

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

Niels Tanis

 Update Conference
Prague 2022

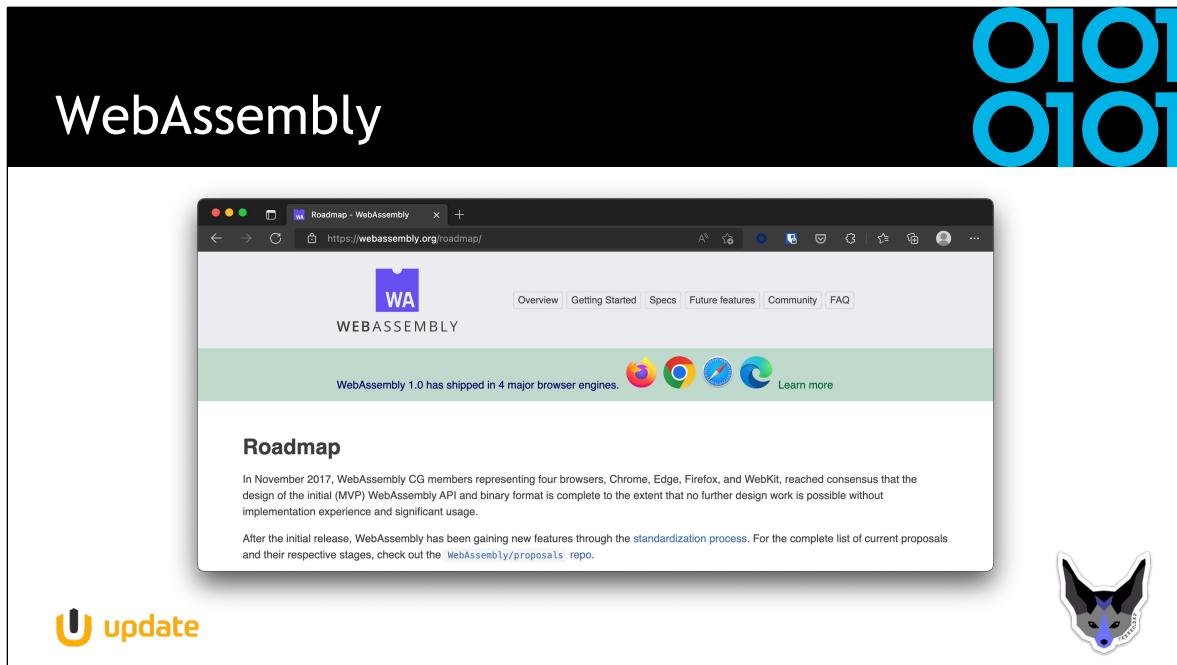


0101
0101

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

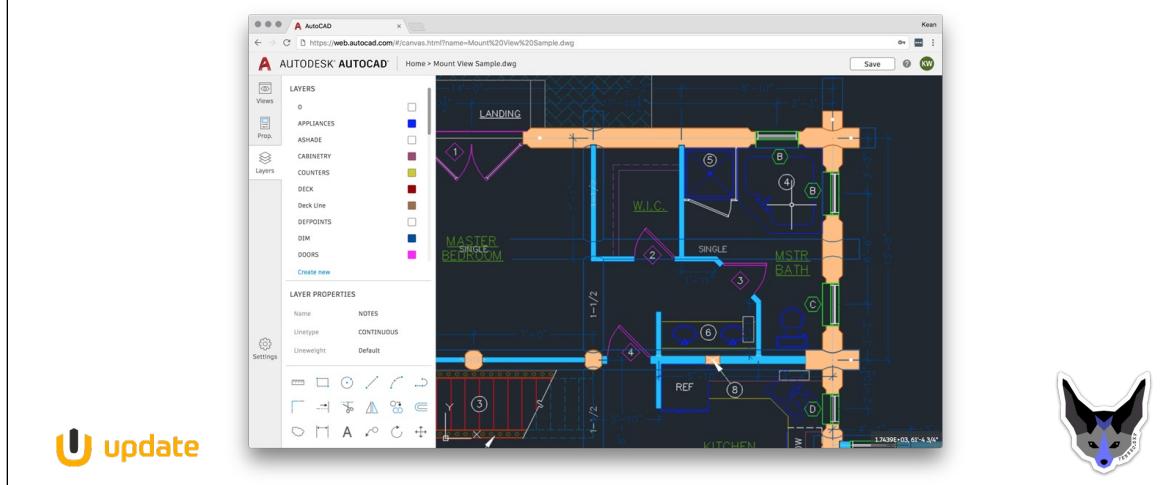




<https://hacks.mozilla.org/files/2019/08/04-01-star-diagram.png>

0101
0101

WebAssembly - AutoCAD



U update

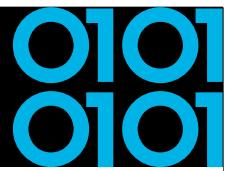
0101 0101

WebAssembly - SDK's

The image shows two side-by-side screenshots of web browser windows. The left window is from Medium.com, titled 'Introducing the Disney+ Application Development Kit (ADK)', published by Mike Hanley on Sep 8, 2021. The right window is from Amazon Science, titled 'How Prime Video updates its app for more than 8,000 device types', published by Alexandru Ene on January 27, 2022. Both articles discuss the use of WebAssembly. At the bottom left is a logo for 'U update' with a stylized 'U'. At the bottom right is a small cartoon fox head wearing a blue collar with the word 'FIREWALL' on it.

