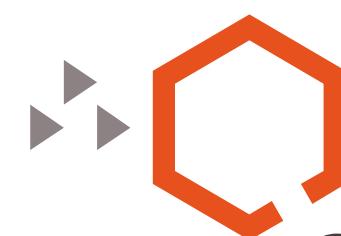


— 5 - 7 March 2019 —

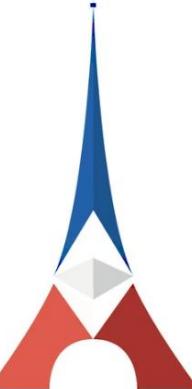
Let's dig inside Ethereum Smart Contracts compiled to WebAssembly

EthCC 2019

© QuoScient | ETHCC 2019



QuoScient





Whoami



Patrick Ventuzelo

@Pat_Ventuzelo



QuoScient GmbH

- ▶ (Blockchain) Security Researcher/Engineer



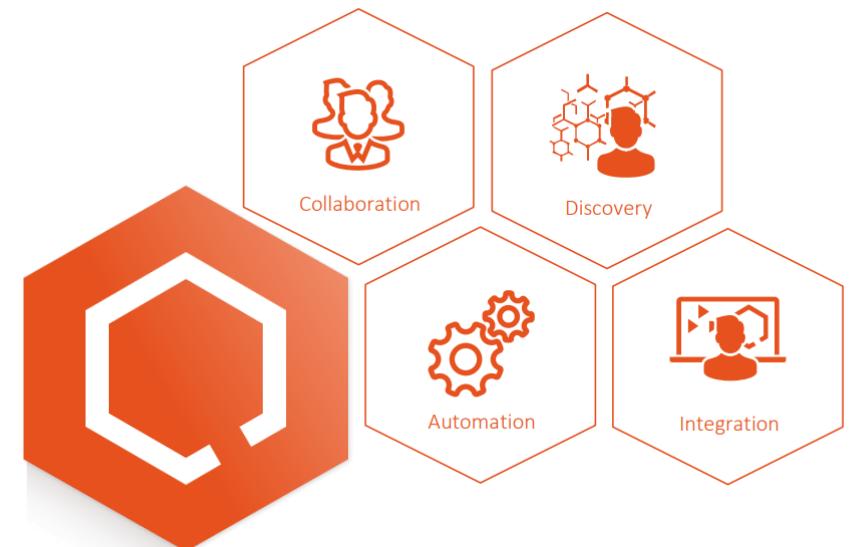
Quolab

- ▶ Threat Intel & Response Platform
- ▶ Collaborative, Decentralized

What's my relation with blockchains?



- ▶ Blockchain [Transaction Tracking](#)
- ▶ [Research](#) about Smart contracts, [WebAssembly](#), ...
- ▶ Vulnerability Analysis/Research
- ▶ Smart contract Audit (ETH, EOS, ...)
- ▶ Security tool Development ([Octopus](#), Quolab, ...)

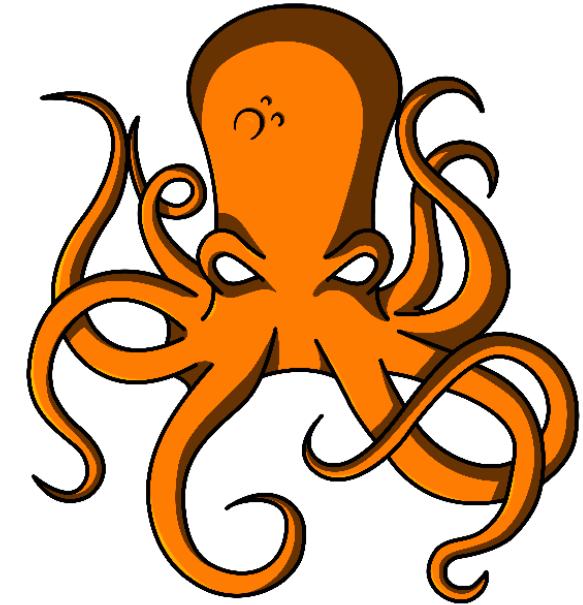




Octopus

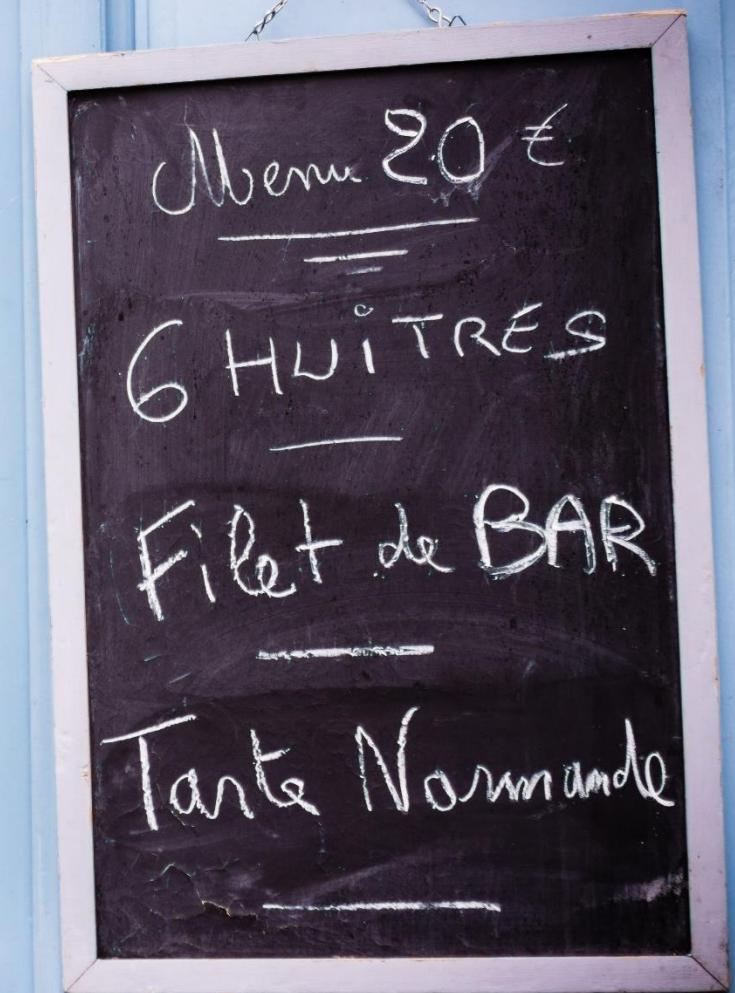
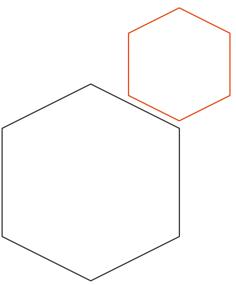
- Security analysis framework
 - ▶ WebAssembly module
 - ▶ Blockchain Smart Contracts (BTC/ETH/NEO/EOS)
- <https://github.com/quoscient/octopus>

