Using WebAssembly to run, extend and secure your app

Niels Tanis
Sr. Principal Security Researcher

VERACODE

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

SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

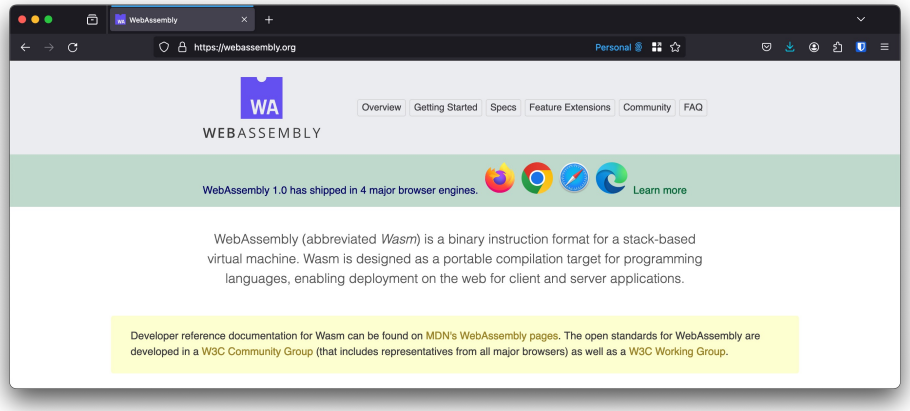
Agenda

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

SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

WebAssembly

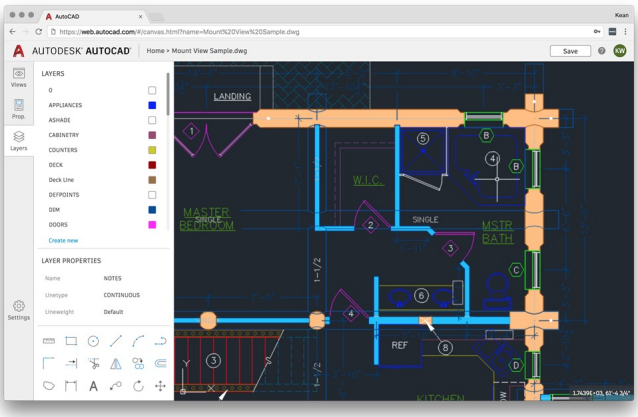


SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

https://webassembly.org/

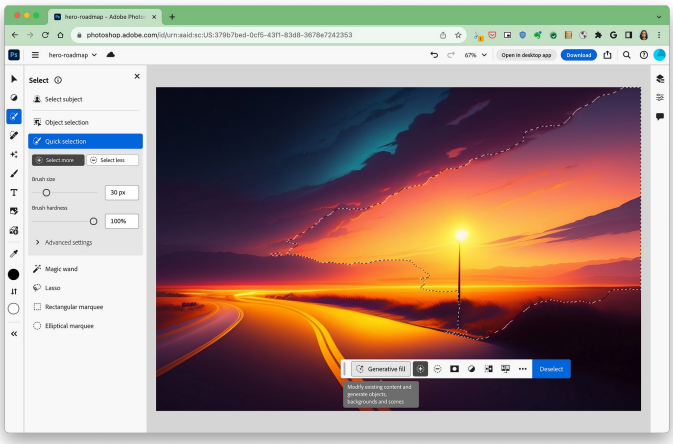
WebAssembly - AutoCAD



SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

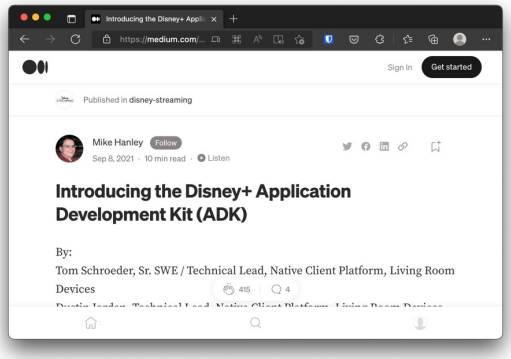
WebAssembly - Photoshop

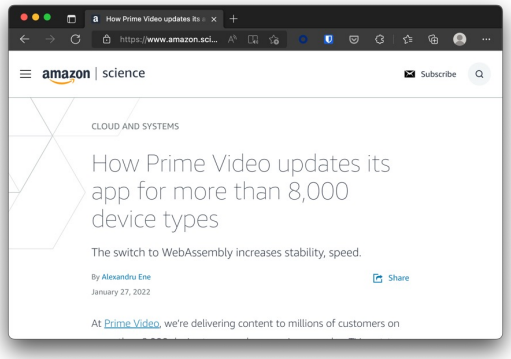


SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

WebAssembly - SDK's



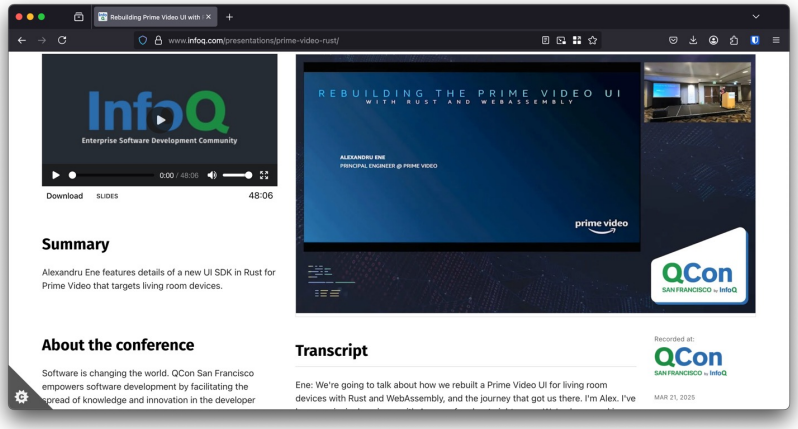


SecAppDev

@niels.fennec.dev @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

Prime Video UI - Rust & WASM



SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

https://www.infoq.com/presentations/prime-video-rust/

WebAssembly – Do you use it?

- Does your organization use WASM?
- Which main tech stacks used?
- WASM for what kind of apps?
  - Mobile
  - Cloud Native
  - Other…

WEBASSEMBLY

SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

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

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

- 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

SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

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

WebAssembly Type System

- WebAssembly's type only supports:
  - `i32 (32-bit integer)`
  - `i64 (64-bit integer)`
  - `f32 (32-bit float)`
  - `f64 (64-bit float)`
- No strings, no objects, no complex data types.
- Basic operations on numerical values.

WebAssembly Stack Based VM

- In a stack-based VM, operations primarily manipulate a last-in-first-out (LIFO) stack of values, rather than working with named registers as in register-based architectures.

