

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

Niels Tanis

NDC { Security }



0101
0101

Who am I?

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

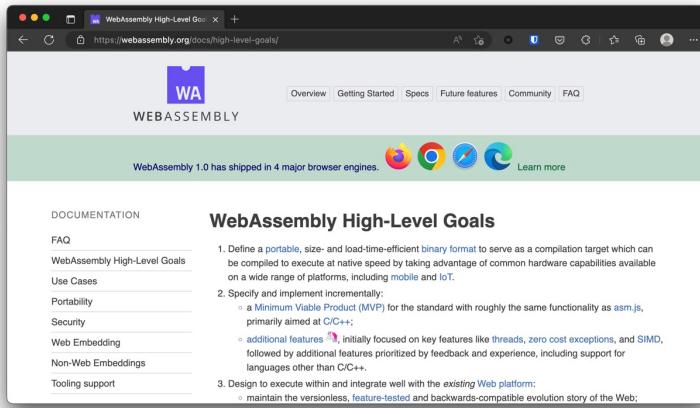


NDC { Security }

 @nielstanis@infosec.exchange

WebAssembly

0101
0101



The screenshot shows a web browser displaying the 'WebAssembly High-Level Goals' page from the official website. The page has a dark header with the 'WA' logo and 'WEBASSEMBLY' text. Below the header is a green banner stating 'WebAssembly 1.0 has shipped in 4 major browser engines.' followed by icons for Firefox, Chrome, Safari, and Edge, and a 'Learn more' link. The main content area is titled 'WebAssembly High-Level Goals' and lists three numbered goals:

1. Define a portable, size- and load-time-efficient binary format to serve as a compilation target which can be compiled to execute at native speed by taking advantage of common hardware capabilities available on a wide range of platforms, including mobile and IoT.
2. Specify and implement incrementally:
 - a Minimum Viable Product (MVP) for the standard with roughly the same functionality as `asm.js`, primarily aimed at C/C++;
 - additional features [↗], initially focused on key features like `threads`, `zero cost exceptions`, and `SIMD`, followed by additional features prioritized by feedback and experience, including support for languages other than C/C++.
3. Design to execute within and integrate well with the existing Web platform:
 - maintains the versionless, `feature-tested` and backwards-compatible evolution story of the Web;

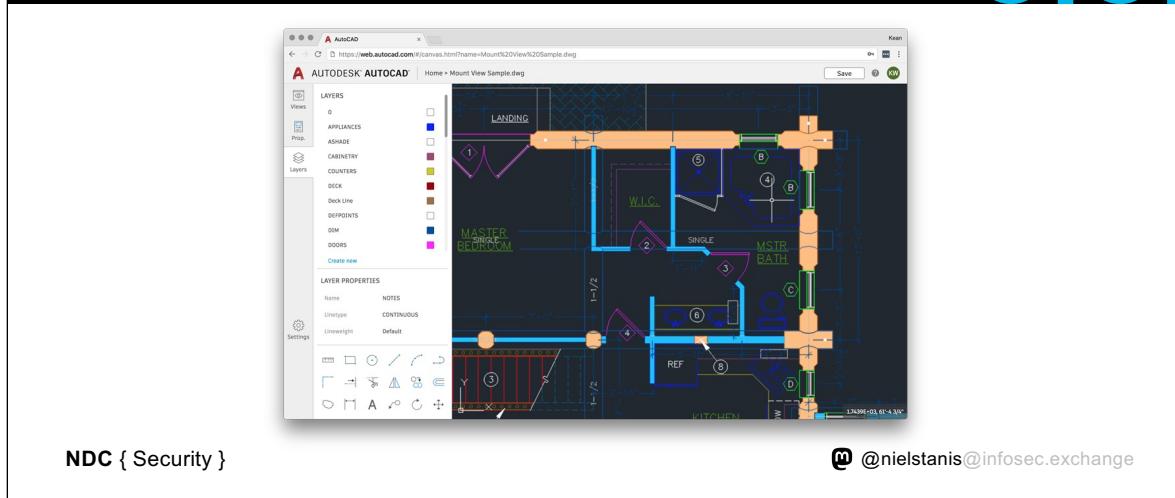
NDC { Security }

 @nielstanis@infosec.exchange

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

0101
0101

WebAssembly - AutoCAD



WebAssembly - SDK's

0101
0101

The screenshot shows a Medium article page. At the top, it says "Published in disney-streaming". Below that is a profile picture of Mike Hanley with the text "Mike Hanley Follow". It shows the date "Sep 8, 2021" and a "10 min read" duration. The title "Introducing the Disney+ Application Development Kit (ADK)" is prominently displayed. Below the title, it says "By: Tom Schroeder, Sr. SWE / Technical Lead, Native Client Platform, Living Room Devices". There are 415 likes and 4 comments. At the bottom, there are navigation links for "Previous Issues" and "Next Issues".