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



◊ Agenda

1. Introduction
2. WebAssembly basics
 - ▶ Binary & text format
3. Program analysis
 - ▶ CFG, Call graph, decompilation
4. Parity Helloworld
 - ▶ Loader + runtime code analysis
5. WASM module Vulnerabilities
 - ▶ Integer overflow, ...
6. Conclusion



Introduction

01





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)



WEBASSEMBLY





Wasm for Blockchain smart contracts...

- 2 of the Top Cryptocurrencies by MarketCap

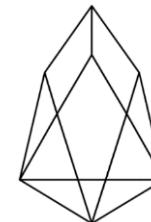
- Ethereum #2

- ▶ Decentralized platform that runs smart contracts
- ▶ (e)Wasm instead of EVM



- EOS #5

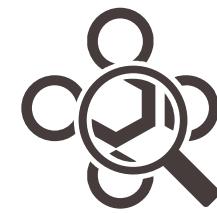
- ▶ Open source smart contract platform
- ▶ Compiled from C++ to WebAssembly



Cryptocurrencies	Exchanges	Watchlist	
#	Name	Market Cap	Price
1	Bitcoin	\$59,952,909,947	\$3,421.54
2	XRP	\$12,028,964,489	\$0.292184
3	Ethereum	\$10,863,935,796	\$103.72
4	EOS	\$2,109,946,677	\$2.33

WebAssembly basics

02





Source code to WebAssembly

C/C++

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

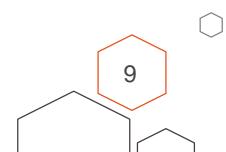
Rust

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



binary format (.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
```





Binary Format - overview

- Binary format
- Magic number: `\x00asm`
- Module structure
 - ▶ Header
 - ▶ 11 Sections + custom sections

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

WA WebAssembly Code Explorer

The screenshot shows a hex dump of WebAssembly code. The left column lists memory addresses from 0x00000000 to 0x00000070. The right column shows the corresponding assembly-like code. The assembly code includes instructions like `i32.const` and `i32.add`. The hex values are color-coded by byte value, with some bytes highlighted in yellow, purple, red, blue, and cyan.

Address	Hex Value	Assembly
0x00000000	00 61 73 6D 01 00 00 00 01 86 80 80 80 00 01 60	.asm.....`
0x00000010	01 7F 01 7F 03 82 80 80 80 00 01 00 04 84 80 80
0x00000020	80 00 01 70 00 00 05 83 80 80 80 00 01 00 01 06	...p.....
0x00000030	81 80 80 80 00 00 07 90 80 80 80 00 02 06 6D 65me
0x00000040	6D 6F 72 79 02 00 03 66 69 62 00 00 0A A7 80 80	mory...fib.....
0x00000050	80 00 01 A1 80 80 80 00 00 02 40 20 00 41 01 72@ .A.r
0x00000060	41 01 47 0D 00 20 00 0F 0B 20 00 41 7F 6A 10 00	A.G.A.j..
0x00000070	20 00 41 7E 6A 10 00 6A 0B	.A~j..j.

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



Source code to WebAssembly

C/C++

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

Rust

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



binary format (.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
```

text format (.wat)

```
(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)
          )
        )
      )
    )
  )
)
```



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
)
```



WebAssembly Instructions set

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
)
)
```



