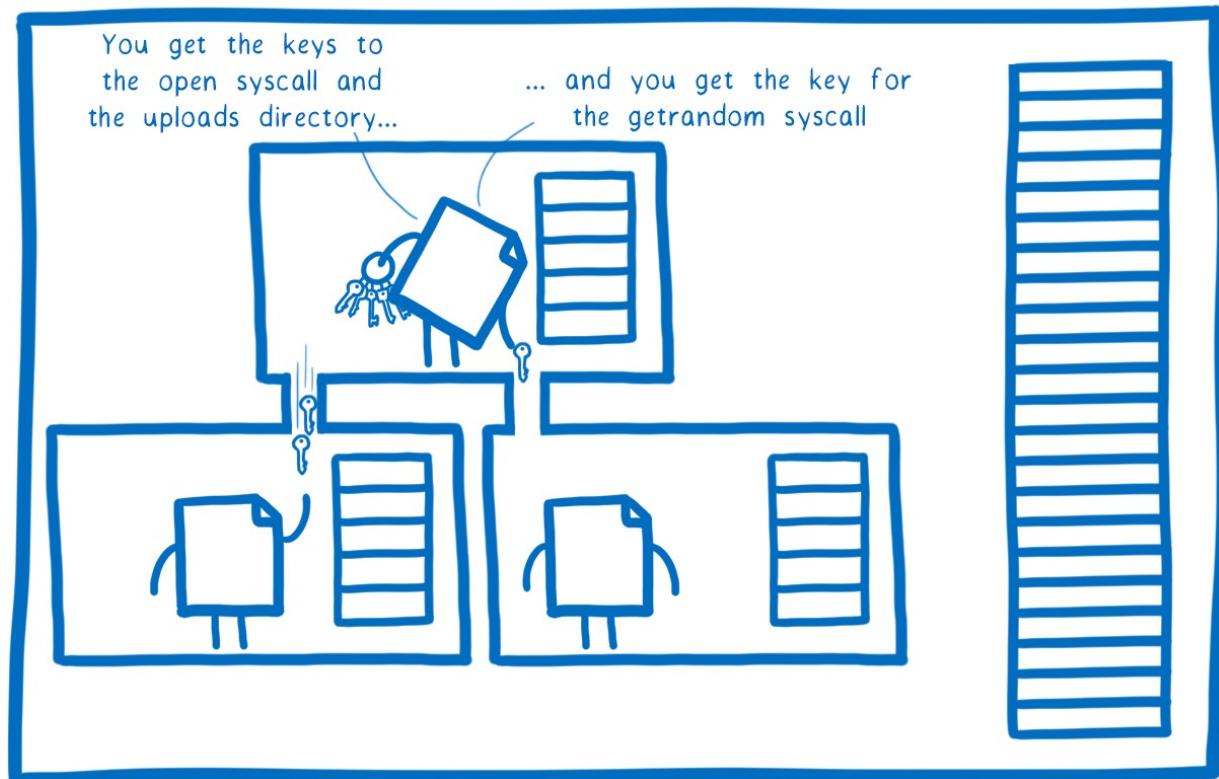
4. WebAssembly System Interface





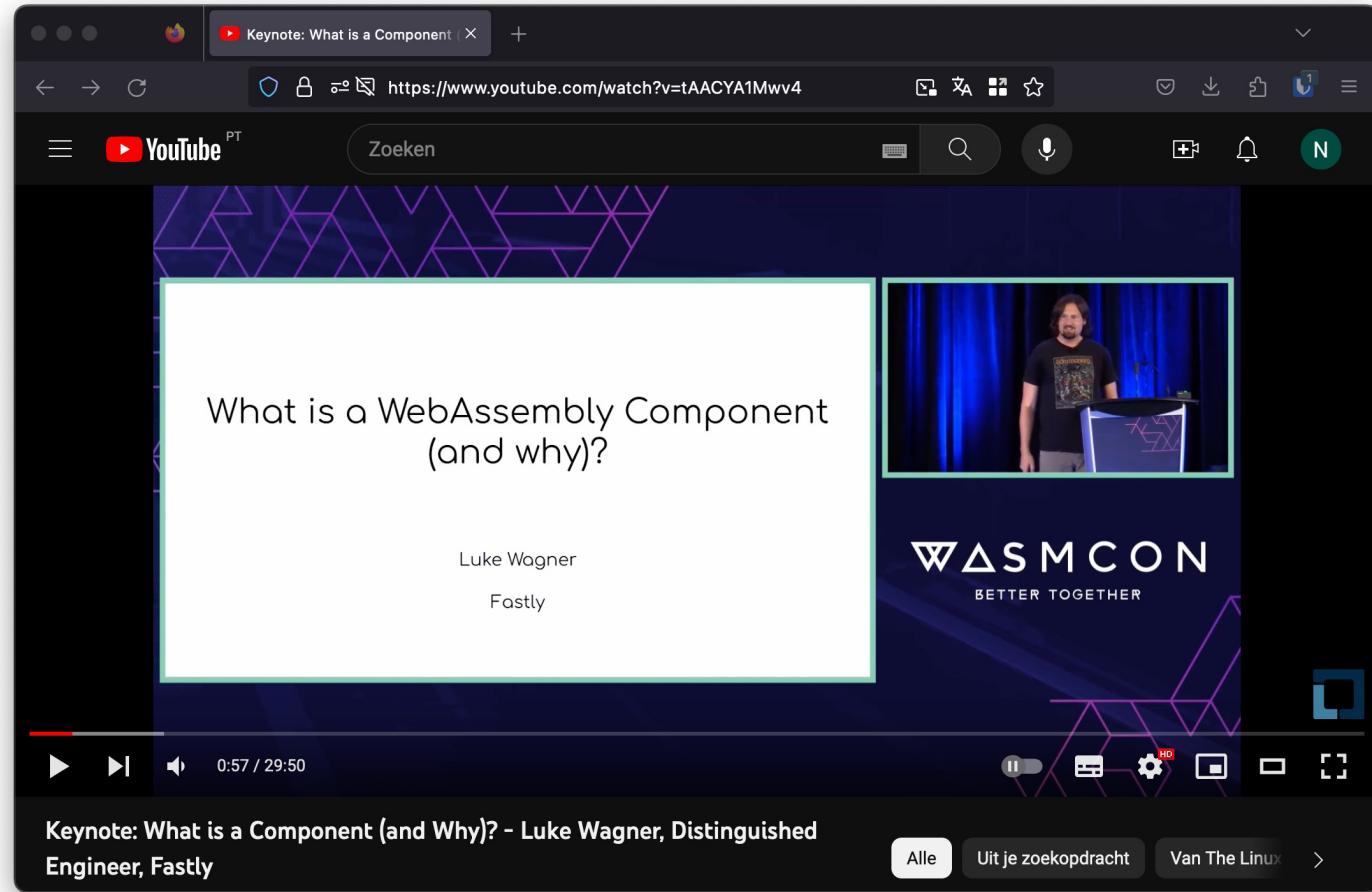
WebAssembly Nano-Process

5. The missing link



0101
0101

WebAssembly Component Model



0101
0101

WASI Preview 2

The screenshot shows a web browser window with the title bar "WASI Preview 2 Launched - sunfishcode". The URL in the address bar 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.

