WHATSABI

Andrey Petrov · @shazow

Let’s talk about...

Part 1: What is WhatsABI? 🤔

Part 2: How is WhatsABI different? 😮

... and why everyone else is doing it wrong!

Part 3: Using WhatsABI to visualize EVM bytecode 😍

Part 4: Future of WhatsABI 🤔

... what else can we solve along the way?

What is ABI?

Webster's Dictionary defines ‘ABI’ as **Acquired Brain Injury**, which is what happens when a programmer tries to interface with a smart contract.

@shazow noooo you frontran my brain injury jooooke lolol sokay

moiseiggy lol sorry

shazow <edits slide about moiseiggy frontrunning>

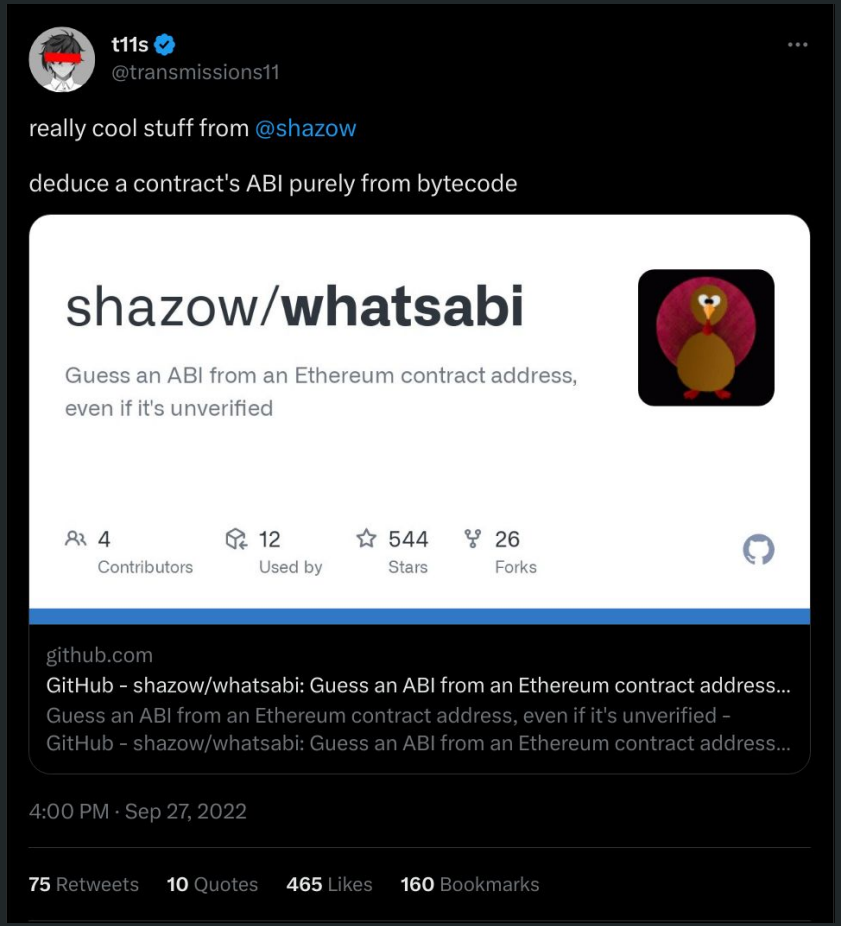
@shazow <edits slide about moiseiggy frontrunning>

moiseiggy

WhatsABI?

“deduce a contract’s ABI purely from bytecode”
- @transmissions11

#socialproof ✨



Goals of WhatsABI

- Extract metadata from **EVM bytecode** (starting with selectors)
- Use **static analysis**, but limit to **O(instructions)** with a small constant factor (ideally 1-2 passes, strictly avoid unbounded computation)
- **No assumptions about source code** (Solidity, Vyper, YUL, Huff, whatever)
- **Runnable in a browser**, embeddable in wallets (TypeScript/JavaScript)
- Permissive open source (**MIT**-licensed) to avoid duplicated effort

README.md

WhatsABI

Guess an ABI from an Ethereum contract address, even if it's unverified.