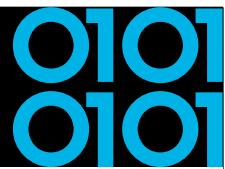
The screenshot shows a blog post from Amazon Science. The header includes the Amazon logo and the word "science". The category "CLOUD AND SYSTEMS" is listed. The title is "How Prime Video updates its app for more than 8,000 device types". Below the title, it says "The switch to WebAssembly increases stability, speed." and "At Prime Video, we're delivering content to millions of customers on". The author is listed as "By Alexandru Ene" and the date is "January 27, 2022". There is a "Share" button.

NDC { Security }

@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>



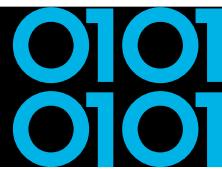
Agenda

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

NDC { Security }

 @nielstanis@infosec.exchange

WebAssembly Design



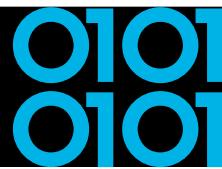
- **Be fast, efficient, and portable**
 - Executed in near-native speed across different platforms
- **Be readable and debuggable**
 - In low-level bytecode but also human readable
- **Keep secure**
 - Run on sandboxed execution environment
- **Don't break the web**
 - Ensure backwards compatibility



NDC { Security }

 @nielstanis@infosec.exchange

WebAssembly



- Binary instruction format for stack-based virtual machine similar to .NET running MSIL
- Designed as a portable compilation target
- The security model of WebAssembly:
 - Protect users from buggy or malicious modules
 - Provide developers with useful primitives and mitigations for developing safe applications

NDC { Security }

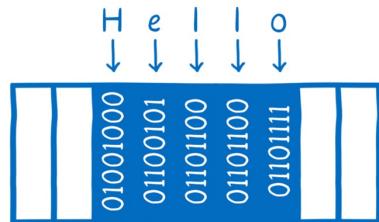
 @nielstanis@infosec.exchange

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

0101
0101

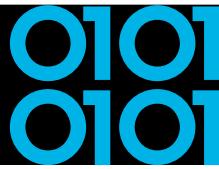
WebAssembly Memory

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



NDC { Security }

 @nielstanis@infosec.exchange

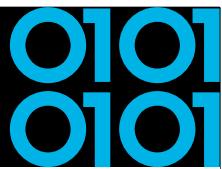


WebAssembly Control-Flow Integrity

```
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!");
```

