

Using WebAssembly to
run, extend, and secure
your [.NET | Java |
Python | Go | Rust | ...]
application

Niels Tanis



WeAreDevelopers
World Congress

VERACODE

0101
0101
0101

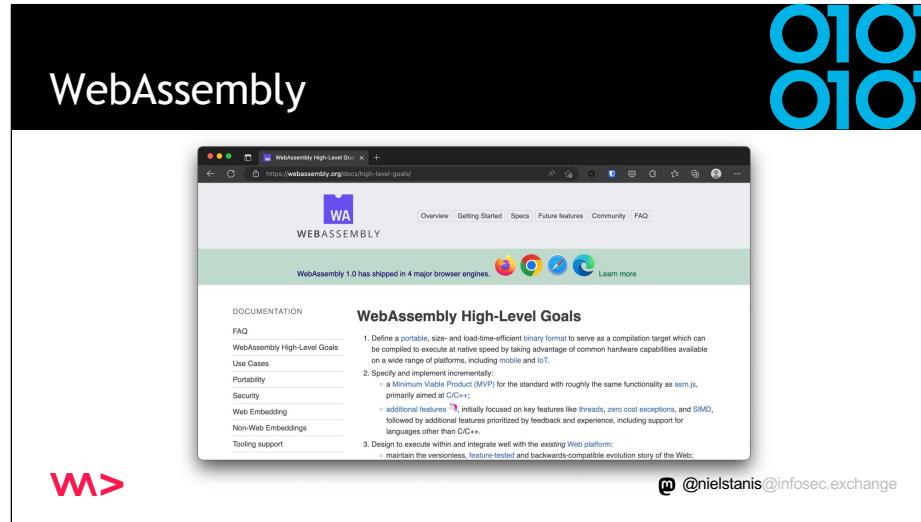
O1O1
O1O1

Who am I?

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



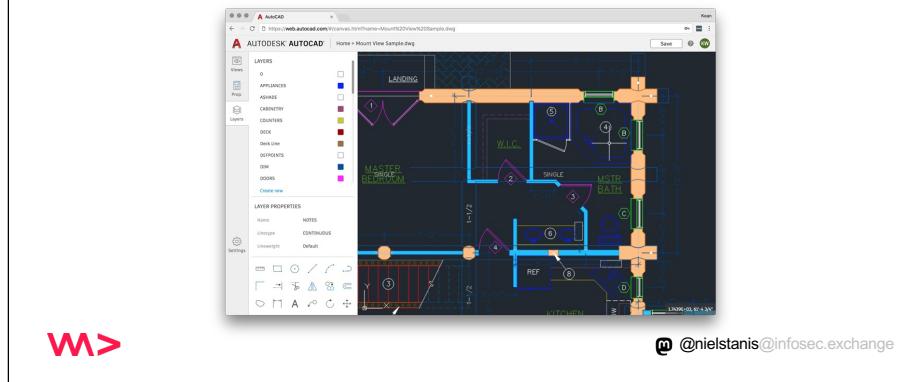
 @nielstanis@infosec.exchange



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

0101
0101

WebAssembly - AutoCAD



W>

WebAssembly - SDK's

The image shows two side-by-side screenshots of web articles. The left article is titled "Introducing the Disney+ Application Development Kit (ADK)" by Mike Harley, published on Medium.com on September 8, 2021. The right article is titled "How Prime Video updates its app for more than 8,000 device types" by Alessandro Iosu, published on Amazon Science on January 27, 2022. Both articles discuss the use of WebAssembly in their respective platforms.

W>

@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 on WebAssembly
- Extending & Securing with WebAssembly
- Conclusion
- Q&A

W>

 @nielstanis@infosec.exchange

WebAssembly Design

0101
0101

- **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



W>

@nielstanis@infosec.exchange

WebAssembly



- Binary instruction format for stack-based virtual machine similar to Java running bytecode and .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

W>

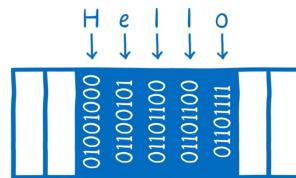
@nielstanis@infosec.exchange

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

WebAssembly Memory

0101
0101

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



W>

@nielstanis@infosec.exchange

WebAssembly Control-Flow Integrity



```
int number = Convert.ToInt32(Console.ReadLine());
Console.WriteLine($"Number {number}");
if (number>5)
{
    Console.WriteLine("Number is larger than 5");
}
else
{
    Console.WriteLine("Number is smaller than 5");
}
Console.WriteLine("Done!");
```