WebAssembly Instructions set

i32.add	i64.add	f32.add	f64.add	i32.wrap/i64	i32.load8_s	i32.store8
i32.sub	i64.sub	f32.sub	f64.sub	i32.trunc_s/f32	i32.load8_u	i32.store16
i32.mul	i64.mul	f32.mul	f64.mul	i32.trunc_s/f64	i32.load16_s	i32.store
i32.div_s	i64.div_s	f32.div	f64.div	i32.trunc_u/f32	i32.load16_u	i64.store8
i32.div_u	i64.div_u	f32.abs	f64.abs	i32.trunc_u/f64	i32.load	i64.store16
i32.rem_s	i64.rem_s	f32.neg	f64.neg	i32.reinterpret/f32	i64.load8_s	i64.store32
i32.rem_u	i64.rem_u	f32.copysign	f64.copysign	i64.extend_s/i32	i64.load8_u	i64.store
i32.and	i64.and	f32.ceil	f64.ceil	i64.extend_u/i32	i64.load16_s	f32.store
i32.or	i64.or	f32.floor	f64.floor	i64.trunc_s/f32	i64.load16_u	f64.store
i32.xor	i64.xor	f32.trunc	f64.trunc	i64.trunc_s/f64	i64.load32_s	
i32.shl	i64.shl	f32.nearest	f64.nearest	i64.trunc_u/f32	i64.load32_u	
i32.shr_u	i64.shr_u			i64.trunc_u/f64	i64.load	call
i32.shr_s	i64.shr_s	f32.sqrt	f64.sqrt	i64.reinterpret/f64	f32.load	call_indirect
i32.rotl	i64.rotl	f32.min	f64.min		f64.load	
i32.rotr	i64.rotr	f32.max	f64.max			
i32.clz	i64.clz				nop	grow_memory
i32.ctz	i64.ctz				block	current_memory
i32.popcnt	i64.popcnt			f32.demote/f64	loop	
i32.eqz	i64.eqz			f32.convert_s/i32	if	get_local
i32.eq	i64.eq			f32.convert_s/i64	else	set_local
i32.ne	i64.ne			f32.convert_u/i32	br	tee_local
i32.lt_s	i64.lt_s			f32.convert_u/i64	br_if	
i32.le_s	i64.le_s			f32.reinterpret/i32	br_table	get_global
i32.lt_u	i64.lt_u	f32.eq	f64.eq	f64.promote/f32	return	set_global
i32.le_u	i64.le_u	f32.ne	f64.ne	f64.convert_s/i32	end	
i32.gt_s	i64.gt_s	f32.lt	f64.lt	f64.convert_s/i64		i32.const
i32.ge_s	i64.ge_s	f32.le	f64.le	f64.convert_u/i32	drop	164.const
i32.gt_u	i64.gt_u	f32.gt	f64.gt	f64.convert_u/i64	select	f32.const
i32.ge_u	i64.ge_u	f32.ge	f64.ge	f64.reinterpret/i64	unreachable	f64.const

Program analysis

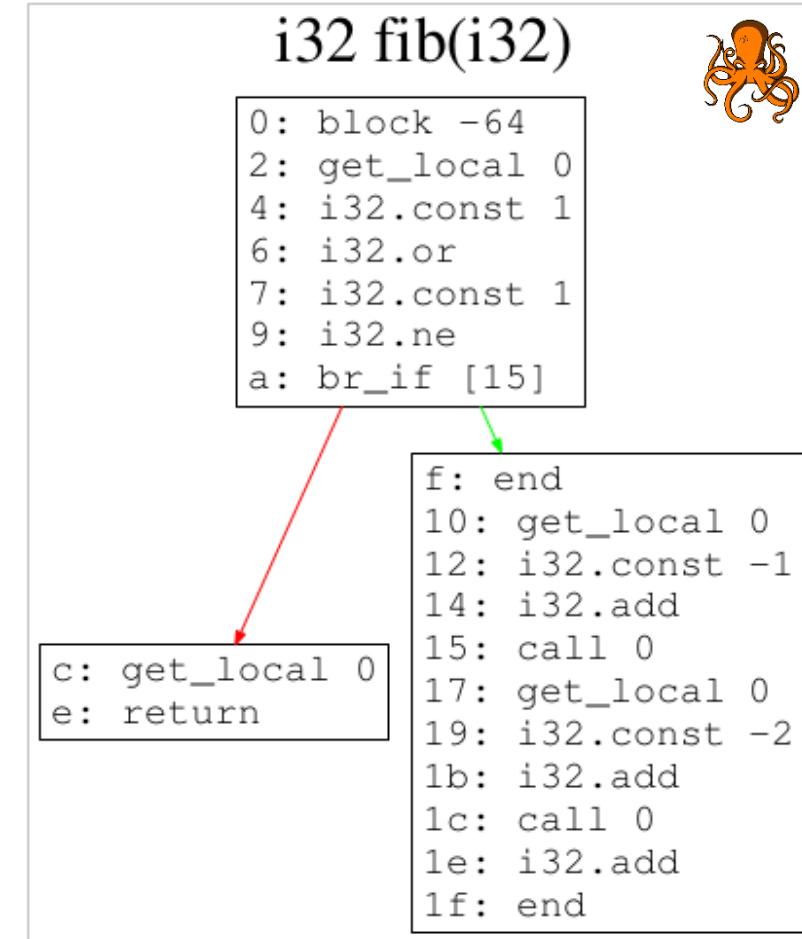
03





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
)
)
```





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
)
)
```