NDC { Security }

 @nielstanis@infosec.exchange



WebAssembly Control-Flow Integrity

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

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

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

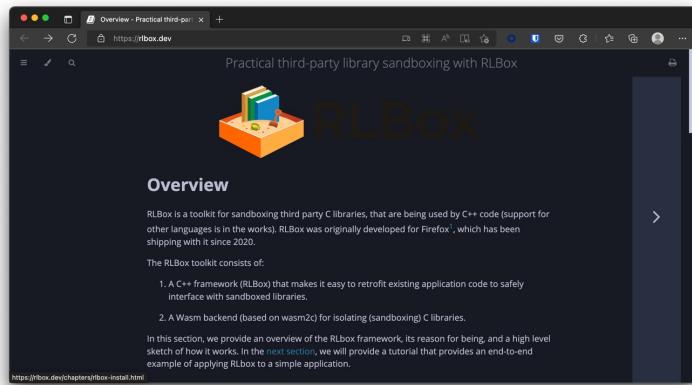
```
Console.WriteLine("Done!");
```

NDC { Security }

 @nielstanis@infosec.exchange

FireFox RLBox

0101
0101

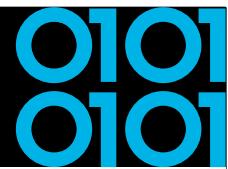


NDC { Security }

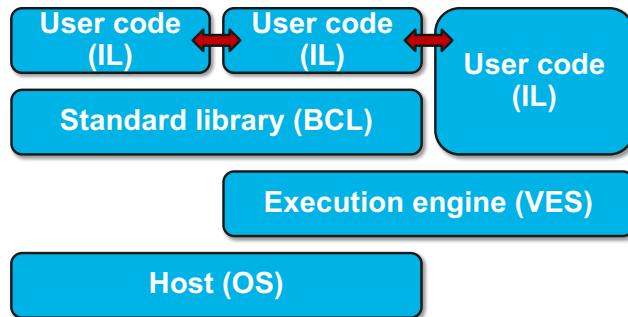
@nielstanis@infosec.exchange

<https://rlbox.dev/>

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



Running .NET on WebAssembly

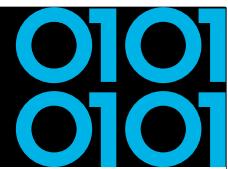


NDC { Security }

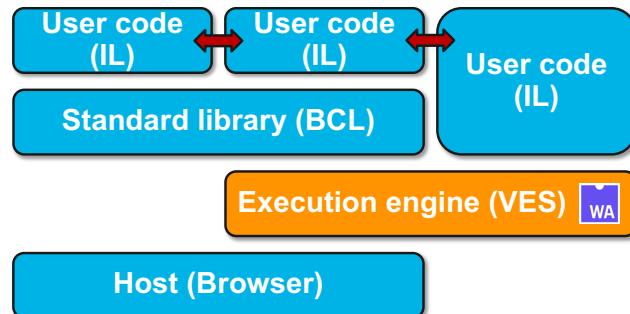
 @nielstanis@infosec.exchange

Diagram:

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



Running .NET on WebAssembly



NDC { Security }

@nielstanis@infosec.exchange

Diagram:

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

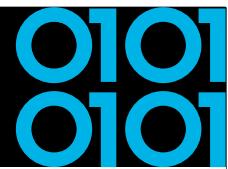
0101
0101

Blazor WebAssembly

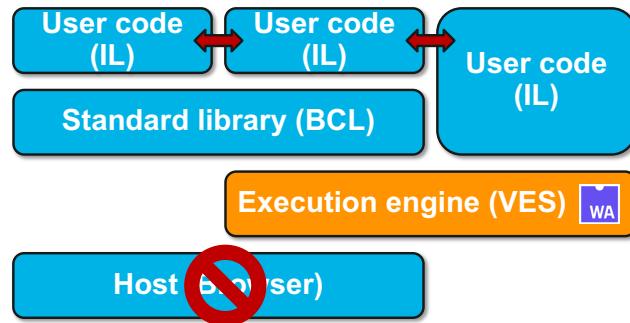


NDC { Security }

@nielstanis@infosec.exchange



Running .NET on WebAssembly

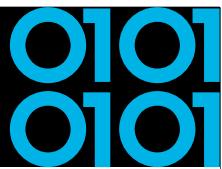


NDC { Security }

@nielstanis@infosec.exchange

Diagram:

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

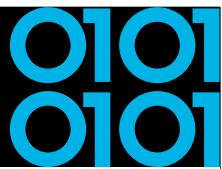


WebAssembly System Interface WASI

- Introduced in March 2019 by Bytecode Alliance
- WasmTime implementation as reference
- POSIX inspired, engine-independent, non-Web system-oriented API for WebAssembly

NDC { Security }

 @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? 😊

NDC { Security }

@nielstanis@infosec.exchange

0101
0101

Docker vs WASM & WASI

The screenshot shows a Twitter interface on a dark-themed browser window. On the left is a sidebar with icons for Home, Search, Notifications, Direct Messages, and Profile. The main area displays a tweet from Solomon Hykes (@solomonstrel) with the text: "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 this is a reply from Lin Clark (@linclark) with the text: "WebAssembly running outside the web has a huge future. And that future gets one giant leap closer today with...". A link to "hacks.mozilla.org/2019/03/standa..." is also visible. At the bottom of the tweet card, it says "D deze collectie weergeven". The timestamp "9:39 p.m. - 27 mrt. 2019 · Twitter Web Client" is at the bottom right of the card.

NDC { Security }

@nielstanis@infosec.exchange

0101
0101

Docker vs WASM & WASI

A screenshot of a Twitter post from Solomon Hykes (@solomonstre). 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 main tweet, there is a reply from Solomon Hykes: "If WASM+WASI existed in 2008, we wouldn't have needed to create Docker. That's how important it is. Webassembly on the server is the future of computing. A standardized system interface was the missing link. Let's hope WASI is up to the task! twitter.com/lindlark/status/110333100000000000". The post has 56 Retweets, 5 Geciteerde Tweets, and 165 Vind-ik-leuks.

NDC { Security }

@nielstanis@infosec.exchange

0101
0101

Docker & WASM

The screenshot displays a presentation slide with the following content:

Introducing the Docker+Wasm Technical Preview

MICHAEL IRWIN
Oct 24 2022

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.