✉ @nielstanis@infosec.exchange

WebAssembly Control-Flow Integrity



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

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

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

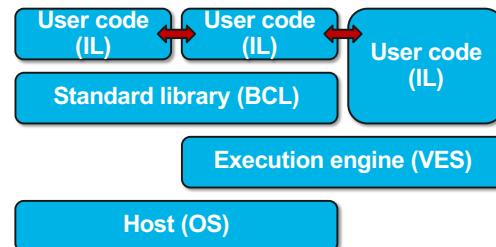
```
Console.WriteLine("Done!");
```

W>

✉ @nielstanis@infosec.exchange



Running .NET on WebAssembly



W>

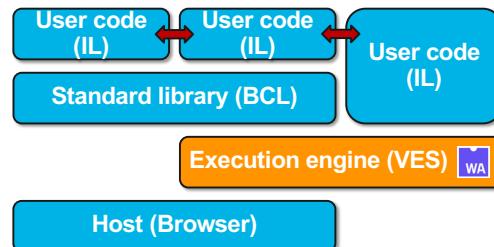
@nielstanis@infosec.exchange

Diagram:

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



Running .NET on WebAssembly



W>

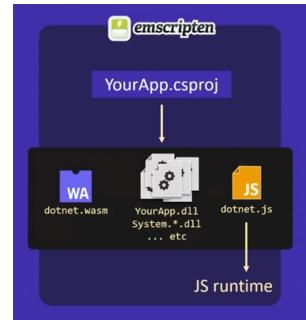
✉ @nielstanis@infosec.exchange

Diagram:

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

0101
0101

Blazor WebAssembly

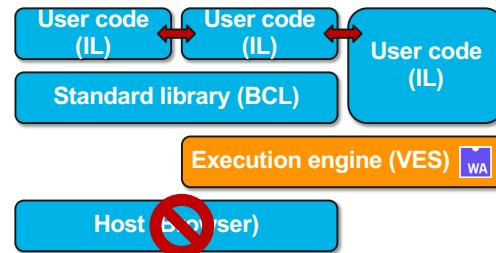


W>

@nielstanis@infosec.exchange



Running .NET on WebAssembly



W>

@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

W>

@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? 😊

W>

@nielstanis@infosec.exchange



0101
0101

Docker vs WASM & WASI

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

"So will wasm replace Docker?" No, but imagine a future where Docker runs linux containers, windows containers and wasm containers side by side. Over time wasm might become the most popular container type. Docker will love them all equally, and run it all :)

The post includes a link to <https://twitter.com/i/next/status/111111111111111>. Below the tweet, it says "Data collected overgrown".

At the bottom of the screenshot, there is a red logo consisting of a stylized 'W' followed by a right-pointing arrow (>).

On the right side of the slide, there is a small icon of a person with the handle @nielstanis@infosec.exchange.

O1O1
O1O1

Docker & WASM

The screenshot displays two browser windows side-by-side. The left window shows the 'Introducing the Docker+Wasm Technical Preview' page, featuring a title, author information (Michael Irwin, Oct 24 2022), and a note about the availability of the preview. The right window shows a diagram of the Docker Engine architecture and a terminal command example.

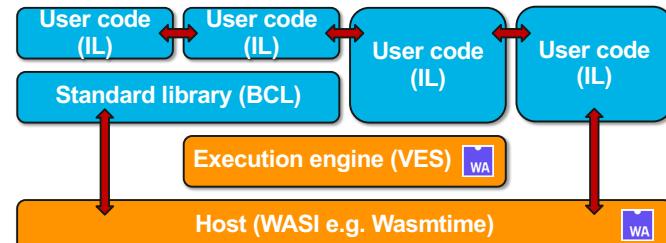
Docker Engine Architecture Diagram:

```
graph TD; DE[Docker Engine] --> container[container]; container --> cshim1[containerd-shim]; container --> cshim2[containerd-shim]; container --> cshim3[containerd-wasm-shim]; cshim1 --> runc1[runc]; cshim2 --> runc2[runc]; cshim3 --> wasmedge[wasmedge]; runc1 --> CP1[Container process]; runc2 --> CP2[Container process]; wasmedge --> WM[Wasm Module]
```