WebAssembly Stack Based VM

- WebAssembly maintains an operand stack during execution:
  - Values are pushed onto the stack by instructions
  - Operations pop their operands from the stack
  - Results are pushed back onto the stack
  - The stack is separate from linear memory and is not directly accessible

Code some WebAssembly

Let’s create some WAT!

http://github.com/nielstanis/secappdev25wasm

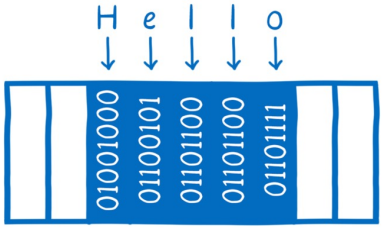
WEBASSEMBLY

SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

WebAssembly Memory

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



SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

Code some WebAssembly

Let’s work with some Memory!

WEBASSEMBLY

SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

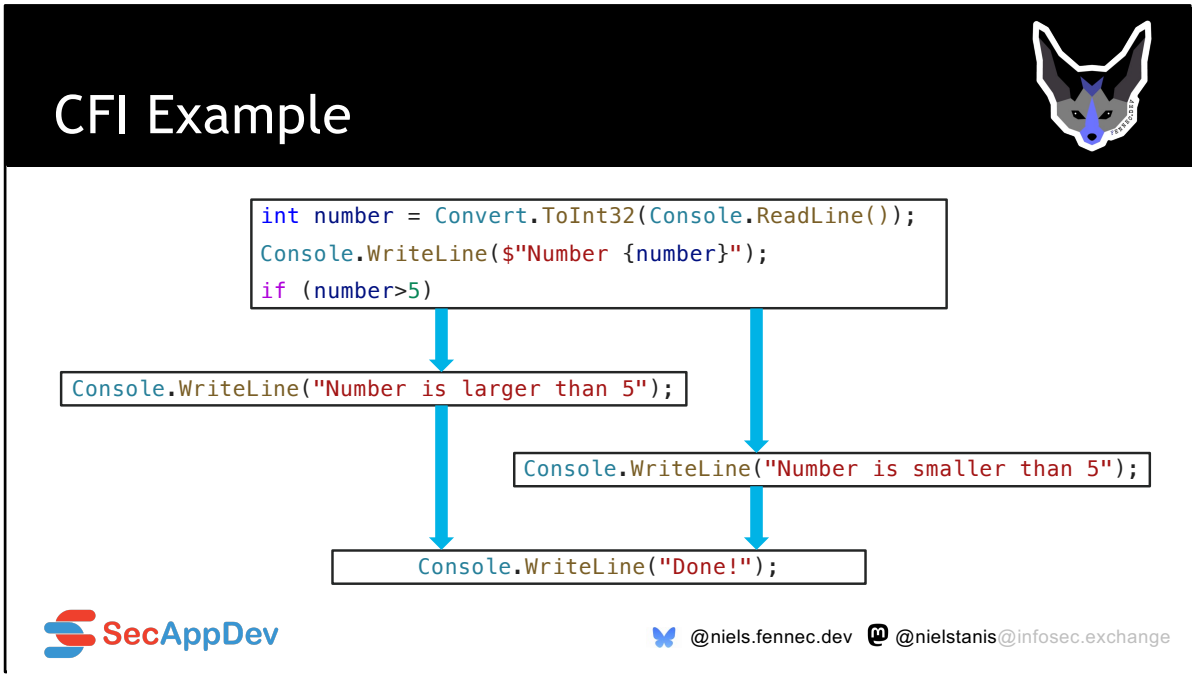
WebAssembly Control-Flow Integrity

- Control-Flow Integrity (CFI) ensures that program execution follows only valid paths as defined by the program's source code.
- In traditional native code, attackers can exploit memory vulnerabilities to hijack execution flow:
  - Redirecting it to malicious code
  - Chaining together existing code fragments in unintended ways (like Return-Oriented Programming attacks).

CFI Example

```
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!");
```

CFI Example



SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

CFI: Structured Control-Flow Only

- No arbitrary jump instructions (no goto, jmp or equivalent)
- All branches have explicit, validated targets within the same function
- No ability to jump to arbitrary memory addresses
- All control flow is validated before execution

CFI: Protected Function Calls

```
(module
  (func $safe_function (param i32) (result i32)
    local.get 0
    i32.const 1
    i32.add
  )

  (func $caller (param i32) (result i32)
    local.get 0
    call $safe_function  ;; Direct call - target validated at compile time
  )
)
```

SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

CFI: Indirect Calls

- Function table entries are validated at module instantiation
- Runtime checks ensure the target index is within table bounds
- Runtime type checking ensures the called function matches the expected signature
- The table is protected from direct memory manipulation

Code some WebAssembly

Indirect calling!

WEBASSEMBLY

SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

CFI: Indirect Calls

```
(module
  (table 2 funcref)
  (func $f1 (result i32)
    i32.const 42)
  (func $f2 (result i32)
    i32.const 13)
  (elem (i32.const 0) $f1 $f2)
  (type $return_i32 (func (result i32)))
  (func (export "callByIndex") (param $i i32) (result i32)
    local.get $i
    call_indirect (type $return_i32))
)
```

SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

CFI: Separation of Code and Data

- Code sections cannot be modified at runtime
- Linear memory (accessible data) cannot contain executable code
- No instruction exists to convert data to code No self-modifying code capabilities

RLBox

RLBox

SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

https://rlbox.dev/
https://hacks.mozilla.org/2020/02/securing-firefox-with-webassembly/

RLBox in Firefox

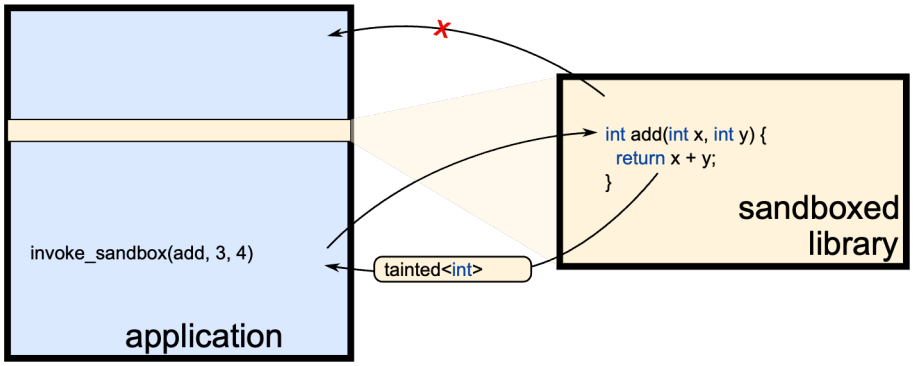
- RLBox uses **WebAssembly as an intermediate compilation target** for creating sandboxed versions of third-party C/C++ libraries.