We started with parsing EVM bytecode to find 4-byte `JUMPI` instructions, but one thing led to another and now we're a bit more sophisticated.

We can also look up the 4-byte selectors on APIs like 4byte.directory to discover possible original function signatures.

Usage

```
import { ethers } from "ethers";
import { whatsabi } from "@shazow/whatsabi";
```

```
import { whatsabi } from "@shazow/whatsabi";

const provider = new ethers.getDefaultProvider(); // substitute with your fav provider
const address = "0x00000000006c3852cbEf3e08E8dF289169EdE581"; // Or your fav contract address
const code = await provider.getCode(address); // Load the bytecode

// Get just the callable selectors
const selectors = whatsabi.selectorsFromBytecode(code);
console.log(selectors); // -> ["0x06fdde03", "0x46423aa7", "0x55944a42", ...]

// Get an ABI-like list of interfaces
const abi = whatsabi.abiFromBytecode(code);
console.log(abi);
// -> [
//  {"type": "event", "hash": "0x721c20121297512b72821b97f5326877ea8ecf4bb9948fea5bfcb6453074d37f"},
//  {"type": "function", "payable": true, "selector": "0x06fdde03", ...},
//  {"type": "function", "payable": true, "selector": "0x46423aa7", ...},
//   ...

// We also have a suite of database loaders for convenience
const signatureLookup = new whatsabi.loaders.OpenChainSignatureLookup();
console.log(await signatureLookup.loadFunctions("0x06fdde03"));
// -> ["name()"]);
console.log(await signatureLookup.loadFunctions("0x46423aa7"));
// -> ["getOrderStatus(bytes32)"]);
```

```
// We also have a suite of database loaders for convenience
const signatureLookup = new whatsabi.loaders.OpenChainSignatureLookup();
console.log(await signatureLookup.loadFunctions("0x06fdde03"));
// -> ["name()"]);
console.log(await signatureLookup.loadFunctions("0x46423aa7"));
// -> ["getOrderStatus(bytes32)"]);
```

… used by

- gnosis/zodiac
- ethcmd.com
- notar-cli
- abi.w1nt3r.xyz
- ondora.xyz
- monobase.xyz
- MEV searchers?
- **… security researchers?!**