<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



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



WebAssembly

0101
0101



- 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

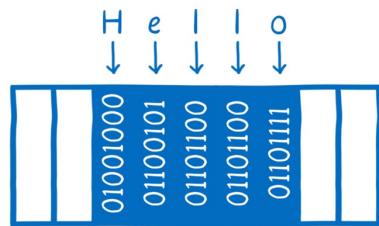


<https://hacks.mozilla.org/2017/02/creating-and-working-with-webassembly-modules/>
<https://webassembly.org/>

0101
0101

WebAssembly Memory

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



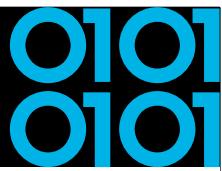
WebAssembly Control-Flow Integrity



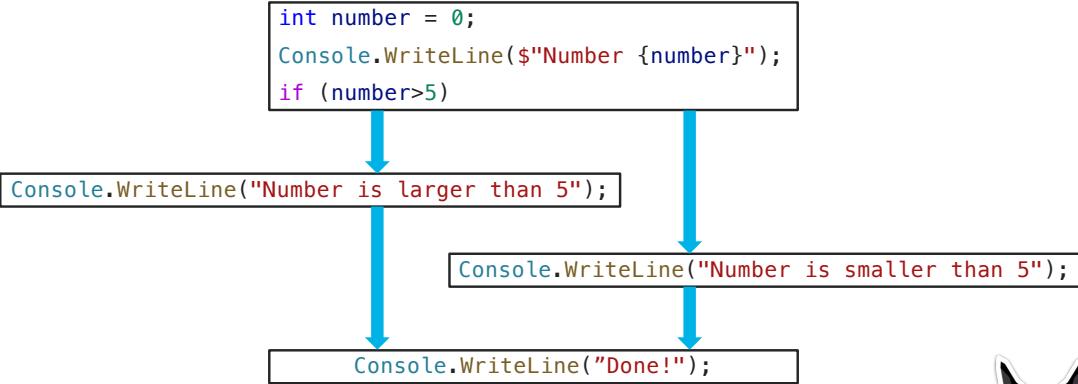
```
int number = 0;
Console.WriteLine($"Number {number}");
if (number>5)
{
    Console.WriteLine("Number is larger than 5");
}
else
{
    Console.WriteLine("Number is smaller than 5");
}
Console.WriteLine("Done!");
```