- Rather than running WebAssembly code directly, RLBox employs a unique "**WebAssembly and Back Again**" approach.

RLBox in Firefox

- RLBox considers all values that originate in the sandbox as untrusted and "taints" them.
- Tainted values are essentially opaque values that cannot be used directly by the application code.
- RLBox's type system marks all data coming out of the sandbox as "tainted" and ensures, through compiler errors, that developers sanitize potentially unsafe data before using it.

RLBox in Firefox



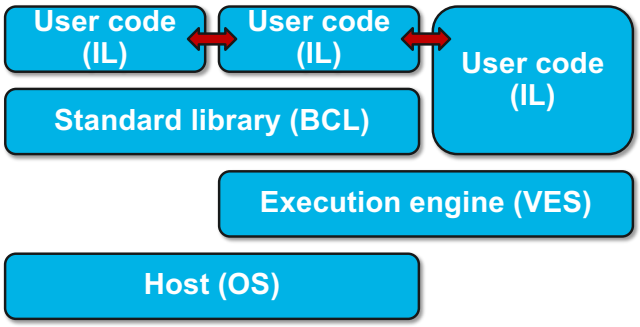
SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

RLBox in Firefox

- RLBox is currently deployed in Firefox to isolate five different modules:
  - Graphite – Font rendering engine - Firefox 95
  - Hunspell – Spell checker – Firefox 95
  - Ogg - Multimedia container format – Firefox95
  - Expat - XML parser – Firefox 96
  - Woff2 – Web font compression format - Firefox 96

Running .NET on WebAssembly

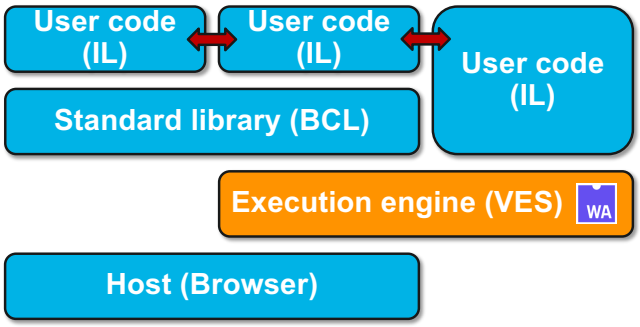


SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

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

Running .NET on WebAssembly

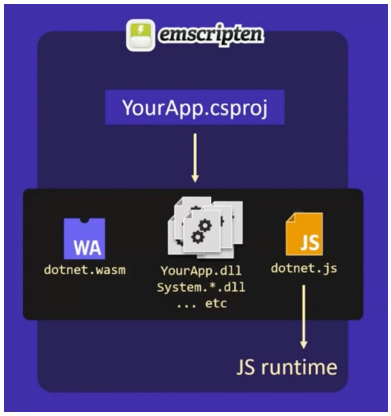


SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

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

Blazor WebAssembly

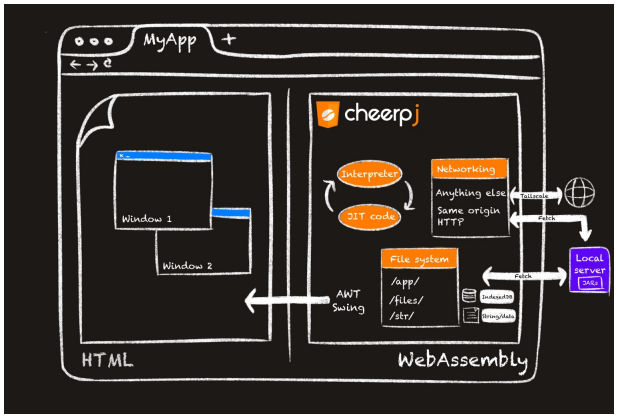


SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

JVM on WASM with CheerpJ

The magic behind CheerpJ is Cheerp, which was used for compiling full Java SE 8 and Java SE 11 runtimes based on OpenJDK.

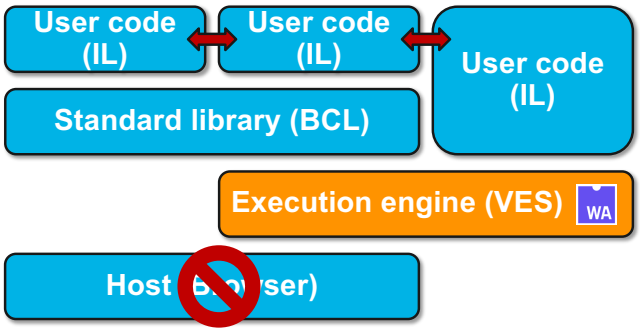


SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

https://cheerpj.com/docs/explanation/architecture
https://cheerpjdemos.leaningtech.com/PHETDemos.html#demo

Running .NET on WebAssembly



SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

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

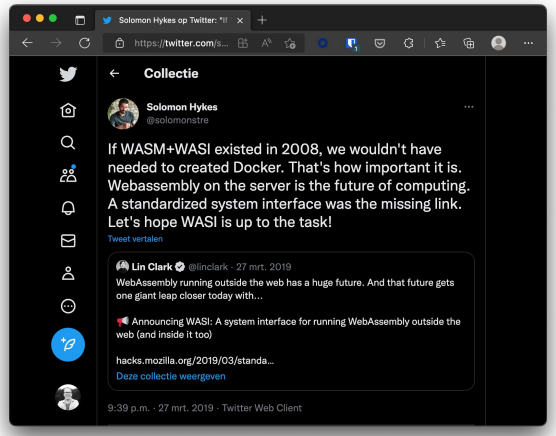
WebAssembly System Interface WASI

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

WebAssembly System Interface WASI

- Strong sandbox with Capability Based Security
- Right now, supports e.g. FileSystem actions, Sockets, CLI and HTTP at version 0.2
- Future support for promise/async and streams
- Anyone recall .NET Standard? ☺

Docker vs WASM & WASI



SecAppDev

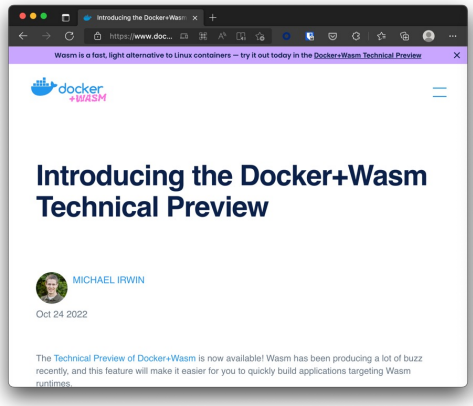
@niels.fennec.dev @nielstanis@infosec.exchange