Terminal Command Example:

```
docker run -it 8888:8888 --openwasm-exefile --runtimeinfo.io.containerd.wasmedge.v1 -g /opt/farmware/Lesson32 MichaelLinux24/wasm-example
```

WebAssembly System Interface WASI

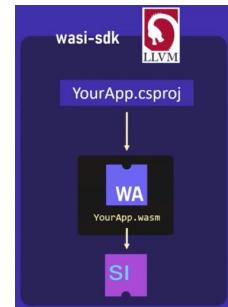


W>

@nielstanis@infosec.exchange



Experimental WASI SDK for .NET



W>

@nielstanis@infosec.exchange

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



Extending .NET with WASM

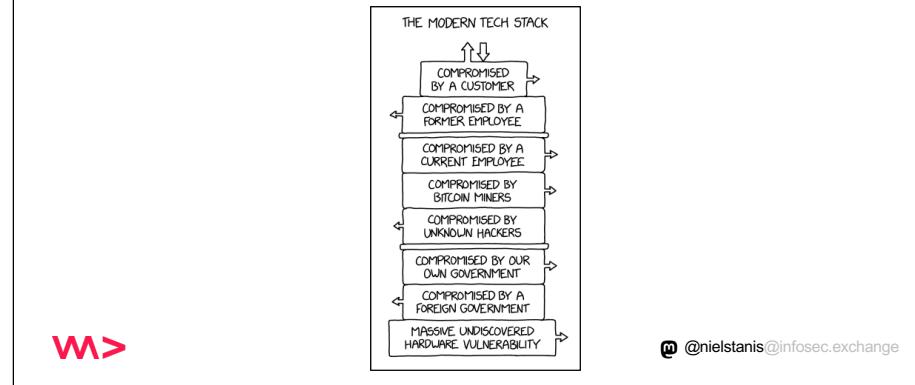
- WasmTime.NET NuGet package
- Can run WASM inside of any .NET application
- Extend with Rust based WASM module
- Demo time!



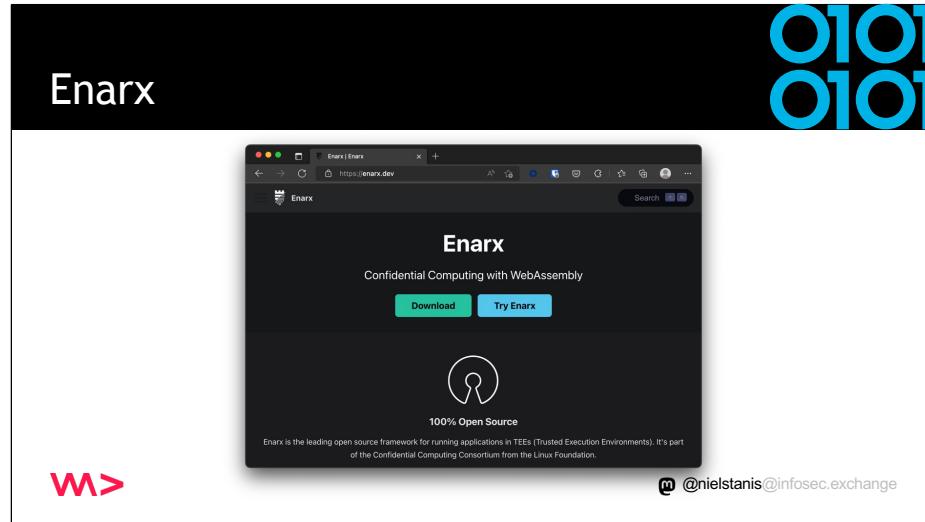
 @nielstanis@infosec.exchange

Trusted Computing - XKCD 2166

0101
0101



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



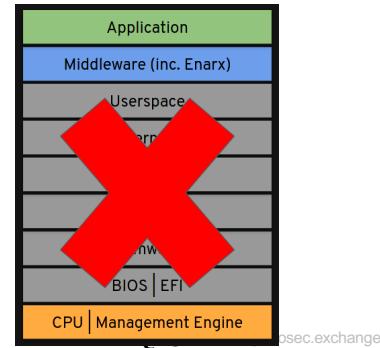
<https://enarx.dev/>



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

W>



WASM - What's next?