update





WebAssembly Control-Flow Integrity

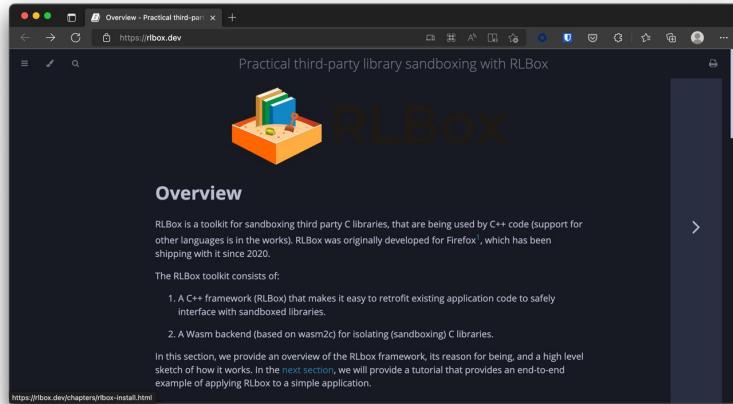


U update



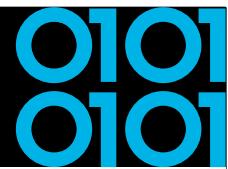
FireFox RLBox

0101
0101



<https://rlbox.dev/>

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



Running .NET on WebAssembly

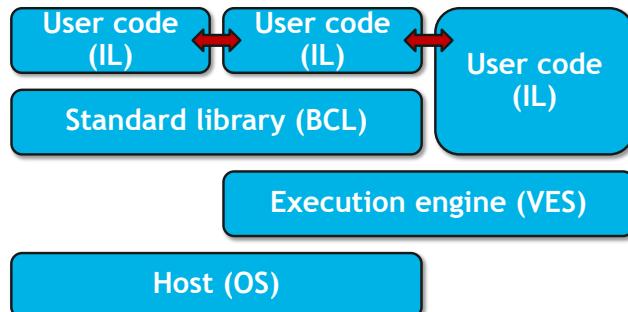
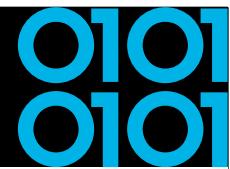


Diagram:

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



Running .NET on WebAssembly

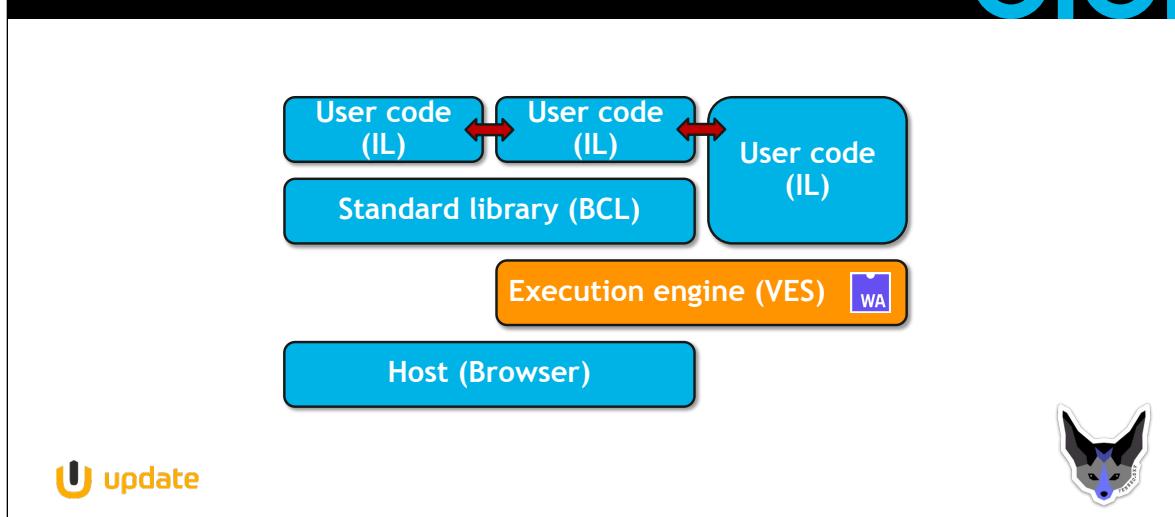


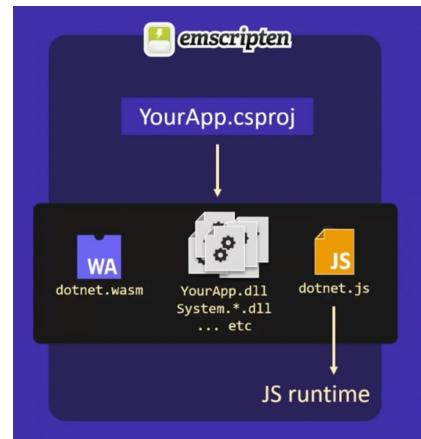
Diagram:

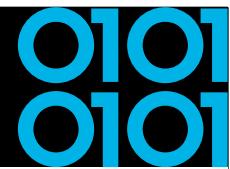
<https://github.com/itowlson/wasmday22/blob/main/slides/Wasm%20Interfaces%20and%20.NET.pptx>

0101
0101

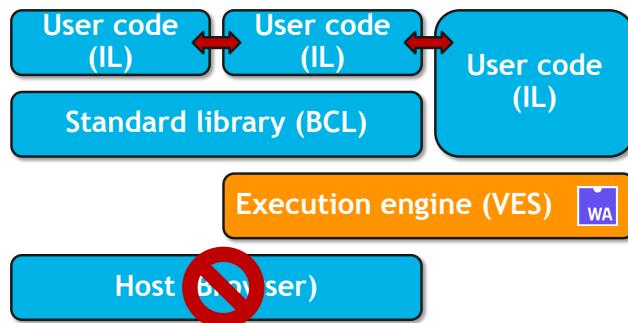
Blazor WebAssembly

 update





Running .NET on WebAssembly

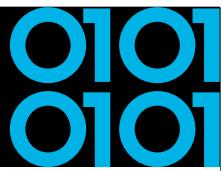


 update



Diagram:

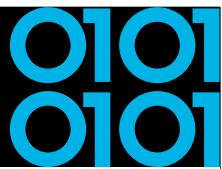
<https://github.com/itowlson/wasmday22/blob/main/slides/Wasm%20Interfaces%20and%20.NET.pptx>



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





WebAssembly System Interface WASI

- Strong sandbox with Capability Based Security
- Right now, supports 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 interface. The main tweet is from Solomon Hykes (@solomonstre) dated March 27, 2019, at 9:39 PM. It 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 this tweet is a reply from Lin Clark (@linclark) also dated March 27, 2019, at 9:39 PM. 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... Deze collectie weergeven". The interface includes a sidebar with icons for Twitter, Home, Search, and Direct Messages. At the bottom left is a logo for 'U update'. On the right side of the screen is a small, stylized fox head icon.

0101
0101

Docker vs WASM & WASI

A screenshot of a Twitter post from Solomon Hykes (@solomonstrel) dated March 27, 2019. The tweet discusses the future of containerization, stating: "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 [twi tter.com/linc Clark/status/111111111111111111](https://twitter.com/linc Clark/status/111111111111111111). The Twitter interface shows 56 Retweets, 5 Geciteerde Tweets, and 165 Vind-ik-leuks.