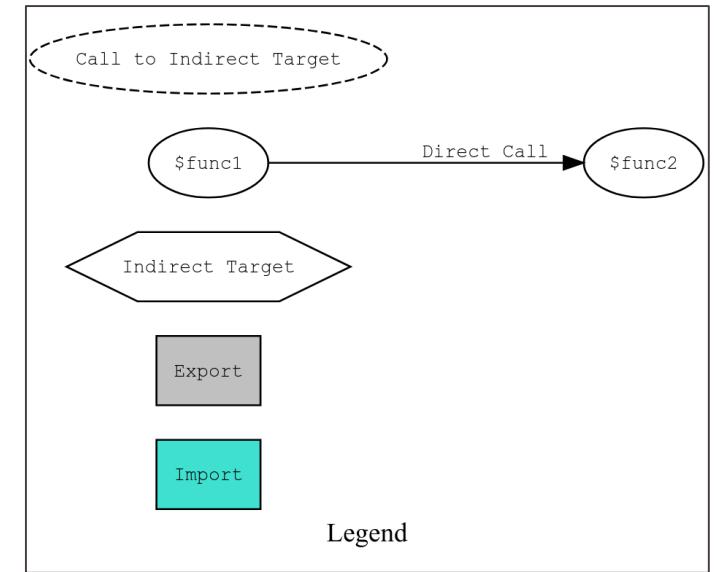
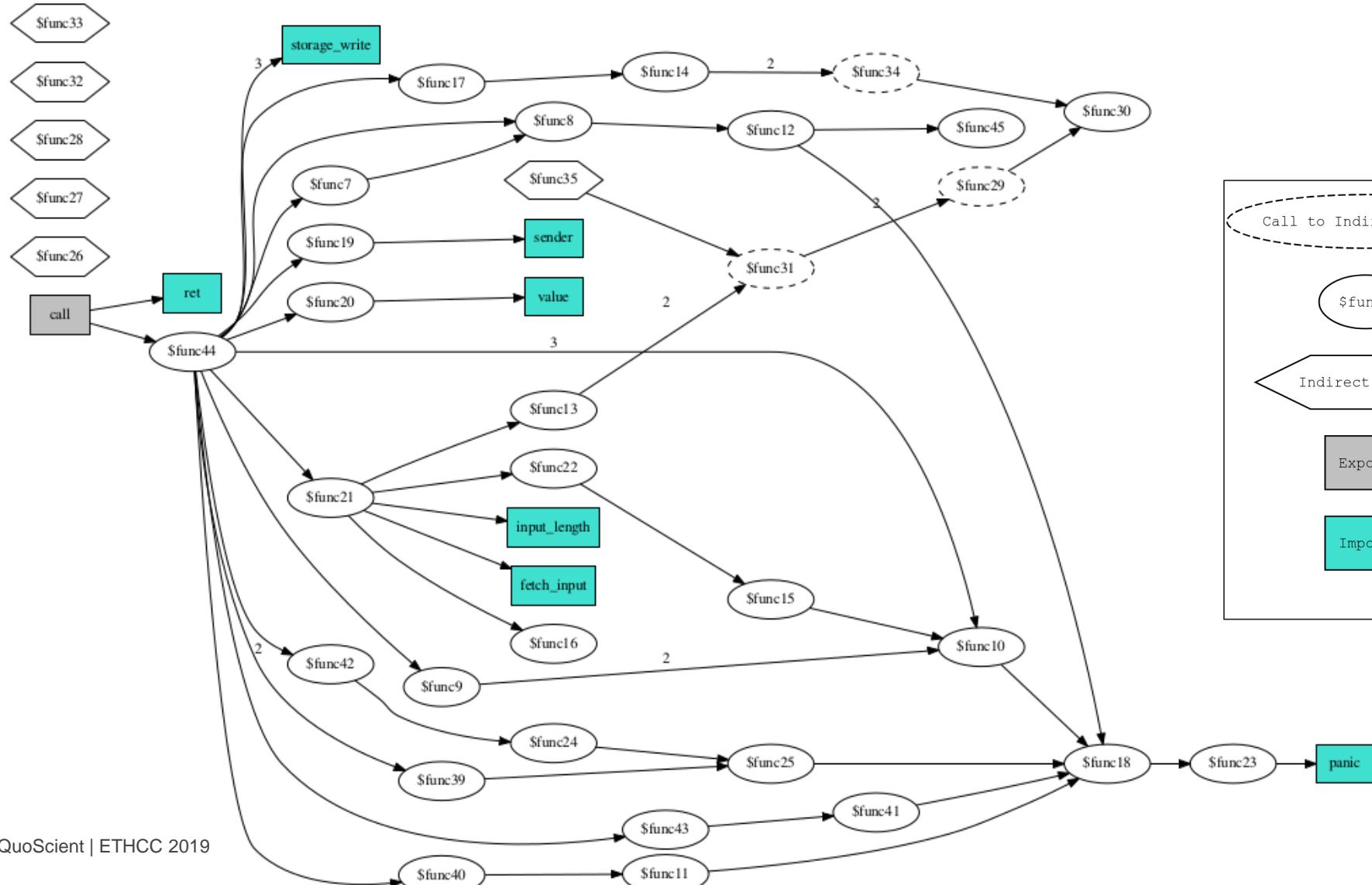
i32 fib(i32)

```
0: block -64
2: get_local 0
4: i32.const 1
6: i32.or
7: i32.const 1
9: i32.ne
a: br_if [15]

f: end
10: get_local 0
12: i32.const -1
14: i32.add
15: call 0
17: get_local 0
19: i32.const -2
1b: i32.add
1c: call 0
1e: i32.add
1f: end
```



Call Flow graph



WABT: WebAssembly Binary Toolkit

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

```
(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
  )
)
```

wat2wasm

```
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
```

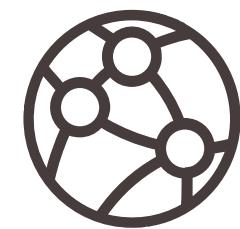
wasm2c

```
196 static void init_func_types(void) {
197   func_types[0] = wasm_rt_register_func
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 = lu;
214   i0 |= i1;
215   i1 = lu;
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
```

wasm2wat

Parity wasm Helloworld

04





Parity “Helloworld” WASM

Overview

Transaction Information

[This is a Kovan Testnet Transaction Only]

Block Height: 6601068 (1509256 block confirmations)

TimeStamp: 116 days 17 hrs ago (Mar-28-2018 03:00:24 PM +UTC)

From: 0x00a329c0648769a73afac7f9381e08fb43dbea7

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
002040501700101010708010463616c6c00020a110202000b0c001001418c0841e80110000b0b810202004180080b0b48656c6c6f2077
6f726c6400418c080be8010061736d0100000001090260000060027f7f00021a0203656e7603726574000103656e76066d656d6f72790
20102100303020000040501700101010501000601000708010463616c6c00010a120205001002000b0a00418008410b10000000b0b1201