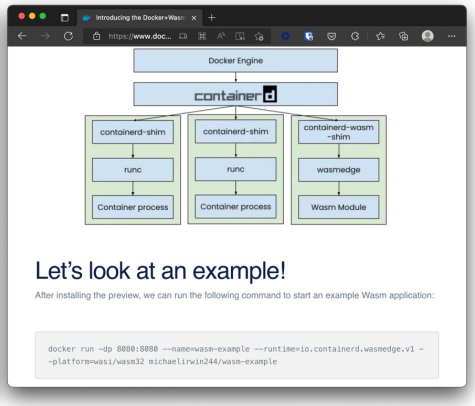
Docker vs WASM & WASI

SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

Docker & WASM

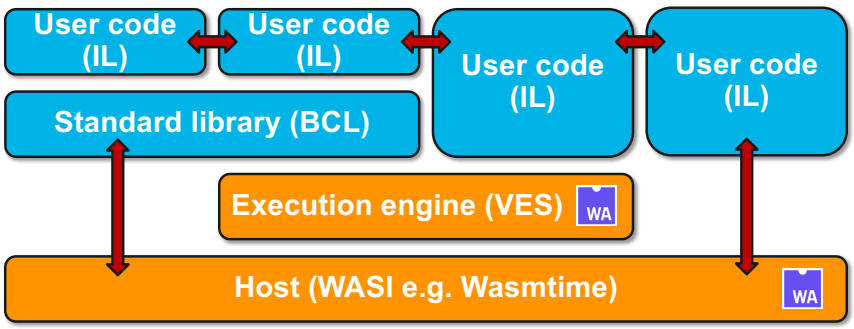




SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

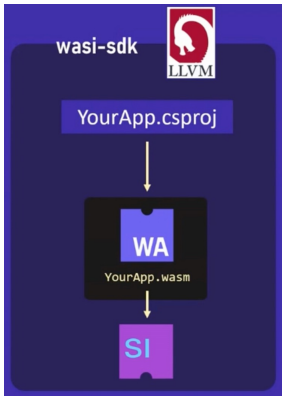
WebAssembly System Interface WASI



SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

Experimental WASI SDK for .NET

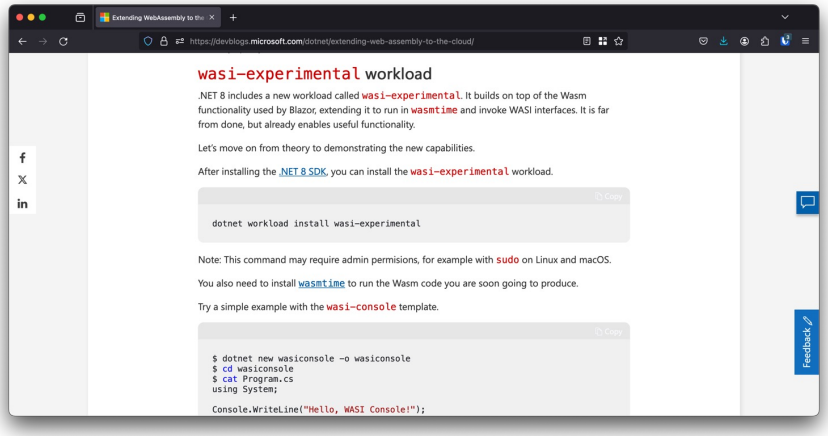


SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

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

.NET 8 WASI-Experimental



SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

https://github.com/dotnet/runtime/issues/65895
https://github.com/SteveSandersonMS/dotnet-wasi-sdk

https://devblogs.microsoft.com/dotnet/extending-web-assembly-to-the-cloud/

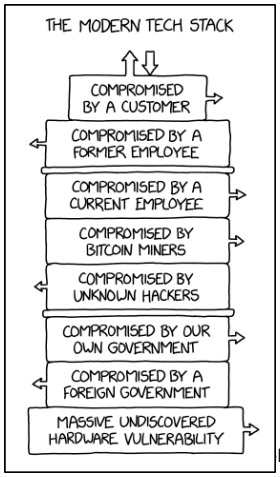
Extending .NET with WASM

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

SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

Trusted Computing – XKCD 2166

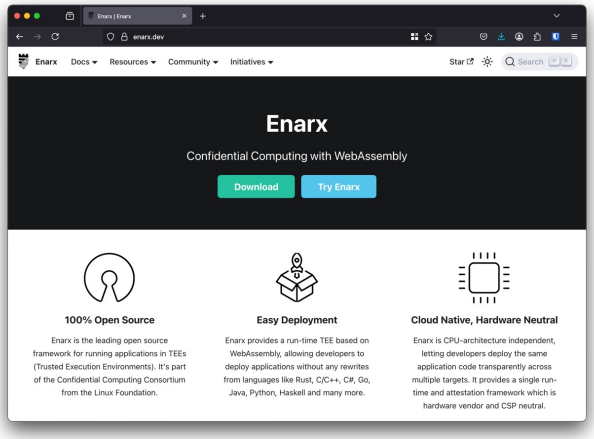


SecAppDev

ls.fennec.dev @nielstanis@infosec.exchange

https://xkcd.com/2166/

Enarx



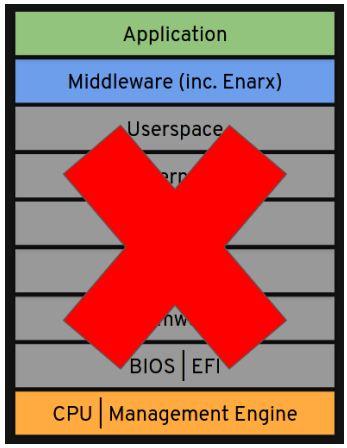
SecAppDev

@niels.fennec.dev @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

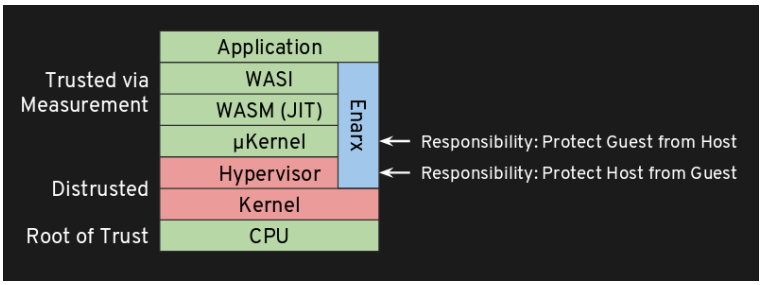


SecAppDev

@niels.fennec.dev @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



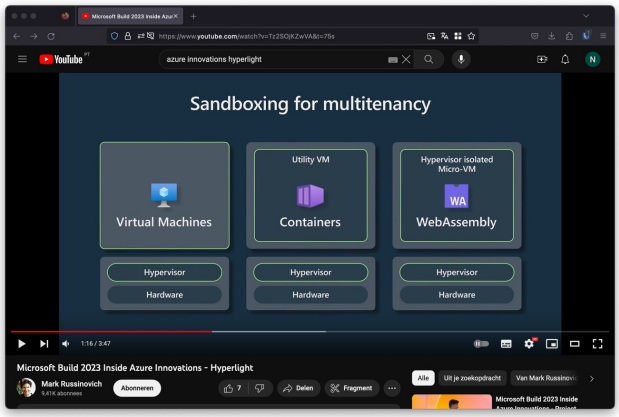
SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