U update

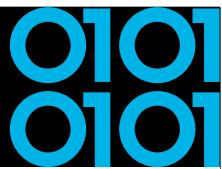
update



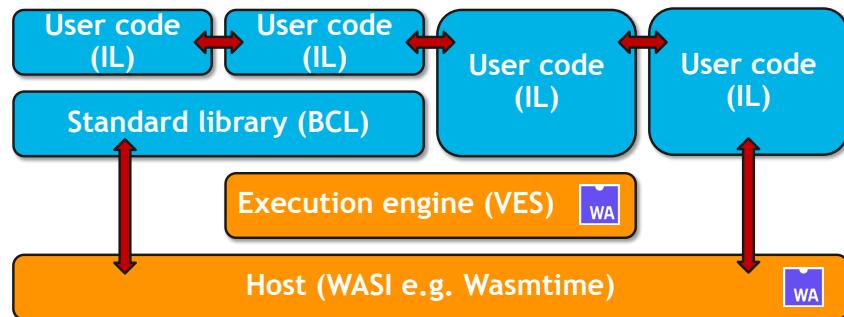
0101
0101

Docker & WASM

The screenshot shows two browser tabs side-by-side. The left tab displays the homepage for 'Introducing the Docker+Wasm Technical Preview'. It features a header with the Docker logo and '+WASM', followed by the title 'Introducing the Docker+Wasm Technical Preview'. Below the title is a bio for Michael Irwin, a photo, 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.' The right tab shows a detailed architectural diagram of the Docker+Wasm stack. At the top is the 'Docker Engine', which oversees a 'container'. Inside the container, there are three 'containerd-shim' components, each containing a 'runc' process. One of these containers is highlighted with a green border. This container also contains a 'Container process'. To the right of this container is another 'containerd-shim' component, which contains a 'runc' process and a 'Container process'. Next to this is a third 'containerd-wasm-shim' component, which contains a 'wasm-edge' process and a 'Wasm Module'. Below the diagram, a section titled 'Let's look at an example!' provides a 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`. A small cartoon fox icon with the word 'TESTCAST' is located in the bottom right corner of the slide.



WebAssembly System Interface WASI



update



0101
0101

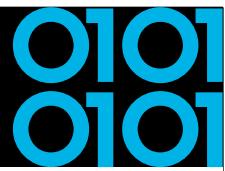
Experimental WASI SDK for .NET



 update



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



Extending .NET with WASM

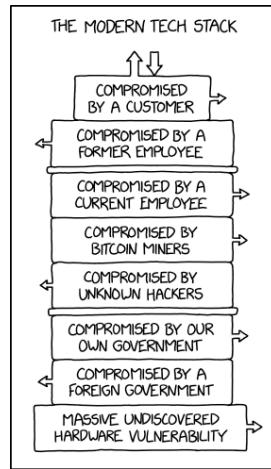
- WasmTime.NET NuGet package
- Can run WASM inside of any .NET application
- Extend with Rust based WASM module
- Demo time!