View Input As ▾



Parity Technologies
@ParityTech

Follow

```
tomusdrw@ ~ $ http localhost:8545 jsonrpc=2.0 id=1 method=et  
3734b0e40e"}]  
HTTP/1.1 200 OK  
Content-Type: application/json  
Date: Wed, 28 Mar 2018 15:28:34 GMT  
Transfer-Encoding: chunked  
  
{  
    "id": 1,  
    "jsonrpc": "2.0",  
    "result": "0x48656c6c6f20776f726c64"  
}  
  
tomusdrw@ ~ $ echo $(hex2str 48656c6c6f20776f726c64)  
Hello world  
tomusdrw@ ~ $ exit
```

6:07 PM - 28 Mar 2018

59 Retweets 187 Likes

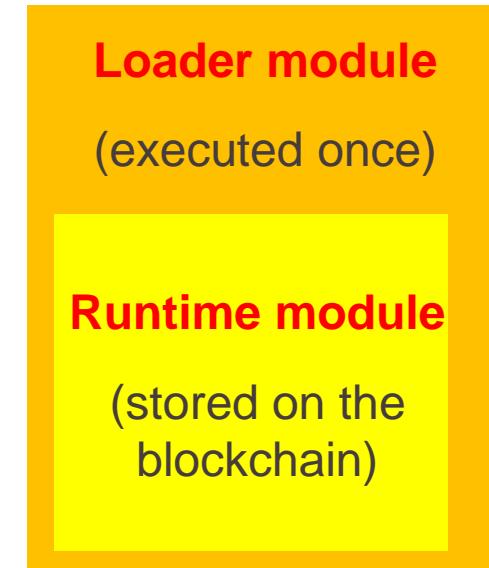


Q 6 17 59 ✓ 187



Parity “Helloworld” analysis

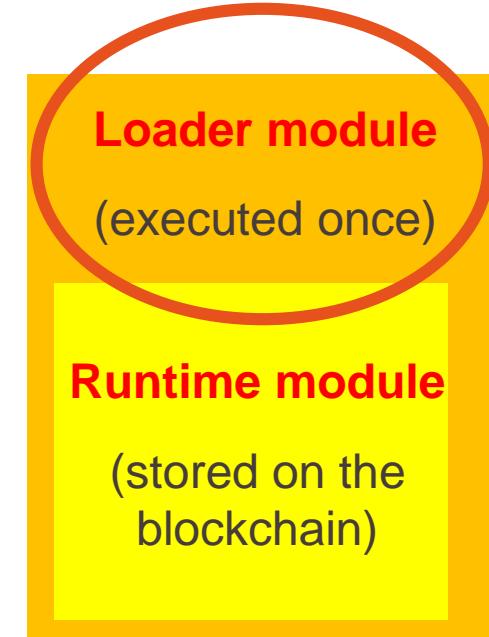
- ◊ Input data of the transaction that create the smart contract
 - ▶ Contain loader bytecode + embedded runtime code
 - ▶ runtime code stored on the blockchain during loader execution
 - ▶ Exactly the same way to create smart contract than with EVM





Parity “Helloworld” analysis

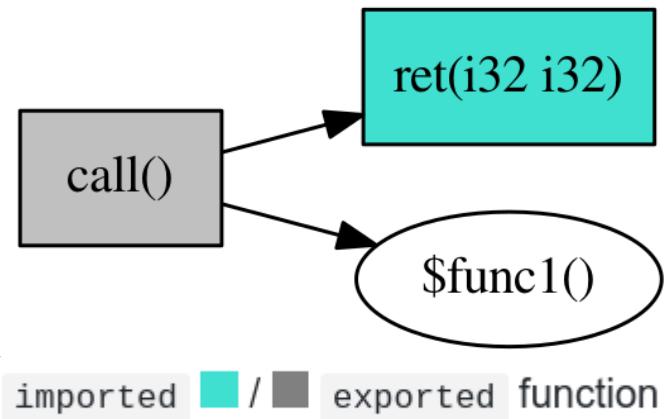
- ◊ Input data of the transaction that **create the smart contract**
 - ▶ Contain loader bytecode + embedded runtime code
 - ▶ **runtime code stored on the blockchain** during loader execution
 - ▶ Exactly the same way to create smart contract than with EVM



Loader module analysis

3 functions in the loader:

- ▶ call(): **exported**
- ▶ ret(i32 i32): **imported**
- ▶ \$func1(): local



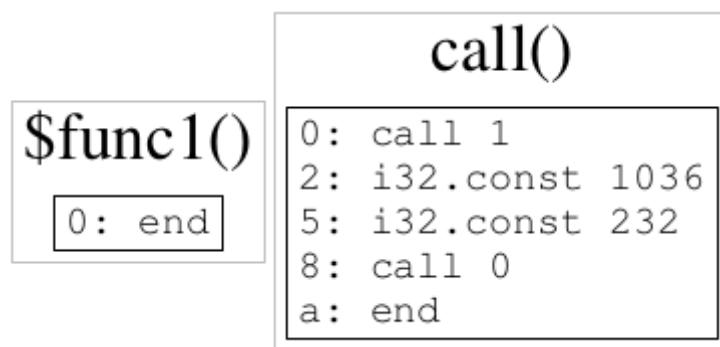
```
(module
  (type (;0;) (func))
  (type (;1;) (func (param i32 i32)))
  (type (;2;) (func))
  (import "env" "memory" (memory (;0;) 2 16))
  (import "env" "ret" (func (;0;) (type 1)))
  (func (;1;) (type 0))
  (func (;2;) (type 2)
    call 1
    i32.const 1036
    i32.const 232
    call 0)
  (table (;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"))
)
```