Enarx

- Each execution:
  - Attestation: Enarx checks that the host to which you're planning to deploy is a genuine TEE instance.
  - Packaging: Once the attestation is complete and the TEE instance verified, the Enarx management component encrypts the application, along with any required data.
  - Provisioning: Enarx then sends the application and data along to the host for execution in the Enarx Keep.
- TEE provides: Data Confidentiality, Data Integrity, Code Integrity

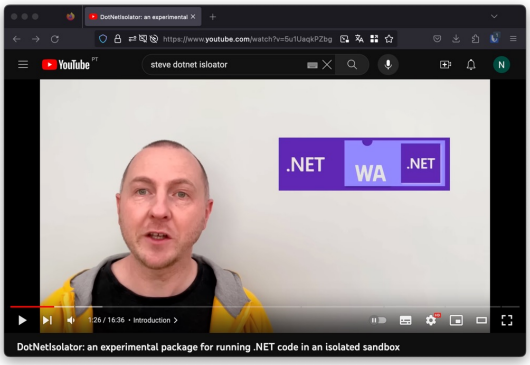
Project Hyperlight

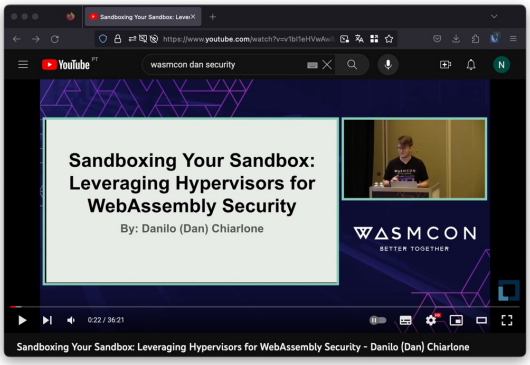


SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

DotNetIsolator & Project Hyperlight

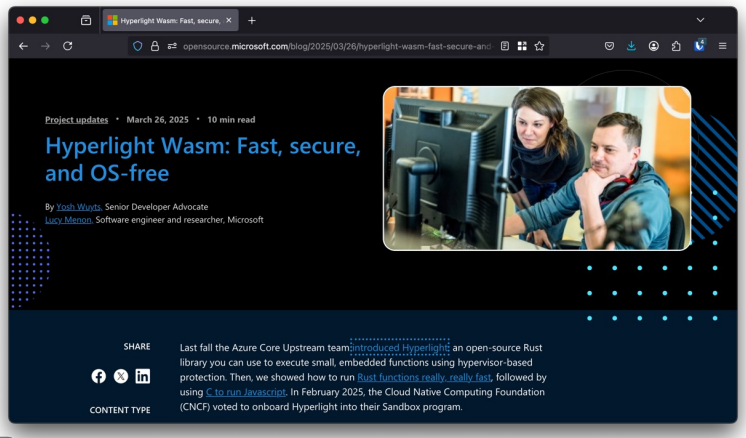




SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

Hyperlight CNCF Sandbox

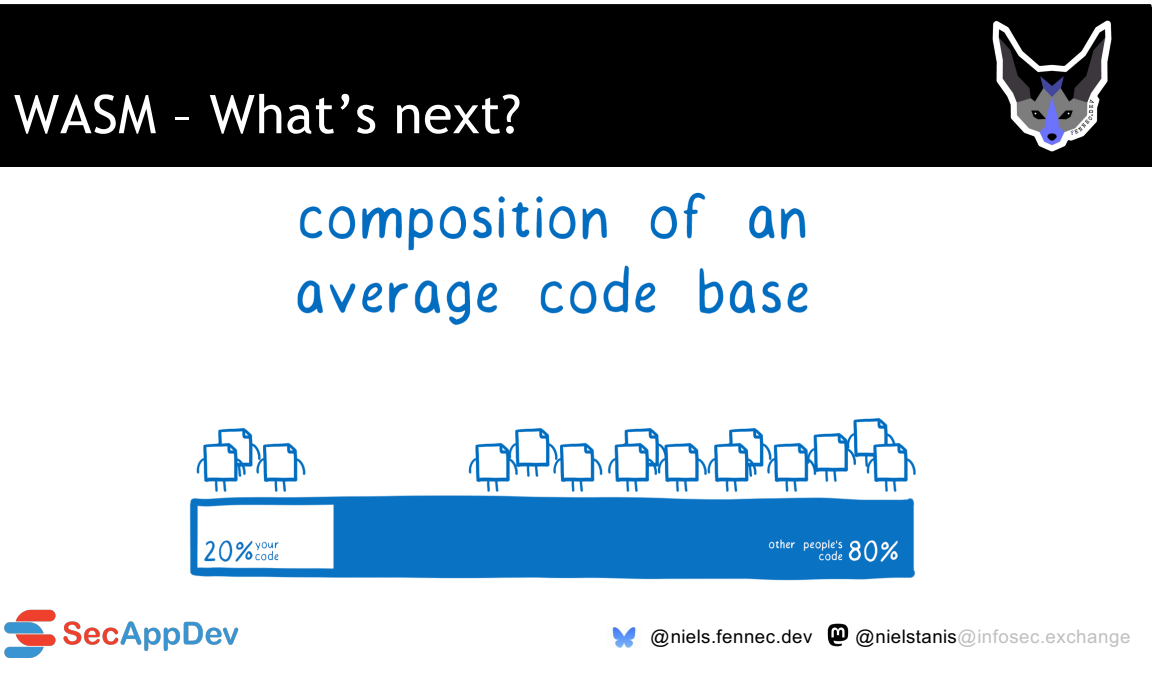


SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

https://opensource.microsoft.com/blog/2025/03/26/hyperlight-wasm-fast-secure-and-os-free/

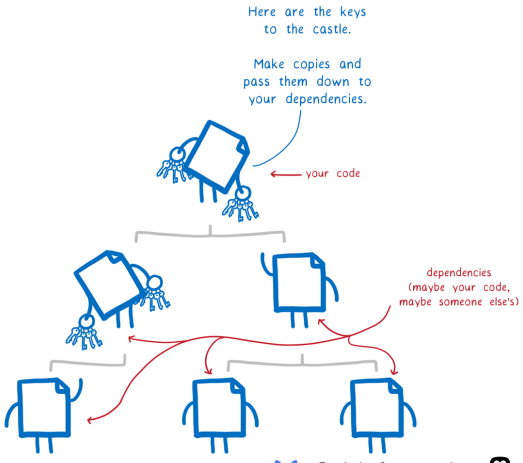
WASM – What's next?



SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

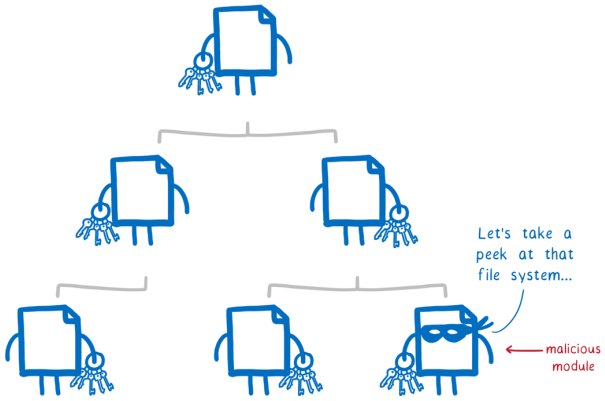
Dependencies



SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

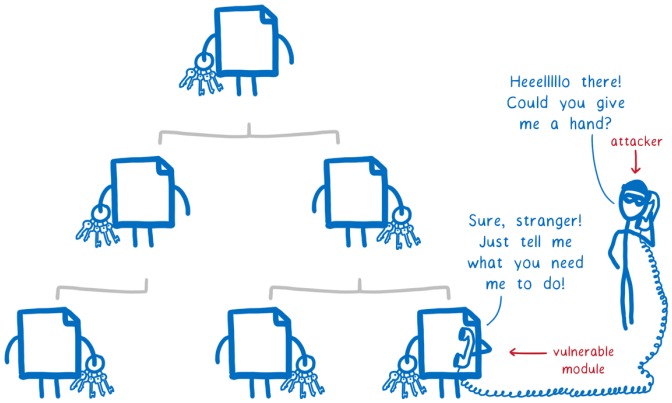
Malicious module



SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

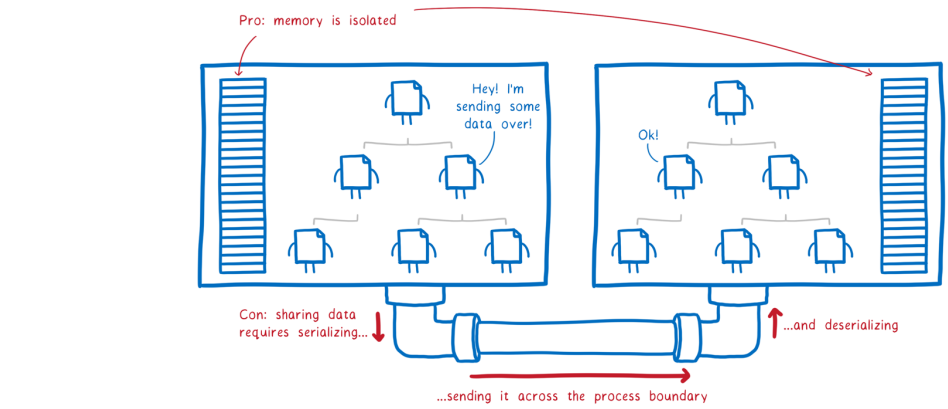
Vulnerable module



SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

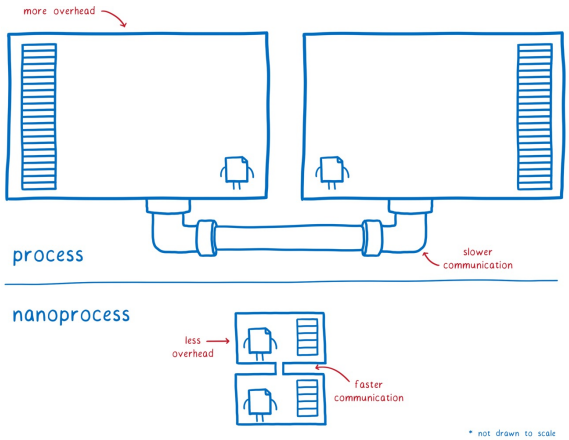
Process Isolation



SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

WebAssembly Nano-Process

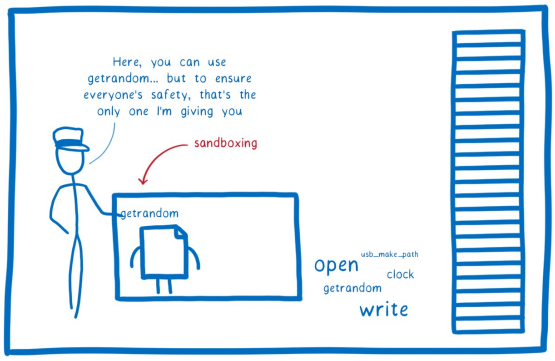


SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

WebAssembly Nano-Process

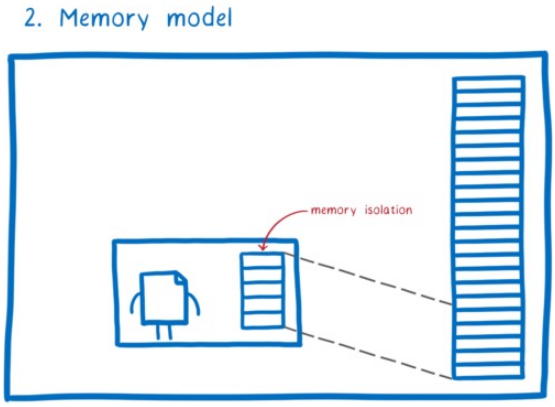
1. Sandboxing



SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

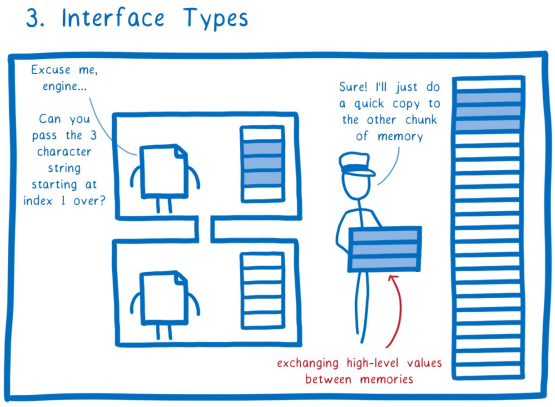
WebAssembly Nano-Process



SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