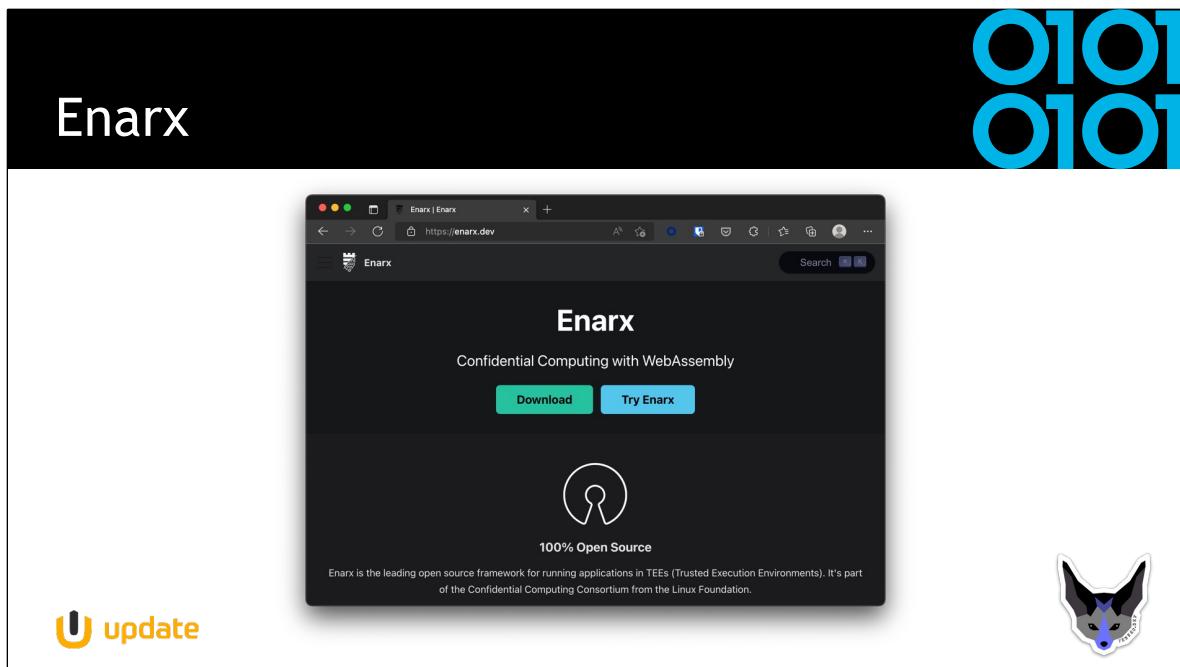
0101
0101

Trusted Computing - XKCD 2166

 update



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

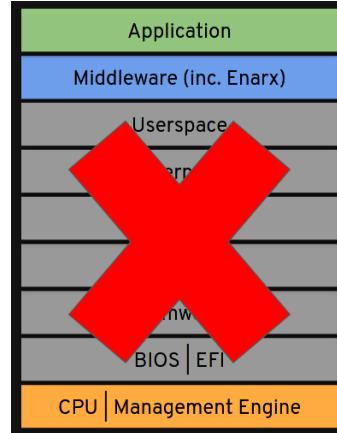


<https://enarx.dev/>

0101
0101

Enarx Threat Model

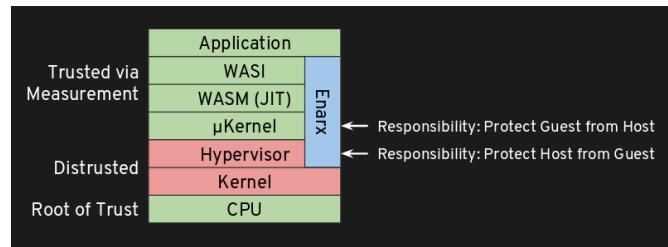
- Don't trust the host
- Don't trust the host owner
- Don't trust the host operator
- Hardware cryptographically verified
- Software audited and cryptographically verified





Enarx

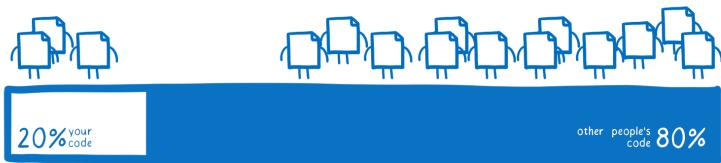
- Leverages Trusted Execution Environment (TEE) direct on processor
 - AMD's SEV, Intel's SGX and IBM's PEF
- Attestation of hardware and Enarx runtime



WASM - What's next?