Loader module analysis

3 functions in the loader:

- ▶ call(): **exported**
- ▶ ret(i32 i32): **imported**
- ▶ \$func1(): local



```
(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")
```

call() pseudo-code:

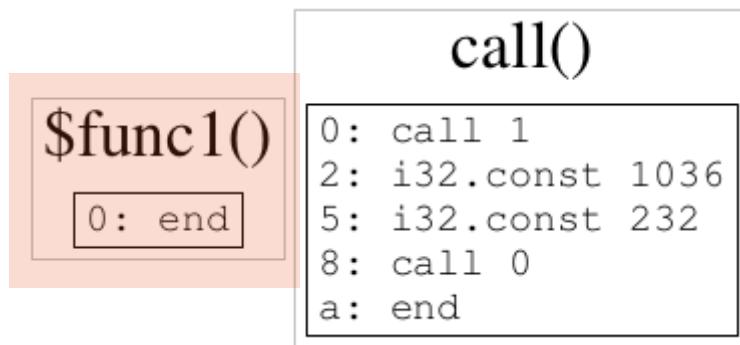
```
1 def call():
2     func1()
3     ret(i32.const 1036, i32.const 232)
```



Loader module analysis

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)
```

```
(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"))
)
```



Loader module analysis - Data

- Linear memory initialize with:



Runtime
module

```
(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")
```



Loader module analysis - Data

- 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
```

- call() return the runtime code & its size

Runtime module

```
(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)
    "\x00asm\x01\x00\x00\x01\x09\x02\x00\x00\x02\x7f"
    "\x7f\x00\x02\x1a\x02\x03env\x03ret\x00\x01\x03env\x06"
    "memory\x02\x01\x02\x10\x03\x03\x02\x00\x00\x04\x05\x01"
    "\x01\x01\x01\x05\x01\x00\x06\x01\x00\x07\x08\x01\x04call"
    "\x00\x01\x0a\x12\x02\x05\x00\x10\x02\x00\x0b\x0a\x00A"
    "\x80\x08A\x0b\x10\x00\x00\x0b\x0b\x12\x01\x00A\x80\x08"
    "\x0b\x0bHello world\x00\x0b\x07linking\x03\x01\x0b"
    "\x00f\x04name\x01\x06\x00\x03ret\x01\x05"
    "panic\x02\x04call\x03"
    "/_ZN14pwasm_ethereum3ext3ret17h604d8098d1686c80E"
    "\x04\x06deploy\x05\x11rust_begin_unwind")
```



Loader module analysis - Data

- Linear memory initialize with:



Not optimized...

Constant not used

- call() pseudo-code:

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

- call() return the runtime code & its size

```
(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"))
)
```