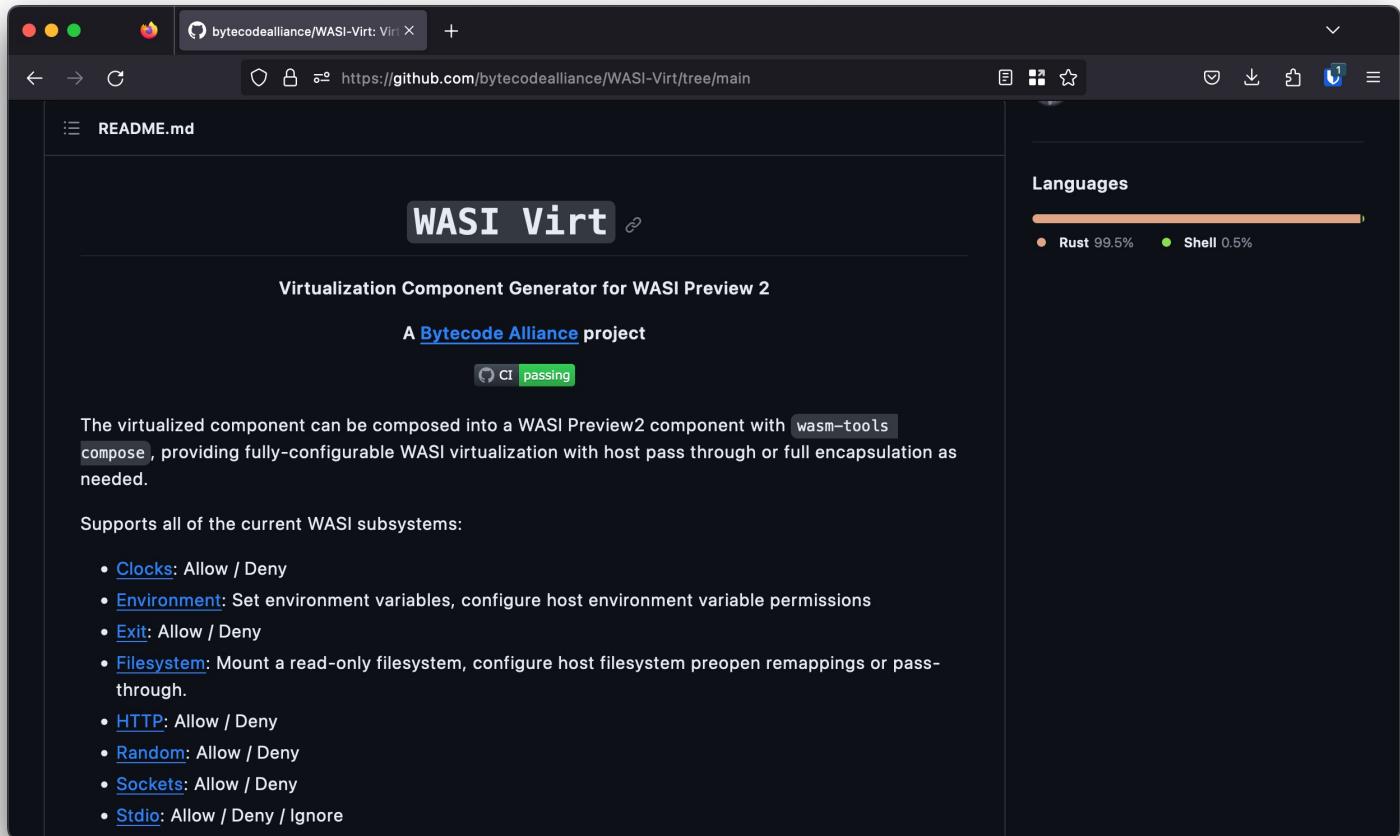
It also means that WASI is now officially based on the Wasm [component model](#), which makes it cross-language and virtualizable. Figuring out what a component model even is, designing it, implementing it, and building APIs using it has been a



@nielstanis@infosec.exchange

0101
0101

WASI Virt



 @nielstanis@infosec.exchange

0101
0101

WasmComponent.SDK

The screenshot shows a GitHub repository page for `SteveSandersonMS/wasm-component-sdk`. The main content is the `README` file, which contains the following text:

WasmComponent.Sdk

An experimental package to simplify building WASI preview 2 components using .NET, including early support for WIT files.

The build output is fully AOT compiled and is known to work in recent versions of wasmtime and WAMR.

Purpose

This is to simplify experimentation and prototyping.

Without this package, if you wanted to build a WASI preview 2 component with .NET, including using WIT imports/exports, there are about 5 different tools you'd need to discover, download, configure, and manually chain together. Just figuring out which versions of each are compatible with the others is a big challenge. Working out how to get started would be very painful.

With this package, you can add one NuGet reference and then get on with your experiments.