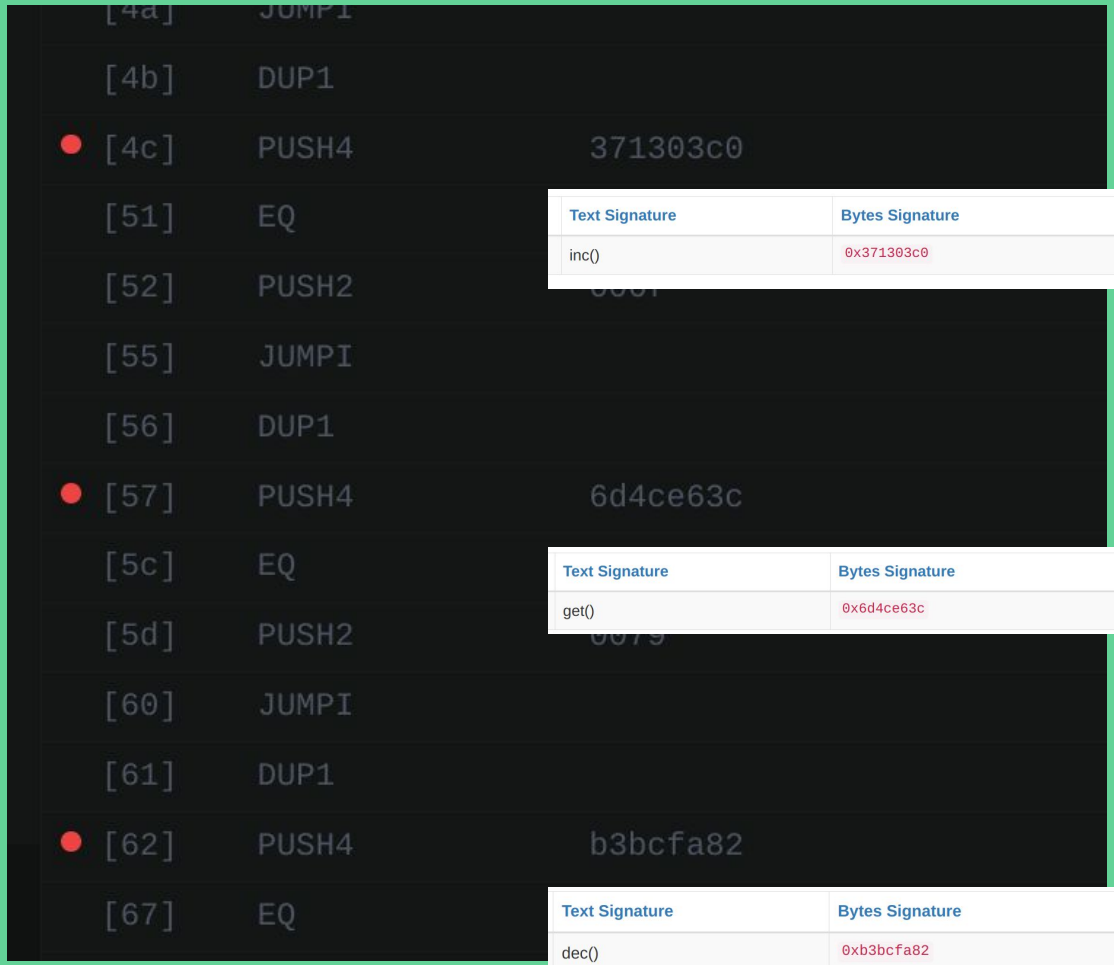
EVM Basics: Selector Jump Table

Pseudocode

```
CALLDATA := CALLDATALOAD()

SELECTOR := CALLDATA << (256 - 4 * 8)   # LAST 4 BYTES

IF SELECTOR == 0x371303c0:

    GOTO 0x006f

IF SELECTOR == 0x6d4ce63c:

    GOTO 0x0079

...
```

EVM Playground ARROWGLACIER

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.3;

contract Counter {
    uint public count;

    // Function to get the current count
    function get() public view returns (uint) {
        return count;
    }

    // Function to increment count by 1
    function inc() public {
        count += 1;
    }

    // Function to decrement count by 1
    function dec() public {
        count -= 1;
    }
}
```



Text Signature	Bytes Signature
inc()	0x371303c0

Text Signature	Bytes Signature
get()	0x6d4ce63c

Text Signature	Bytes Signature
dec()	0xb3bcfa82

Parsing EVM Jump Tables

Get all the PUSH4’s

Parsing EVM Jump Tables

Get all the PUSH4's

Match against 4byte databases

Prune the garbage?

Examples:

- OpenZeppelin/contract-bot-gang
- Jon-Becker/heimdall-rs

```
40    var mapping: Map<string, boolean> = new Map();
41
42    for (var i = 0; i < disassembled.length; i++) {
43      var mnemonic = disassembled[i].opcode.mnemonic;
44      if (
45        (mnemonic == "PUSH4" || mnemonic == "PUSH32") &&
46        !mapping.has(disassembled[i].pushValue as string)
47      ) {
48        mapping.set(disassembled[i].pushValue as string, true);
49        if (mnemonic == "PUSH4")
50          byte4DirectoryFunctions.push(disassembled[i].pushValue as string);
51        else byte4DirectoryEvents.push(disassembled[i].pushValue as string);
52      }
53    }
54
55    return {
56      byte4DirectoryFunctions
```

contract-bot-gang

```rust
// find all function selectors in the given EVM.
pub fn find_function_selectors(assembly: String) -> Vec<String> {
    let mut function_selectors = Vec::new();