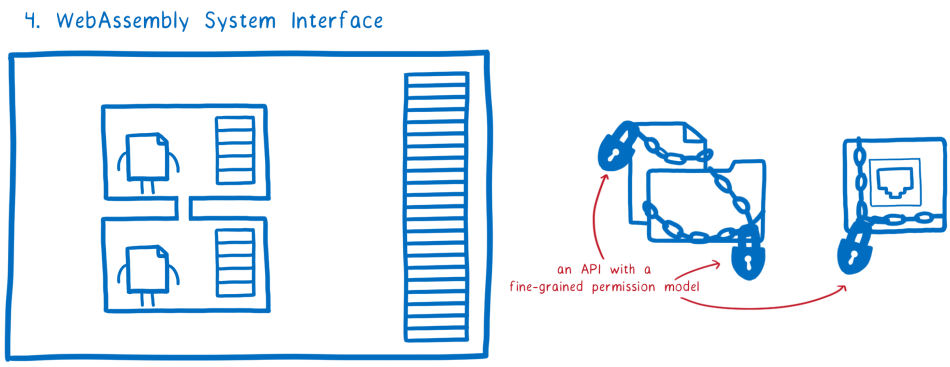
WebAssembly Nano-Process



SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

WebAssembly Nano-Process

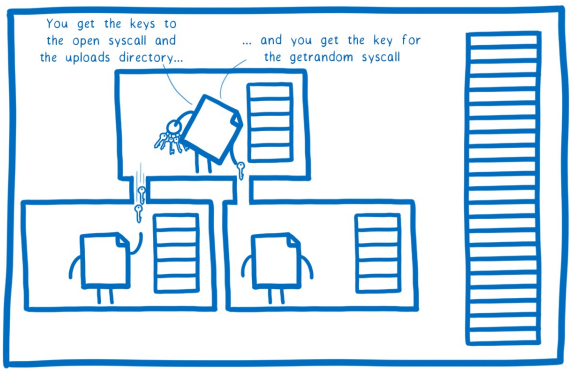


SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

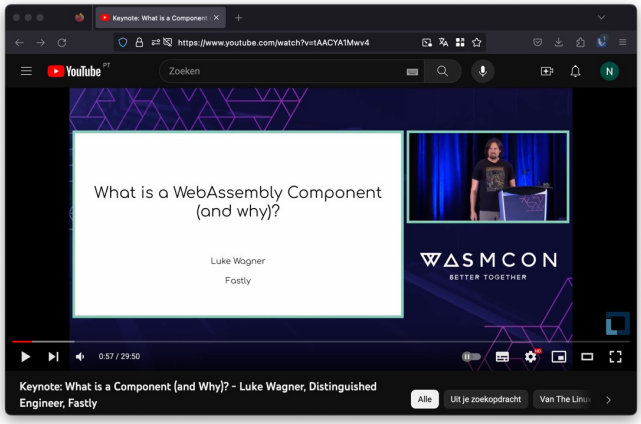
WebAssembly Nano-Process

5. The missing link



SecAppDev

WebAssembly Component Model

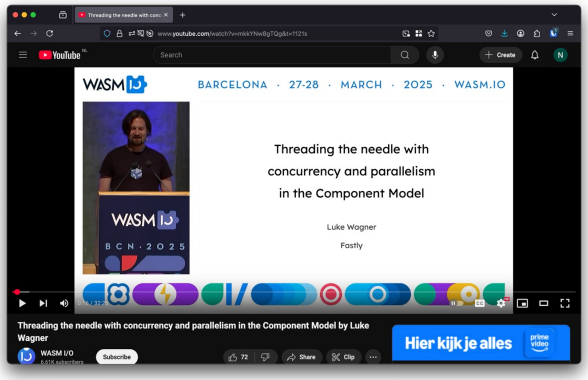


SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

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

WebAssembly Component Model

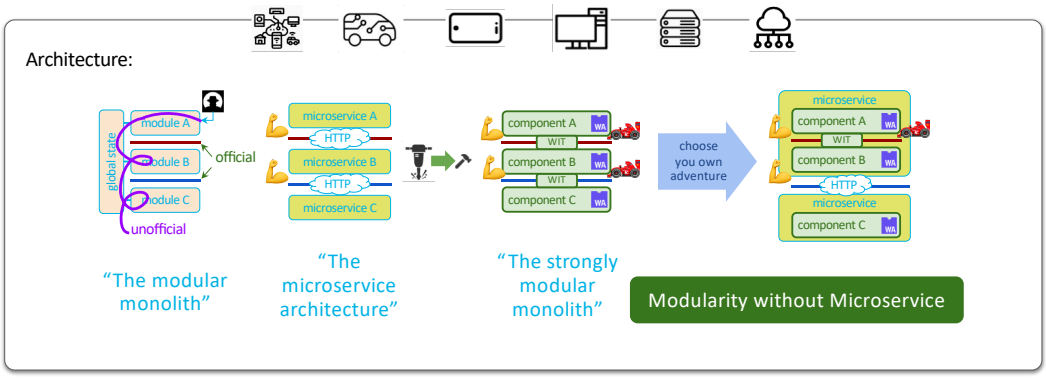


SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

https://www.youtube.com/watch?v=mkkYNw8gTQg&t=1121s

WASI 0.3

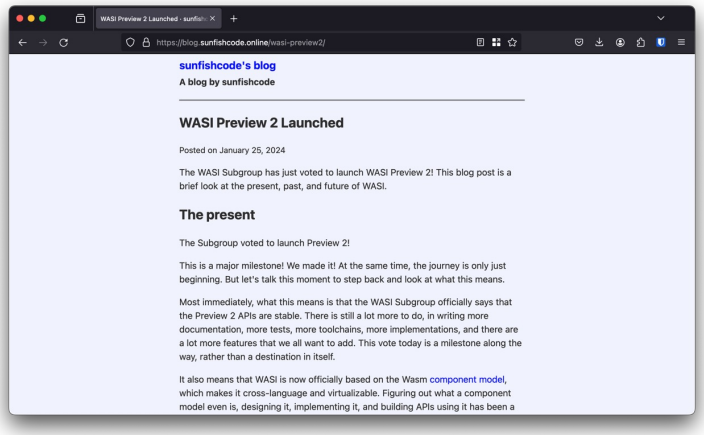


SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

https://www.youtube.com/watch?v=mkkYNw8gTQg&t=1121s

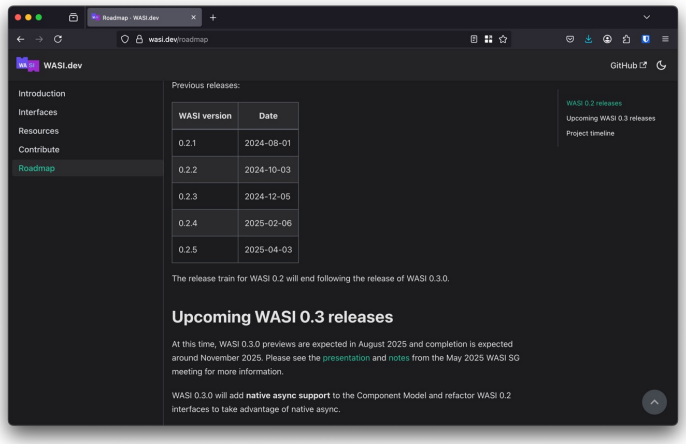
WASI Preview 2 (0.2)



SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

WASI 0.3

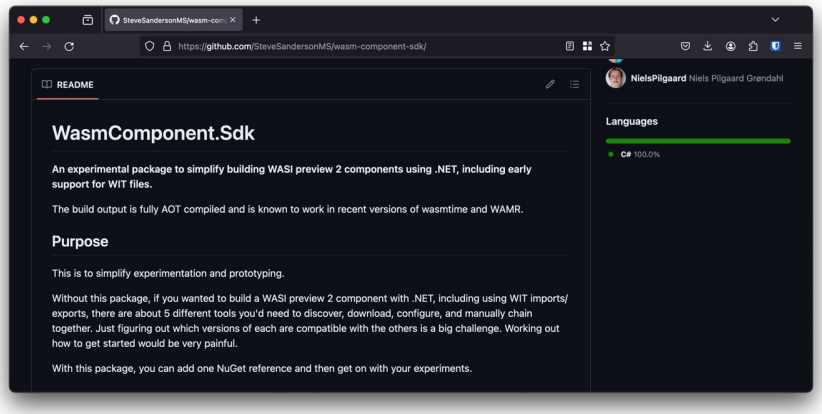


SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

https://www.youtube.com/watch?v=mkkYNw8gTQg&t=1121s

WasmComponent.SDK

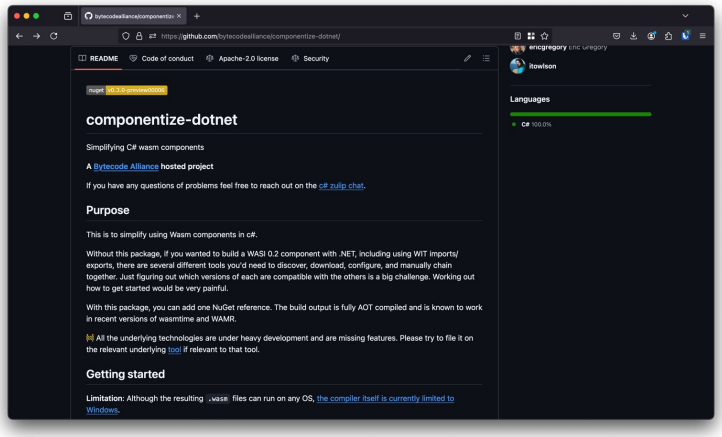


SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

https://github.com/SteveSandersonMS/wasm-component-sdk/

componentize-dotnet

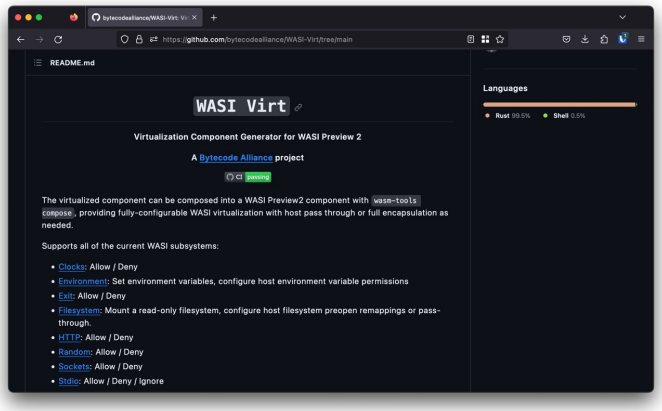


SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

https://github.com/bytecodealliance/componentize-dotnet/

WASI Virt



SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

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

Runtimes and Security

- Most security research published focusses on correctness of WASM runtimes/VM's

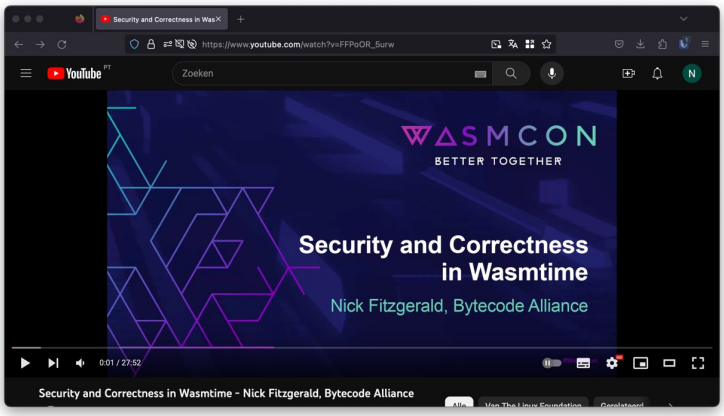
- Bytecode Alliance Blogpost:
  - "Security and Correctness in Wasmtime"
  - Written in Rust → Using all it's LangSec features
  - Continues Fuzzing & formal verification
  - Security process & vulnerability disclosure

SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

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

Runtimes and Security



SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

https://www.youtube.com/watch?v=FFPoOR_5urw

WebAssembly Lineair Memory

SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

https://www.usenix.org/conference/usenixsecurity20/presentation/lehmann

WebAssembly Lineair Memory

Securing Stack Smashing Protection in WebAssembly Applications

Quentin Michaud[1,2], Yohan Pipereau[2], Olivier Levillain[2], and Dhouha Ayed[1]

[1] Thales Group, Palaiseau, France
firstname.lastname@thalesgroup.com
[2] SAMOVAR, Télécom SudParis, Institut Polytechnique de Paris, Palaiseau, France
firstname.lastname@telecom-sudparis.eu

Abstract. WebAssembly is an instruction set architecture and binary format standard, designed for secure execution by an interpreter. Previous work has shown that WebAssembly is vulnerable to buffer overflow due to the lack of effective protection mechanisms.
In this paper, we evaluate the implementation of Stack Smashing Protection (SSP) in WebAssembly standalone runtimes, and uncover two weaknesses in their current implementation. The first one is the pos-

R] 23 Oct 2024

SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

https://arxiv.org/pdf/2410.17925v1

Conclusion

- 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 0.2 big milestone; tooling in progress!
- WASI 0.3 due in August 2025
- Cloud Native ❤️ WebAssembly

SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange

Merci! Bedankt! Thanks!

- https://github.com/nielstanis/secappdev25wasm
- ntanis at Veracode.com
- @nielstanis@infosec.exchange
- https://blog.fennec.dev

SecAppDev

@niels.fennec.dev @nielstanis@infosec.exchange