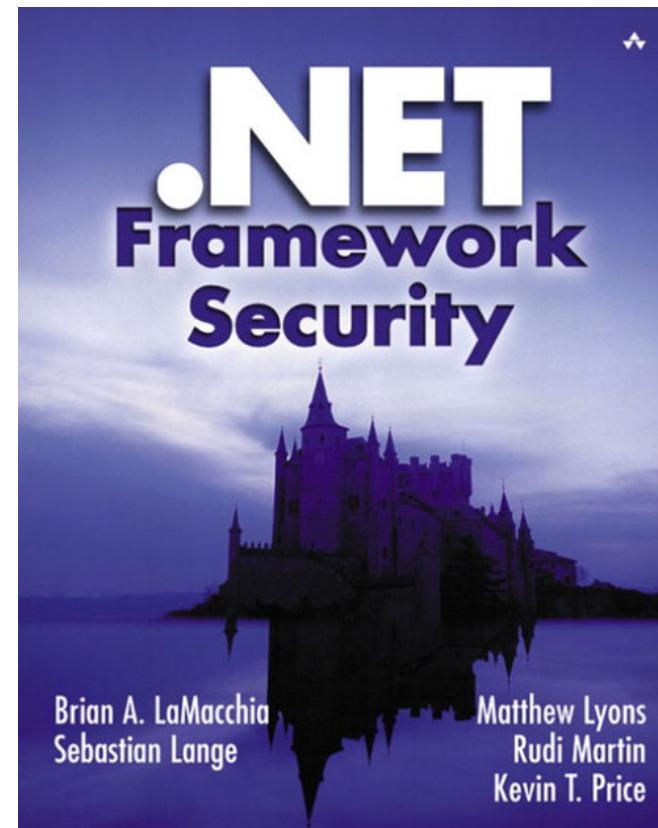
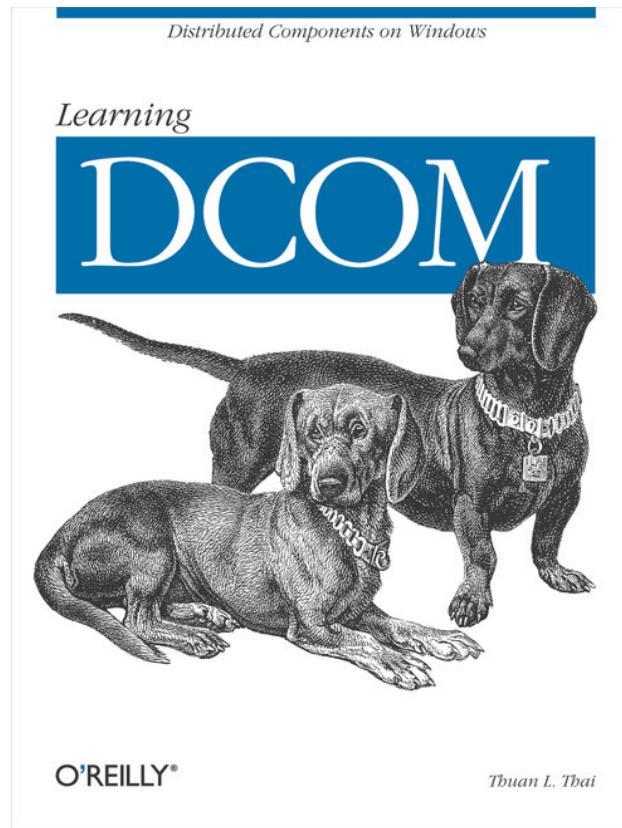
On the right side of the screen, there is a sidebar with a user profile for **NielsPilgaard** (Niels Pilgaard Grøndahl). Below the profile, there is a **Languages** section showing a progress bar where **C#** is at 100.0%.



@nielstanis@infosec.exchange

0101
0101

Have we seen this before?