Docker Engine Architecture Diagram:

```
graph TD; DE[Docker Engine] --> C[container]; C --> CH1[containerd-shim]; C --> CH2[containerd-shim]; C --> CH3[containerd-wasm-shim]; CH1 --> R1[runc]; CH2 --> R2[runc]; CH3 --> WM[wasmEdge]; R1 --> CP1[Container process]; R2 --> CP2[Container process]; WM --> WMModule[Wasm Module]
```

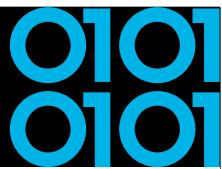
Let's look at an example!

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

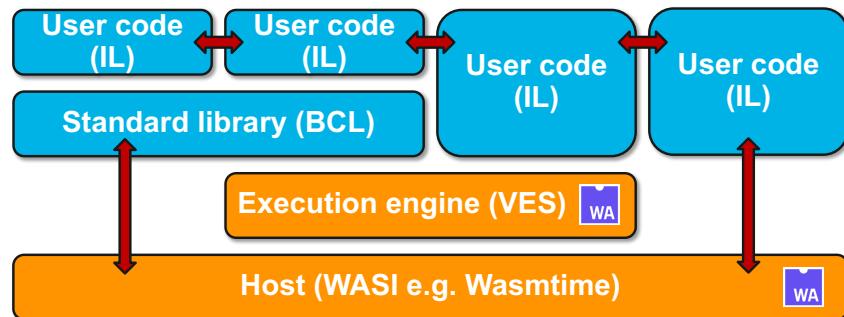
```
docker run -dp 8888:8888 --name=wasm-example --runtime=io.containerd.wasmedge.v1 --platform=wasi/wasi32 michaelirwin244/wasm-example
```

NDC { Security }

@nielstanis@infosec.exchange

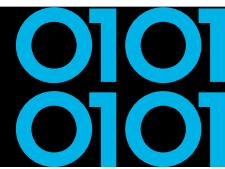


WebAssembly System Interface WASI

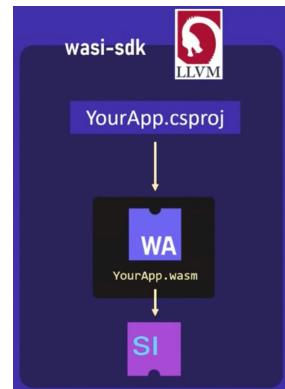


NDC { Security }

@nielstanis@infosec.exchange



Experimental WASI SDK for .NET

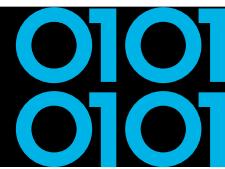


NDC { Security }

 @nielstanis@infosec.exchange

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

Experimental WASI SDK for .NET



The screenshot shows a GitHub issue page for 'dotnet/runtime' issue #65895. The issue is titled 'WASI support tracking'. It is marked as 'Open' and has 3 of 12 tasks assigned. The description notes that the issue tracks known issues in early WASI-enabled runtime builds. A list of pull requests is provided for review, including:

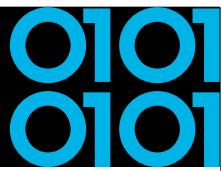
- [wasi][RID] Add new RID for WASI #77780
- [API Proposal] ReportTagSystem.Takeoff() #78389 API change
- [mono][wasi] Add a CI lane for a new wasi RID #72641

NDC { Security }

@nielstanis@infosec.exchange

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

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



Extending .NET with WASM

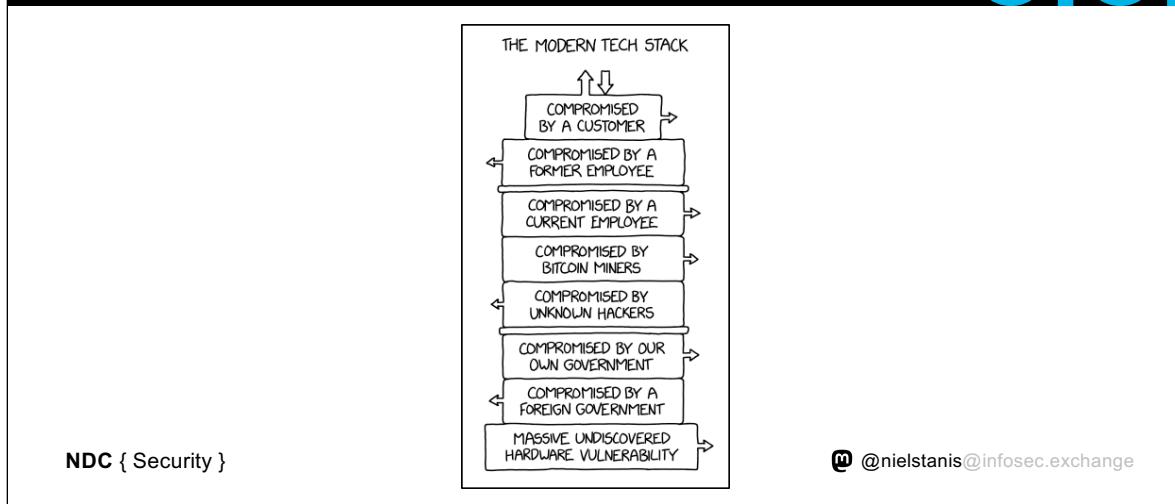
- WasmTime.NET NuGet package
- Can run WASM inside of any .NET application
- Extend with Rust based WASM module
- Demo time!