    // search through assembly for PUSH4 instructions, optimistically assuming that they are functio
    let assembly: Vec<String> = assembly
        .split('\n')
        .map(|line| line.trim().to_string())
        .collect();
    for line in assembly.iter() {
        let instruction_args: Vec<String> = line.split(' ').map(|arg| arg.to_string()).collect();

        if instruction_args.len() >= 2 {
            let instruction = instruction_args[1].clone();

            if instruction == "PUSH4" {
                let function_selector = instruction_args[2].clone();
                function_selectors.push(function_selector);
            }
        }
    }
    function_selectors.sort();
    function_selectors.dedup();
    function_selectors
```

heimdall-rs

Parsing EVM Jump Tables

Get all compared `PUSH4`'s linked to a `JUMPI`?

Examples:

- WhatsABI v0.1
- hananbeer/1regex4bytes

```
63([0-9a-f]{8})1461([0-9a-f]{4})57
PUSH4 {8 hex} EQ PUSH2 {4 hex} JUMPI
```

Parsing EVM Jump Tables: Gotchas

- Okay, but what about jumps that are outside of the dispatch table?
  - How soon can we stop? Will we miss anything else?
  - Jump Trees! (E.g. Uniswap)
- What if the selector is `0x00abcdef`?
  - We get a **`PUSH3`** instead of **`PUSH4`**
- What if the selector is `0x00000000`?