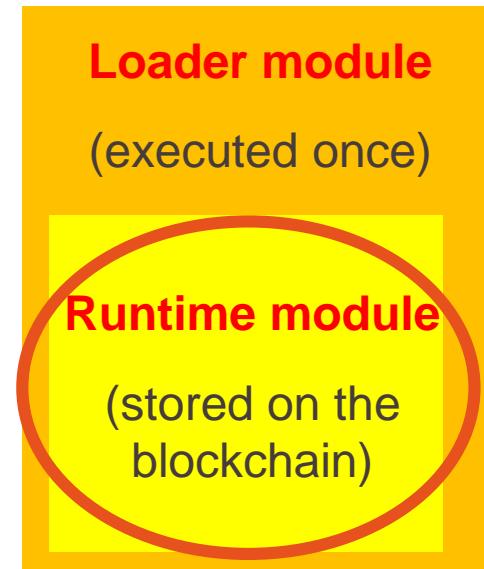


Parity “Helloworld” analysis

- ◊ 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

```
(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





Runtime module analysis

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



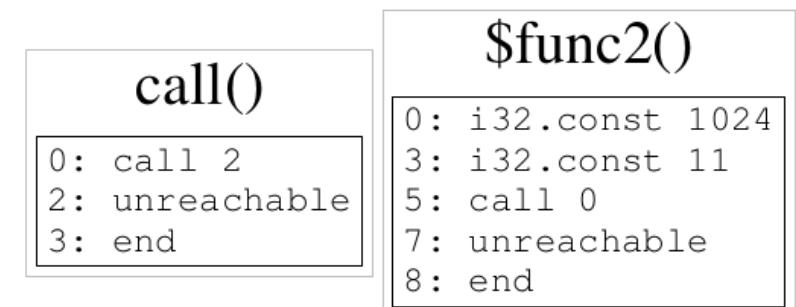
imported █ / █ exported function

- Pseudo code:

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

def call():
    func2()

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





Runtime module analysis

- In short:
 - ▶ Same **imported/exported** functions names
 - ▶ Data section contain “Hello world” string - offset 1024
- Pseudo code:

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

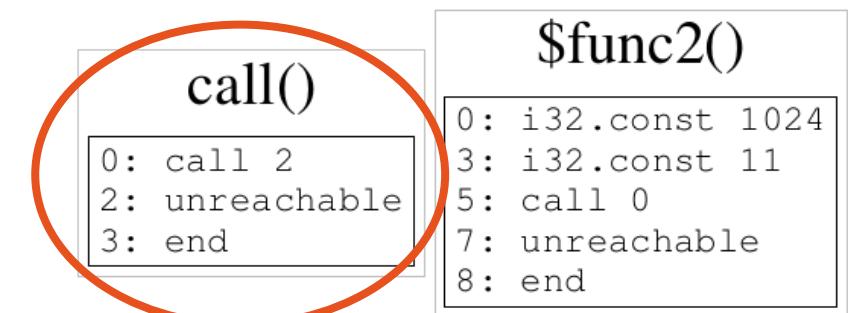
def call():
    func2()

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

Not optimized...
func2() should be call() instead



imported [cyan] / [grey] exported function





Runtime module analysis

↳ In short:

- ▶ Same **imported/exported** functions names
- ▶ Data section contain “Hello world” string - offset 1024



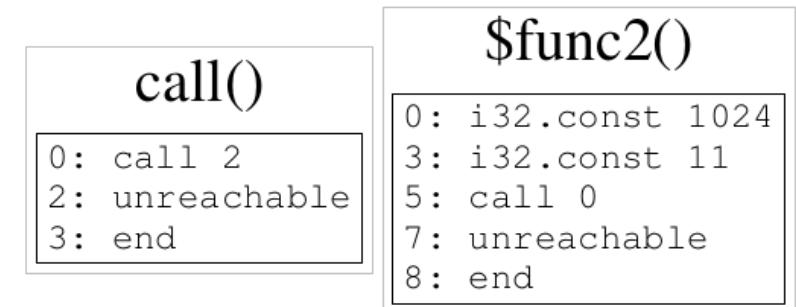
imported / exported function

↳ Pseudo code:

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

def call():
    func2()

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



↳ 2 customs section: “linking” & “name”

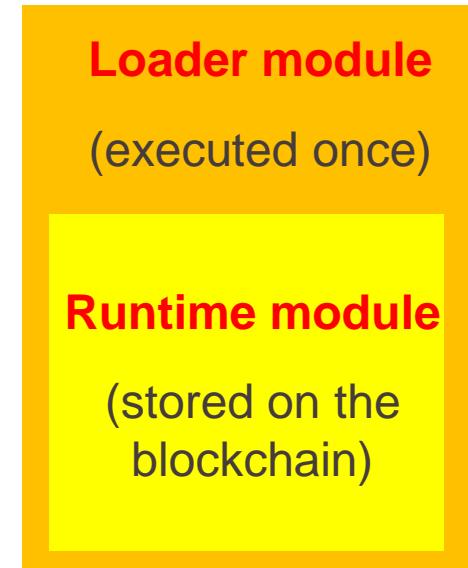
- ▶ “name” section contains debug strings

```
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')]
```



Parity “Helloworld” analysis - conclusion

- 2 stages WebAssembly modules (loader + payload)
- WebAssembly makes analysis of Ethereum smart contract easier
- This version is not optimized:
 - ▶ execution of **useless instructions/functions**
 - ▶ **unused data** in the loader code
 - ▶ **storing debug data** (Name section) in the runtime code
- All this non optimization can **COST MONEY** at execution
- Official tutorial if you want to try: [Writing smart contracts in Wasm for Kovan](#)





Parity is working on optimization

- ◊ I don't know if Parity team have **modified/optimized** the generation of this wasm module (loader/runtime code) since last year
- ◊ But in the interview of Jack Fransham, Parity Core Developer ([link](#))

What are you working on right now?

An optimising linear-time compiler for WebAssembly. We want to be able to produce native code for Wasm so we can get maximum performance, but traditional compilers are basically impossible to write without allowing so-called “compiler bombs” - pieces of code that you can send to the compiler that cause it to take an extremely long amount of time. I’m building a compiler that produces high-quality native code while also being immune to compiler bombs.



WASM module Vulnerabilities

05



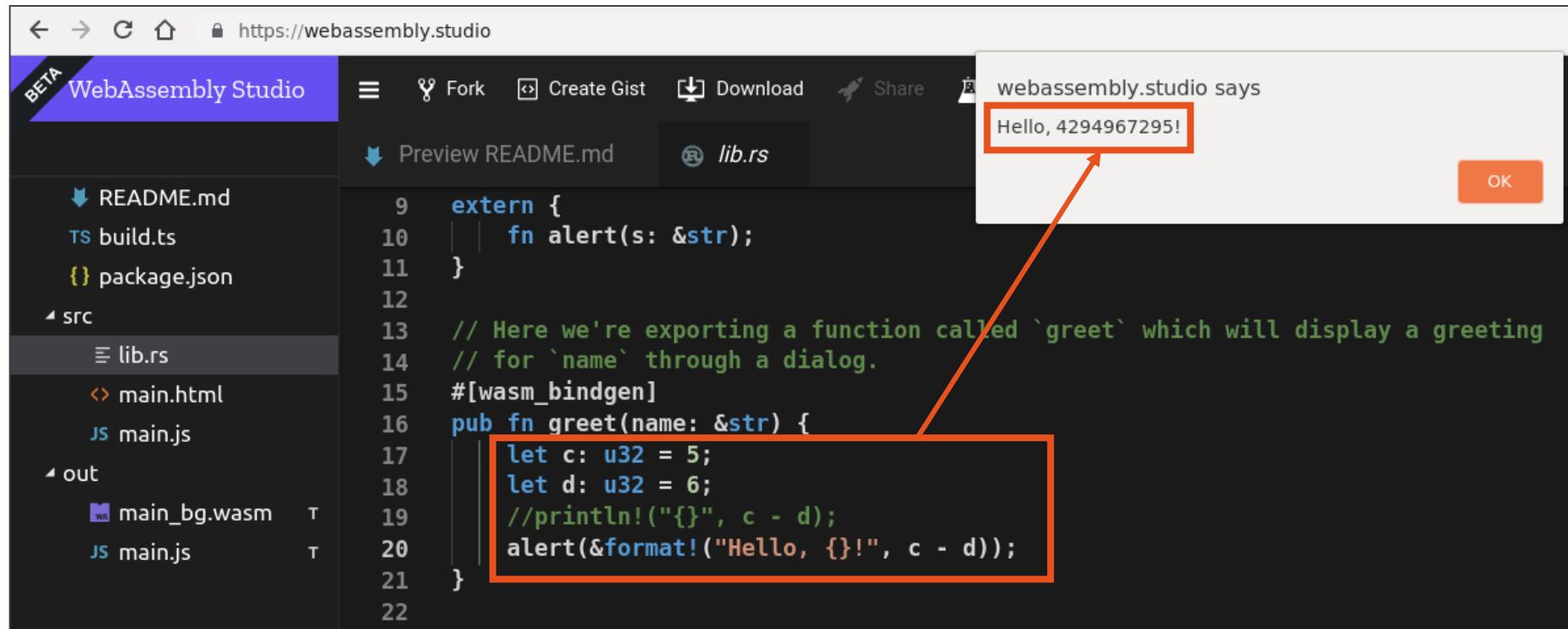
WebAssembly doesn't mean secured

- Old vulnerabilities classes can be inside
 - ▶ Buffer overflow
 - ▶ Integer overflow/underflows
 - ▶ Function pointer overwrite
 - ▶ Format string
- Advanced vulnerabilities classes as well
 - ▶ Use-After-Free (UaF)
 - ▶ Time Of Check to Time Of Use (TOCTOU)
 - ▶ ...
- Some links:
 - ▶ [Security Chasms of WASM](#)
 - ▶ [Hijacking the control flow of a WebAssembly program](#)



Simple Integer Overflow example

- Simple HelloWorld modified to have a runtime integer overflow
 - ▶ Written in **Rust** and compile to **wasm** (with `wasm_bindgen`)
 - ▶ Try by yourself on [WebAssembly Studio](https://webassembly.studio)



The screenshot shows the WebAssembly Studio interface. On the left, there's a sidebar with files: README.md, build.ts, package.json, SRC (lib.rs, main.html, main.js), and OUT (main_bg.wasm, main.js). The lib.rs file is open in the main editor area. A red box highlights the following Rust code:

```
9  extern {
10     fn alert(s: &str);
11 }
12
13 // Here we're exporting a function called `greet` which will display a greeting
14 // for `name` through a dialog.
15 #[wasm_bindgen]
16 pub fn greet(name: &str) {
17     let c: u32 = 5;
18     let d: u32 = 6;
19     //println!("{}", c - d);
20     alert(&format!("Hello, {}!", c - d));
21 }
```

An orange arrow points from this highlighted code to an alert dialog box titled "webassembly.studio says" containing the text "Hello, 4294967295!". An "OK" button is at the bottom right of the dialog.



Conclusion

06





Conclusion - Future

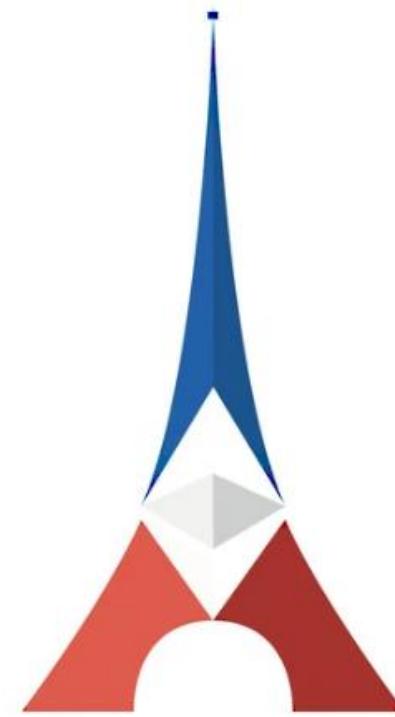
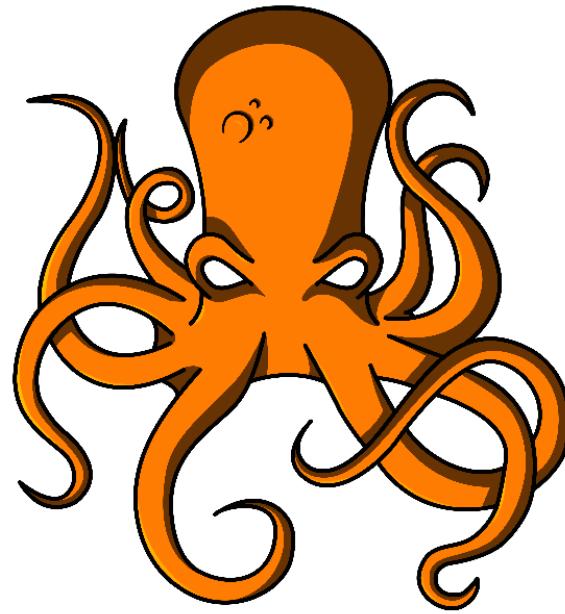
- Future of Ethereum WebAssembly module is promising
 - ▶ Ethereum WebAssembly VM ([wasmi](#), [hera](#))
 - ▶ ERC-20 wasm contract - [link](#)
 - ▶ Pre-compiled (e)Wasm module – [link](#)
 - ▶ Community !!!
- WebAssembly **WILL:**
 - ▶ Make analysis and debugging easier
 - ▶ Help non-blockchain people to enter in the game
- WebAssembly **WILL NOT:**
 - ▶ Secure poor code
 - ▶ Make



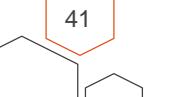
WEBASSEMBLY



Thanks & Questions



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



QuoScient

Radilostrasse 43

60489 Frankfurt

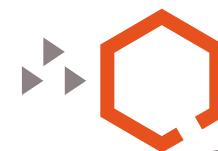
Germany

+49 69 33 99 79 38

curious@quoscient.io

www.quoscient.io

CONTACT



QuoScient

Digital Active Defense