NDC { Security }

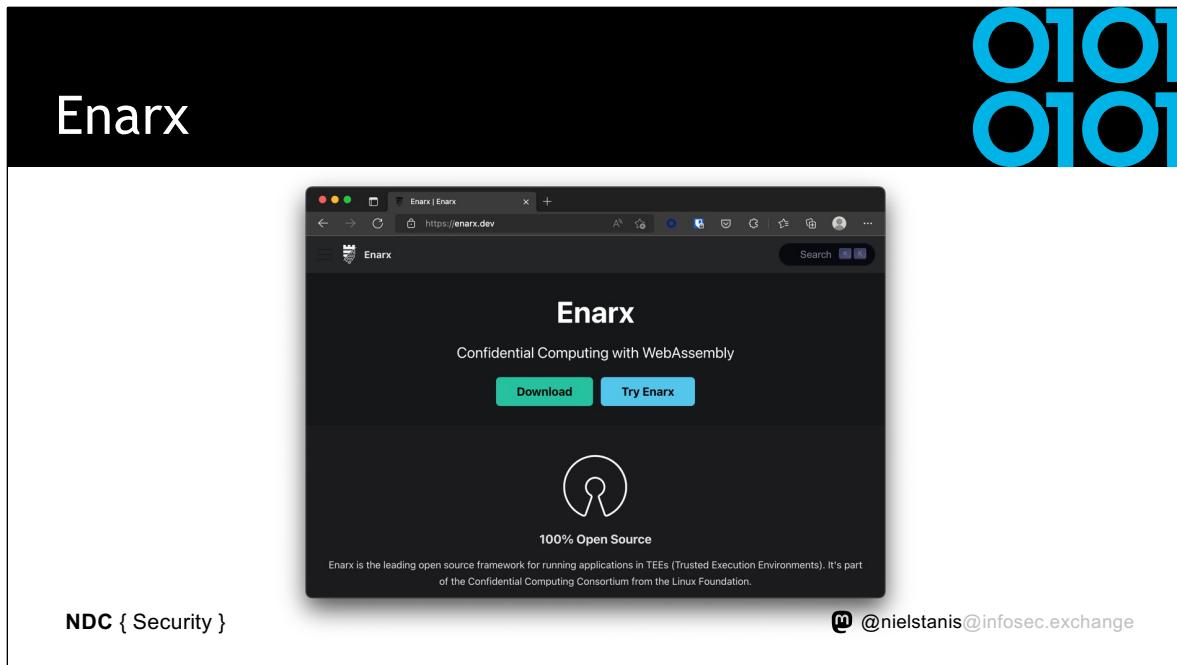
 @nielstanis@infosec.exchange

0101
0101

Trusted Computing - XKCD 2166



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



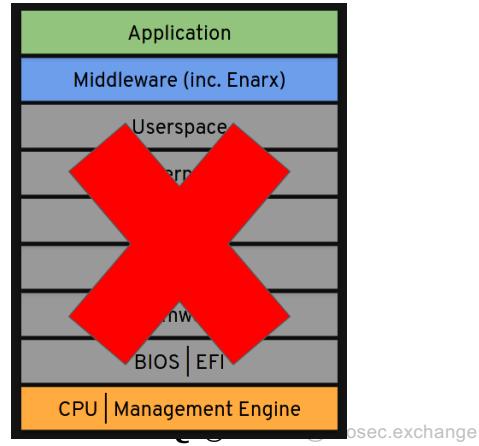
<https://enarx.dev/>

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

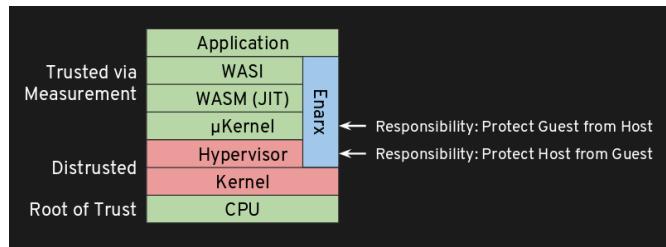
NDC { Security }



Enarx

0101
0101

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



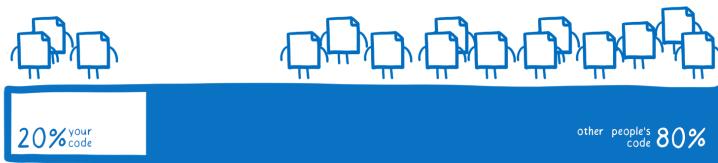
NDC { Security }

@nielstanis@infosec.exchange

0101
0101

WASM - What's next?

composition of an
average code base



NDC { Security }

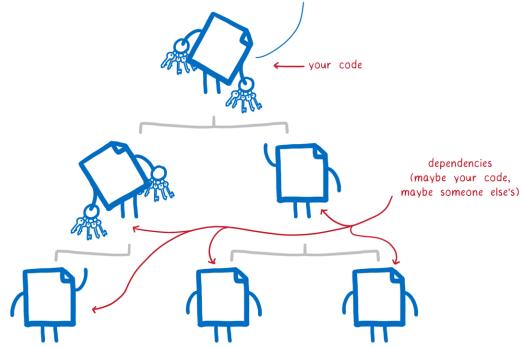
 @nielstanis@infosec.exchange

0101
0101

Dependencies

Here are the keys
to the castle.

Make copies and
pass them down to
your dependencies.