0101
0101

composition of an
average code base



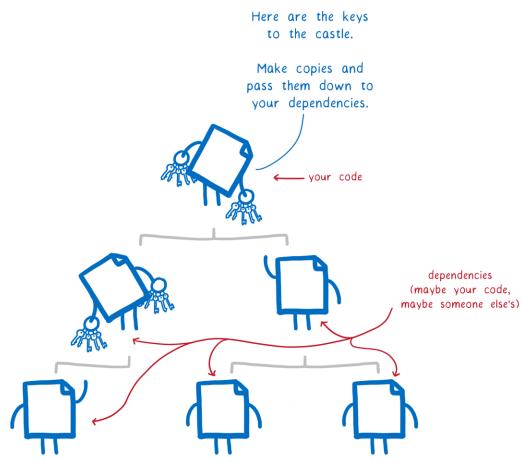
 update



0101
0101

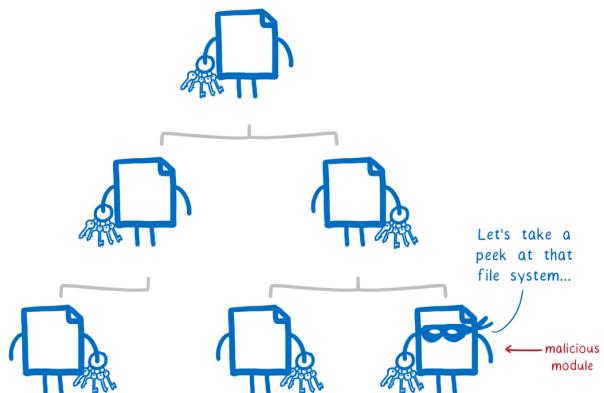
Dependencies

 update



0101
0101

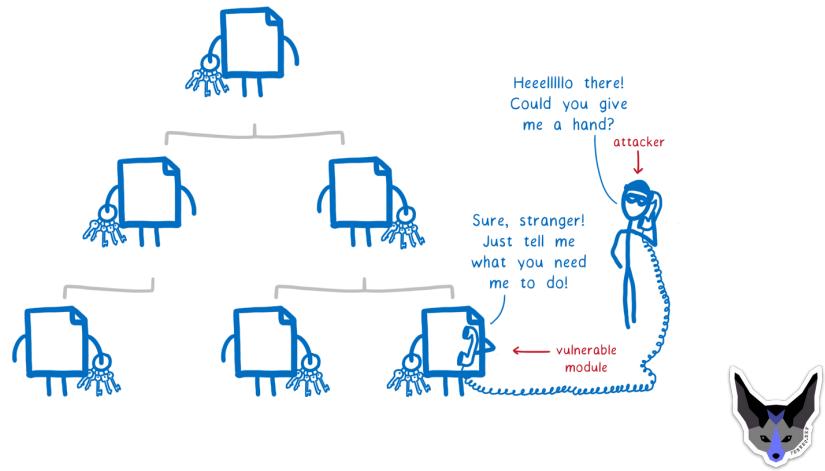
Malicious module



 update

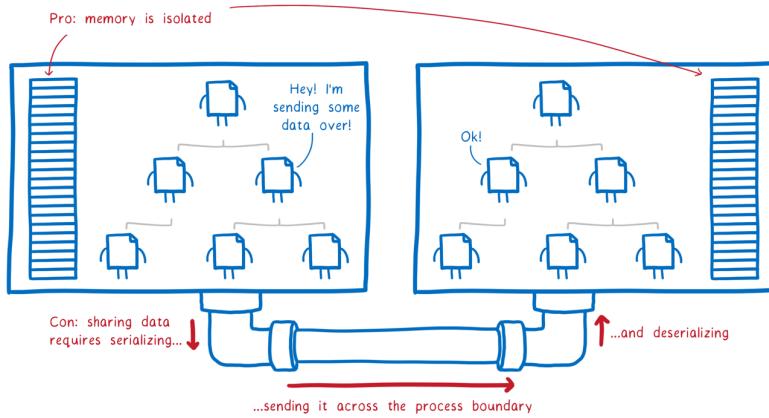
0101
0101

Vulnerable module



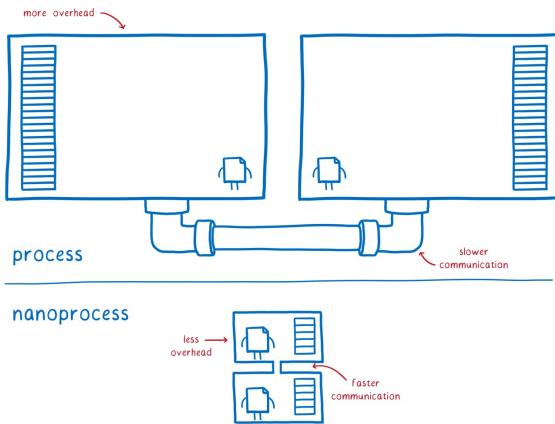
0101
0101

Process Isolation



0101
0101

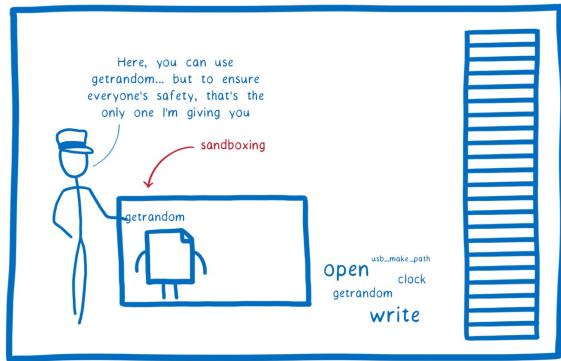
WebAssembly Nano-Process



0101
0101

WebAssembly Nano-Process

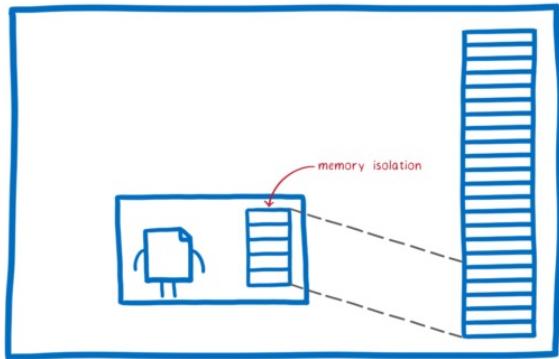
1. Sandboxing



0101
0101

WebAssembly Nano-Process

2. Memory model



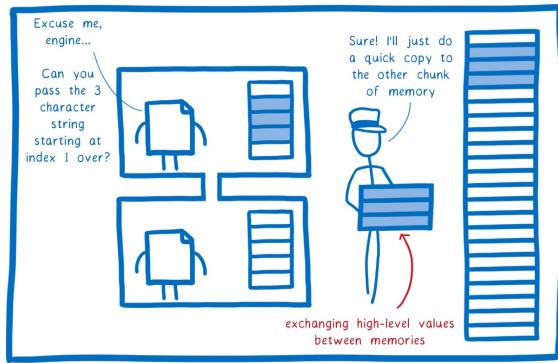
U update



0101
0101

WebAssembly Nano-Process

3. Interface Types



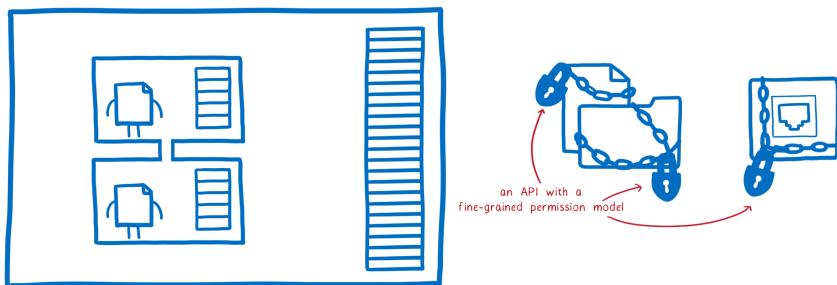
update



0101
0101

WebAssembly Nano-Process

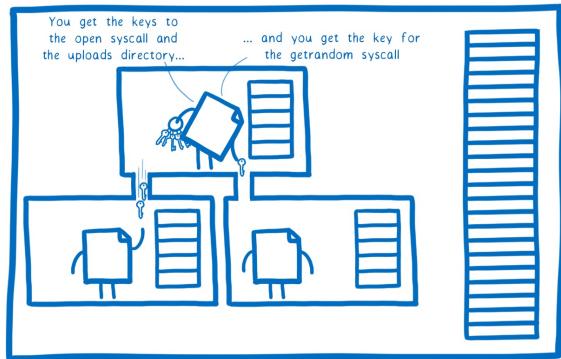
4. WebAssembly System Interface



0101
0101

WebAssembly Nano-Process

5. The missing link



update



0101
0101

WebAssembly Component Model