 @nielstanis@infosec.exchange



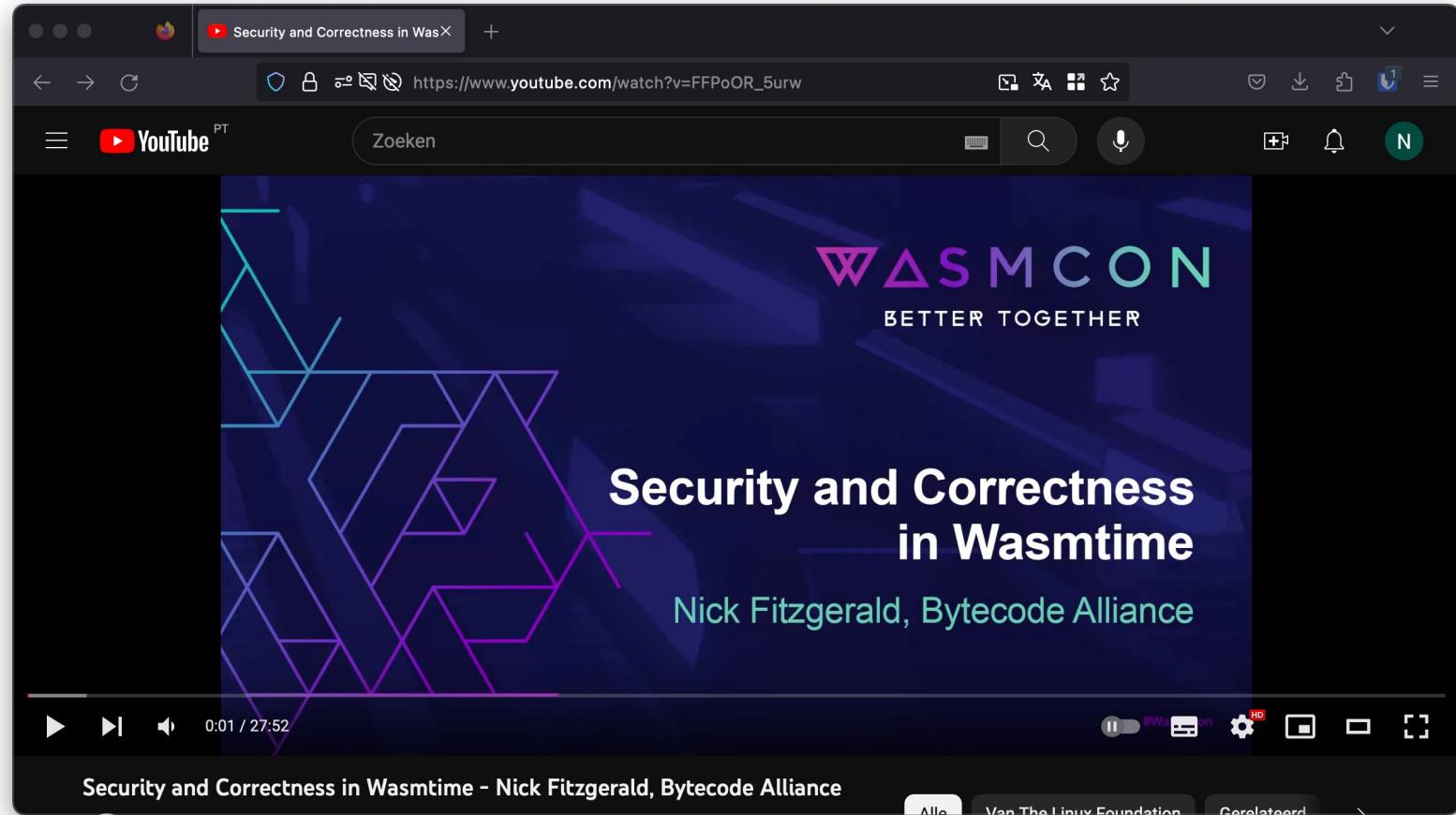
Runtimes and Security

- 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



0101
0101

Runtimes and Security



@nielstanis@infosec.exchange



Conclusion

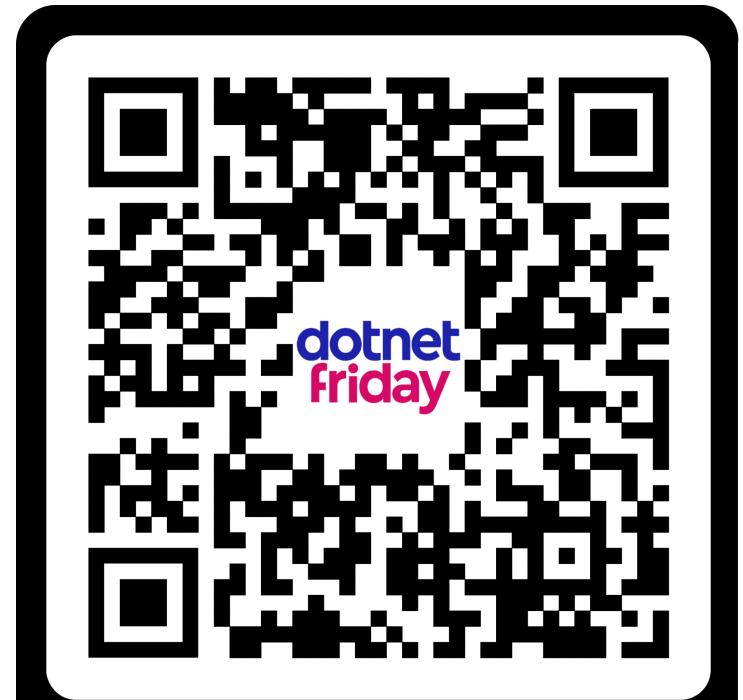
- Cloud Native ❤️ WebAssembly
- WebAssembly has a lot of potential to be used to run, extend, and secure your applications!
- Its as secure as the WebAssembly runtime implementation!
- WASI Preview 2 big milestone; now tooling can be implemented!



0101
0101

Questions?

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



Review My Session