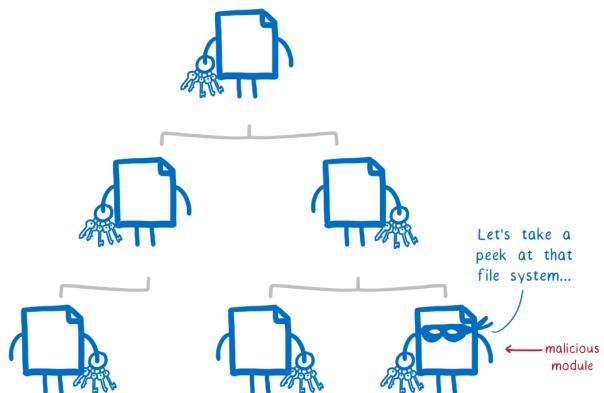


NDC { Security }

iis@infosec.exchange

0101
0101

Malicious module

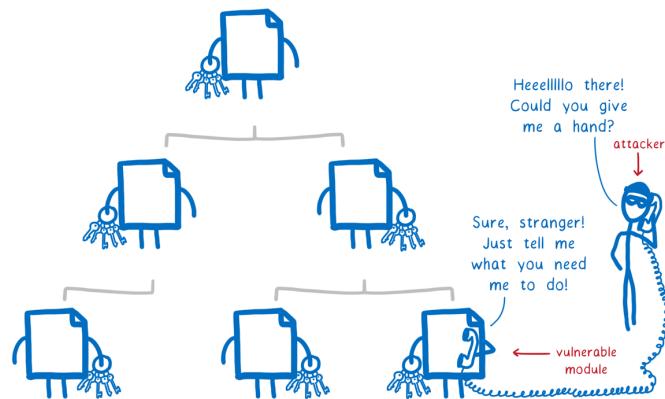


NDC { Security }

 @nielstanis@infosec.exchange

0101
0101

Vulnerable module

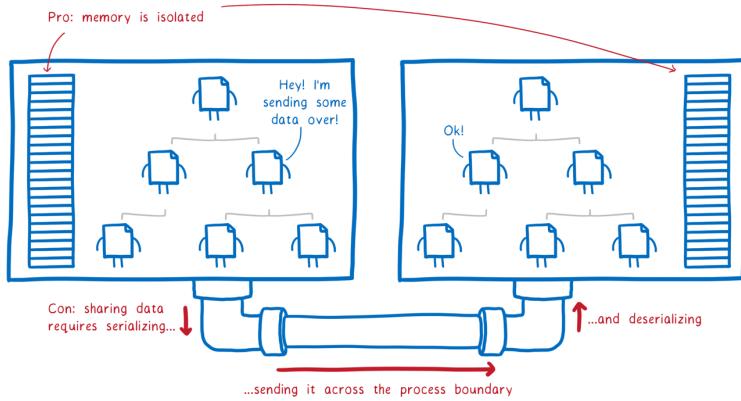


NDC { Security }

 @nielstanis@infosec.exchange

0101
0101

Process Isolation

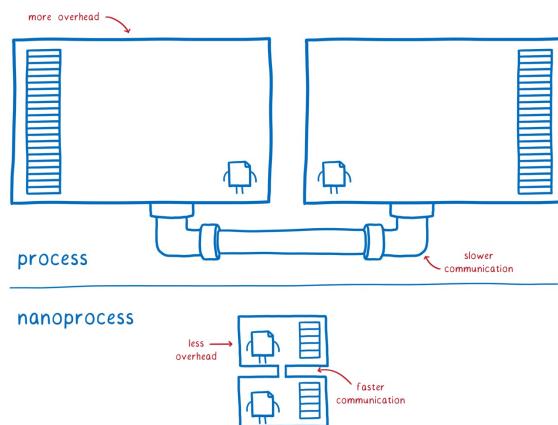


NDC { Security }

 @nielstanis@infosec.exchange

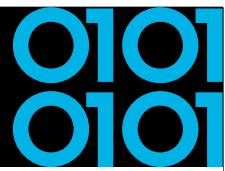
0101
0101

WebAssembly Nano-Process



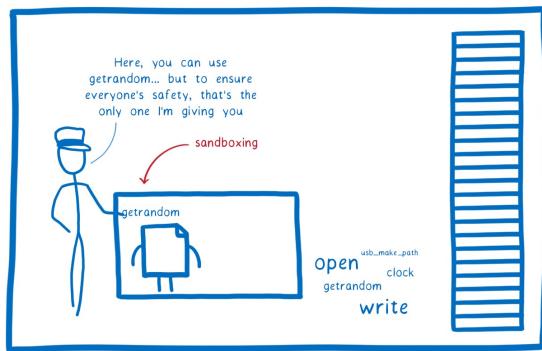
NDC { Security }

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



WebAssembly Nano-Process

1. Sandboxing



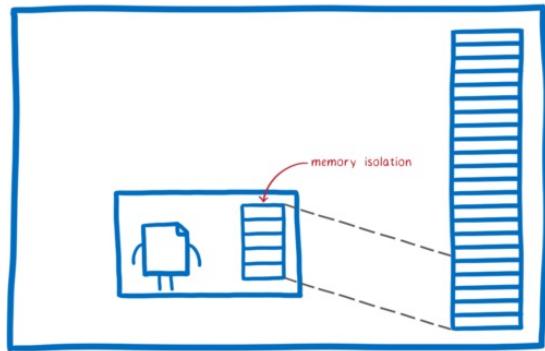
NDC { Security }

 @nielstanis@infosec.exchange

0101
0101

WebAssembly Nano-Process

2. Memory model