There's a big new standards proposal we've been working on called the Component Model. It sits in-between WASI and Core WebAssembly and provides a way to compose code together like Lego bricks.

So what this talk will cover is:

- The path leading up to the Component Model
- What problems the Component Model is working to solve
- What the developer experience is shaping up to look like
- What's next

Keynote: The Path to Components - Luke Wagner, Distinguished Engineer, Fastify

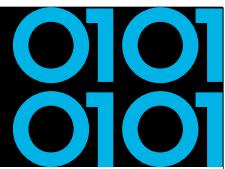
CLOUD NATIVE Wasm DAY NORTH AMERICA

Cloud Native Wasm Day NA 2022

0:25 / 25:40

U update



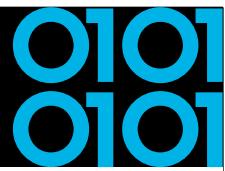


Runtimes and Security

- 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



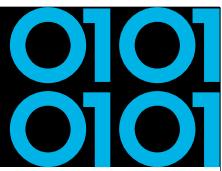
<https://bytecodealliance.org/articles/security-and-correctness-in-wasmtime>



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





Questions?

- <https://github.com/nielstanis/updateconference2022>
- ntanis at Veracode.com
- <https://blog.fennec.dev>
- Děkuji! Thank you!