```
AaANwg8((address,address,address,uint136,uint40,uint40,uint24,uint8,uint256,bytes
MonkahmmXXXXXXXXXXXXACSKFIOY()
QKJ0gRzrmgZzBLWm(address,uint256,uint256,bytes)
ROOT4146650865()
abcei51243fdgjkh(bytes)
arb_ybtltp(uint256[])
bdspwouamqsxyabc(uint256,bytes)
blockHashAddendsInexpansible(uint256)
blockHashAmarilloNonspontaneously(uint256)
blockHashAmphithyronVersify(uint256)
blockHashAskewLimitary(uint256)
buyAndFree22457070633(uint256)
buy_bca86a0f(address,uint256,int256)
cehbdjakgfi(address,address,uint256,uint8)
contrivedNameThatisVeryUnlikelyToBeFoundInTheWild_visd4o(address,uint256)
execute_44g58pv()
f09140466846285922(address,bytes)
f7836435186477227(address)
fulfillBasicOrder_efficient_6GL6yc((address,uint256,uint256,address,address,addre
fulfillBasicOrder_efficient_fGEoT((((uint8,address,uint256,uint256,uint256)[],(ui
get_block_hash_257335279069929(uint256)
jIUTh(bytes)
left_branch_block(uint32)
mev_abcd1g3ekj2f4ih5(bytes)
mint_d22vi9okr4w(address)
mint_efficient_1268F998()
overdiffusingness(bytes,uint256,uint256,uint256,uint256)
randallAteMySandwich(uint256[],address[],uint8,uint256[],bool,bytes32,address,add