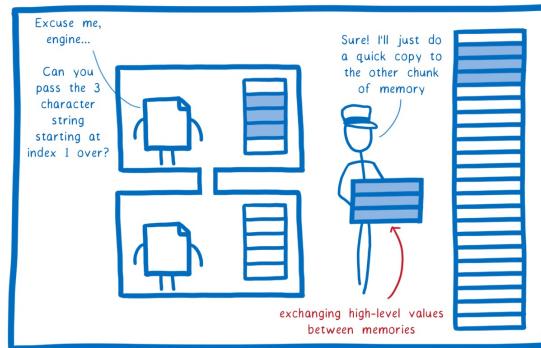
NDC { Security }

 @nielstanis@infosec.exchange

0101
0101

WebAssembly Nano-Process

3. Interface Types



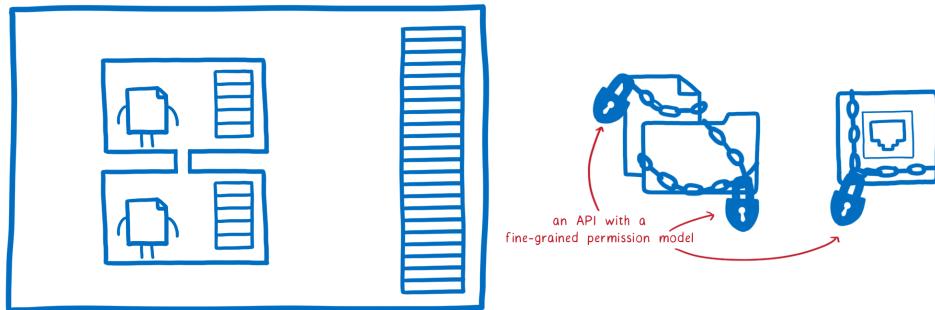
NDC { Security }

 @nielstanis@infosec.exchange

0101
0101

WebAssembly Nano-Process

4. WebAssembly System Interface



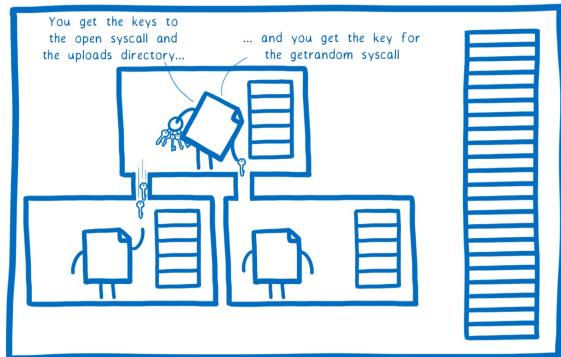
NDC { Security }

 @nielstanis@infosec.exchange

0101
0101

WebAssembly Nano-Process

5. The missing link

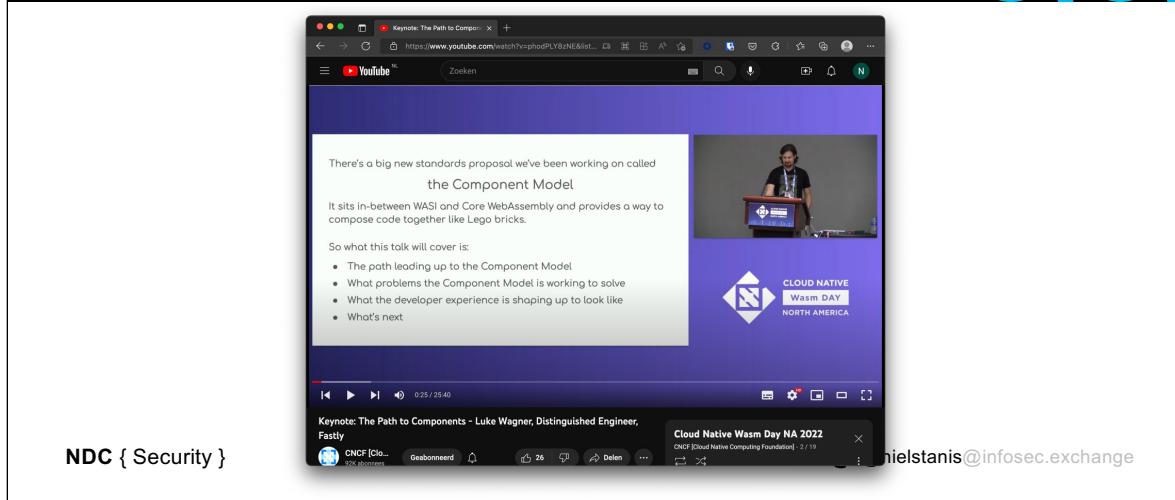


 @nielstanis@infosec.exchange

NDC { Security }

0101
0101

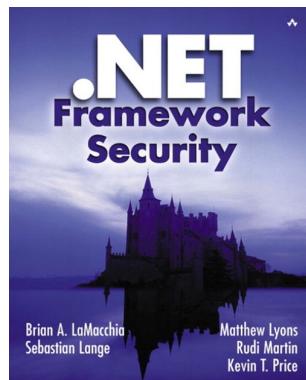
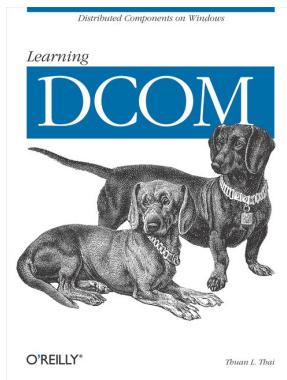
WebAssembly Component Model



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

0101
0101

Have we seen this before?



NDC { Security }

@nielstanis@infosec.exchange



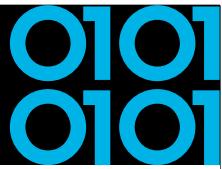
Runtimes and Security

- 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

NDC { Security }

@nielstanis@infosec.exchange

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

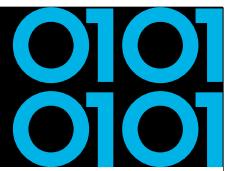


Conclusion

- Cloud Native ❤️ WebAssembly
- WebAssembly has a lot of potential to be used to run, extend, and secure your .NET applications
- It's as secure as the WebAssembly runtime implementation!
- I like top-down approach Bytecode Alliance is taking in moving forward, feedback will change

NDC { Security }

 @nielstanis@infosec.exchange



Questions?

- <https://github.com/nielstanis/ndcsecurity2023>
- ntanis at Veracode.com
- @nielstanis@infosec
- <https://blog.fennec.dev>
- Takk! Thank you!

NDC { Security }

 @nielstanis@infosec.exchange