0101
0101

composition of an
average code base



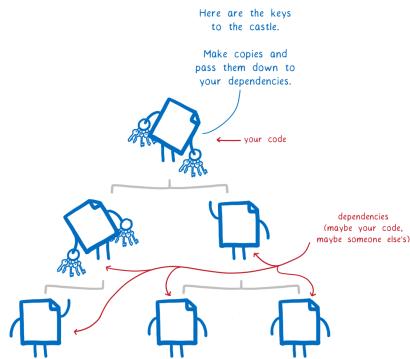
W>

@nielstanis@infosec.exchange

Dependencies

0101
0101

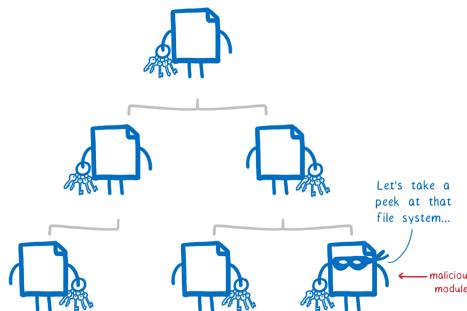
W>



iis@infosec.exchange

Malicious module

0101
0101

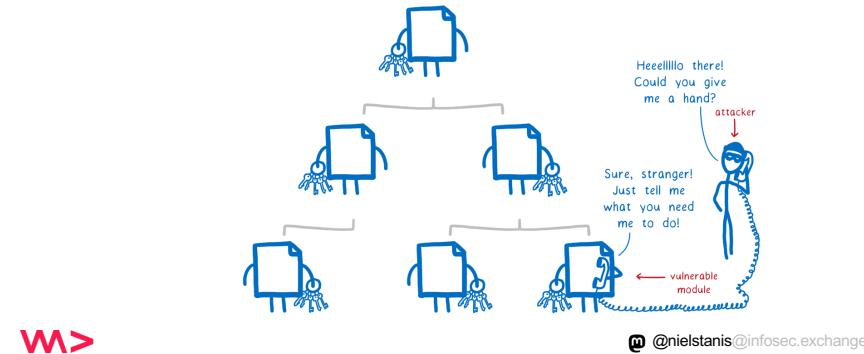


W>

@nielstanis@infosec.exchange

Vulnerable module

0101
0101

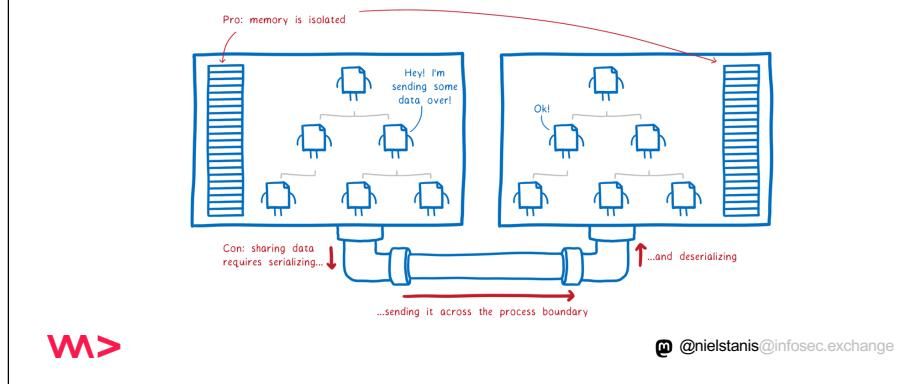


W>

@nielstanis@infosec.exchange

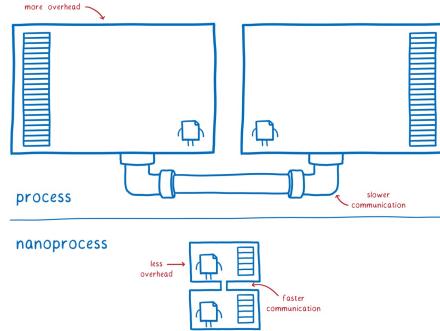
0101
0101

Process Isolation



WebAssembly Nano-Process

0101
0101



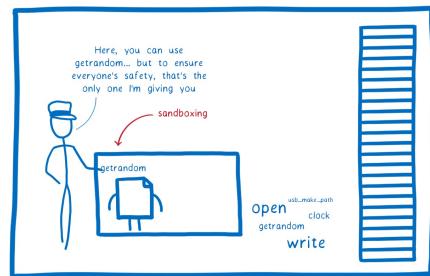
W>

* not drawn to scale
@nielstanis@infosec.exchange

0101
0101

WebAssembly Nano-Process

1. Sandboxing



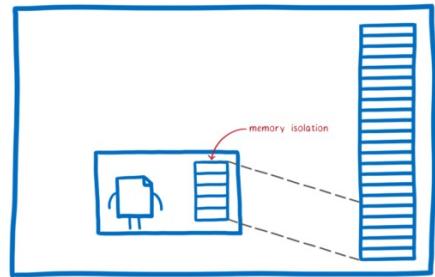
@nielstanis@infosec.exchange

W>

WebAssembly Nano-Process

0101
0101

2. Memory model



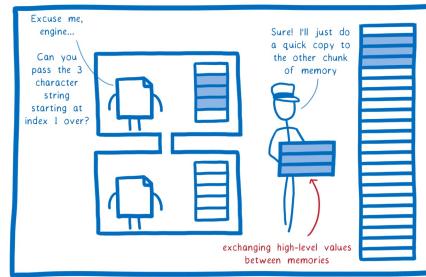
✉ @nielstanis@infosec.exchange

W>

0101
0101

WebAssembly Nano-Process

3. Interface Types



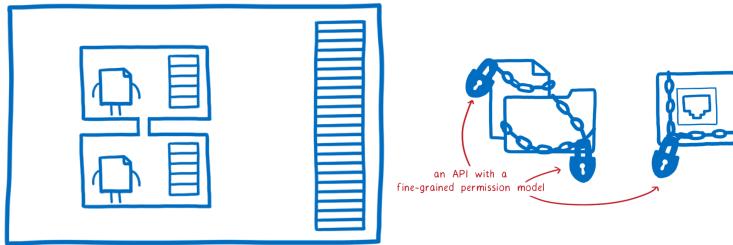
✉ @nielstanis@infosec.exchange

W>

0101
0101

WebAssembly Nano-Process

4. WebAssembly System Interface



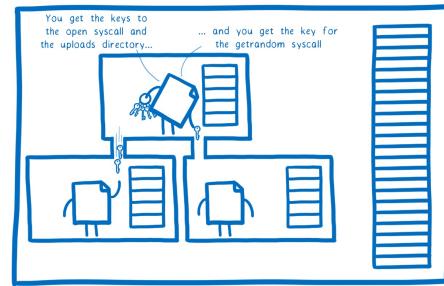
W>

@nielstanis@infosec.exchange

WebAssembly Nano-Process

0101
0101

5. The missing link

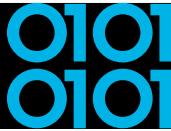


@nielstanis@infosec.exchange

W>

The slide has a dark background with a large blue '0101' logo in the top right corner. The title 'WebAssembly Component Model' is centered at the top in white. Below the title are two YouTube video thumbnails side-by-side. The left thumbnail shows a speaker at a podium with a 'CNCF Native Components' logo, and the right thumbnail shows a diagram of a 'WebAssembly component' with various interfaces. At the bottom left is a red 'W>' logo.

<https://www.youtube.com/watch?v=phodPLY8zNE>
<https://www.youtube.com/watch?v=JClwpc7x4jU>



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

W>

@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
- Its as secure as the WebAssembly runtime implementation!
- I like top-down approach Bytecode Alliance is taking in moving forward, feedback will change/influence

W>

 @nielstanis@infosec.exchange



Questions?

- <https://github.com/nielstanis/wearedevs2023>
- ntanis at Veracode.com
- @nielstanis@infosec.exchange
- <https://blog.fennec.dev>
- Danke! Thank you!
- Want to talk more about Veracode?
Our booth is at A51!

W>

 @nielstanis@infosec.exchange