randallAteMySandwich_atrxxnf(bytes)
randallAteMySandwich_bixiwot(uint256,uint256)
```

Parsing EVM Jump Tables: Gotchas

- Okay, but what about jumps that are outside of the dispatch table?
  - How soon can we stop? Will we miss anything else?
  - Jump Trees! (E.g. Uniswap)
- What if the selector is `0x00abcdef`?
  - We get a **`PUSH3`** instead of **`PUSH4`**
- What if the selector is `0x00000000`?
  - We get an **`ISZERO`** instead of **`EQ`**
- Can the Program Counter be small enough for PUSH1? Or big enough for a PUSH3?
- What if the selector is not 4 bytes?
- Init code?

What does WhatsABI do today?

WhatsABI v0.4.1:

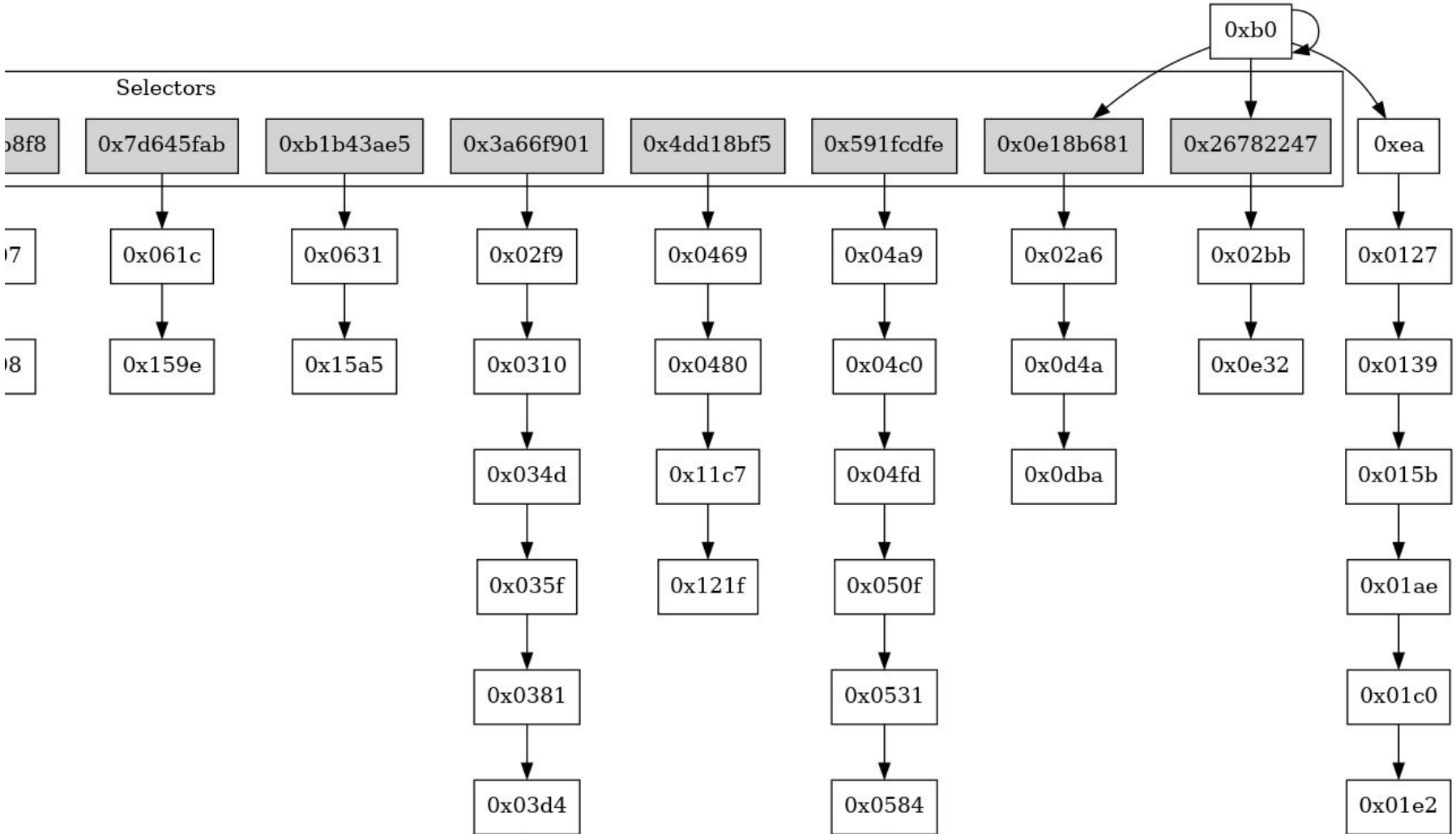
- `PUSHN <SELECTOR> EQ PUSHN <OFFSET> JUMPI`
- `PUSHN <SELECTOR> DUP2 EQ PUSHN <OFFSET> JUMPI`
- `ISZERO PUSHN <OFFSET> JUMPI`
- `PUSHN <SELECTOR> GT/LT PUSHN <OFFSET> JUMPI`
- Track jump windows (almost like function scopes)
- Multi-window dispatch tables (jump trees and jump chains)
- Non-standard selectors
- Annotation/analysis of jump windows
- v0.5 will detect static init code (program within a program)

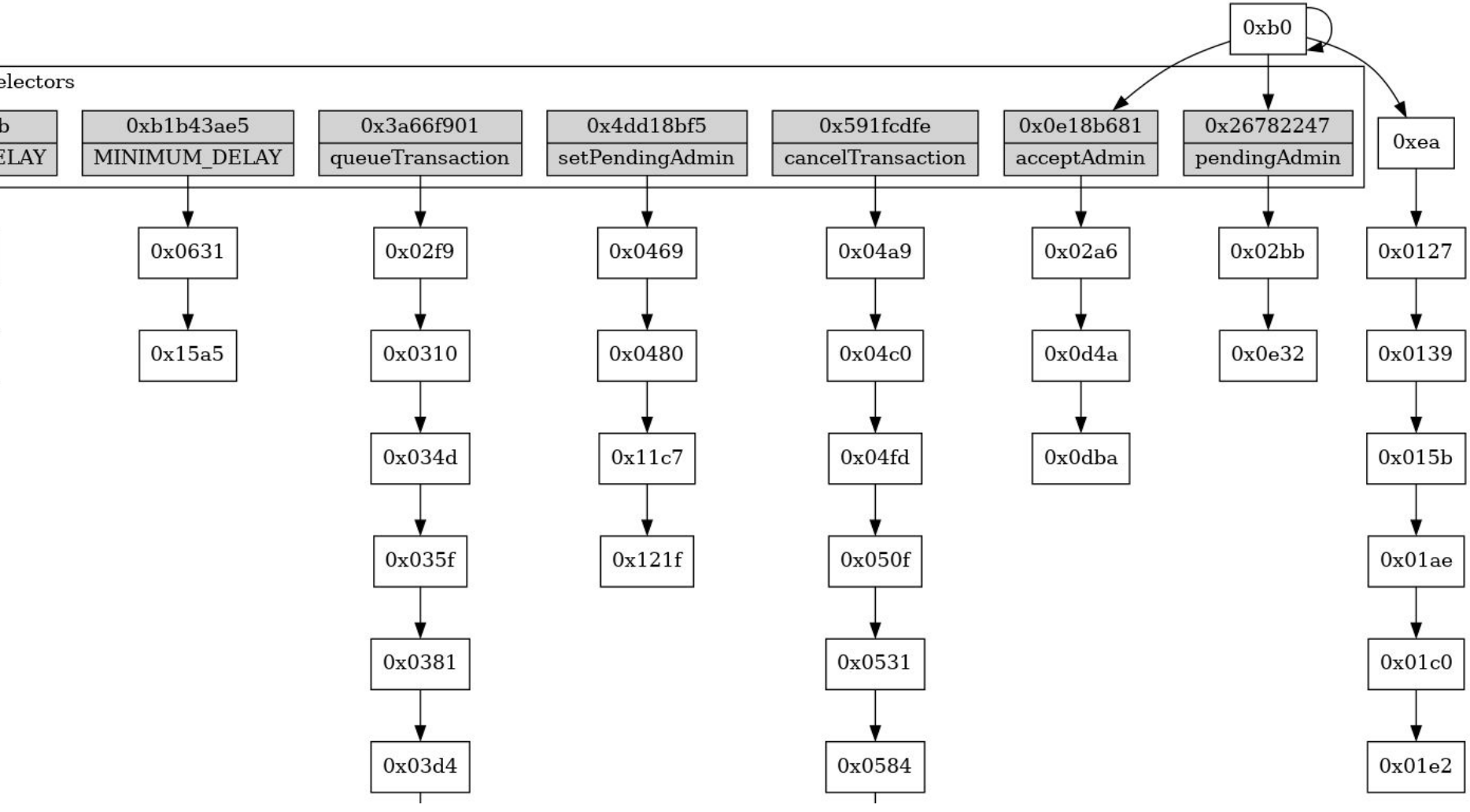
WhatsABI’s secret debugging tools

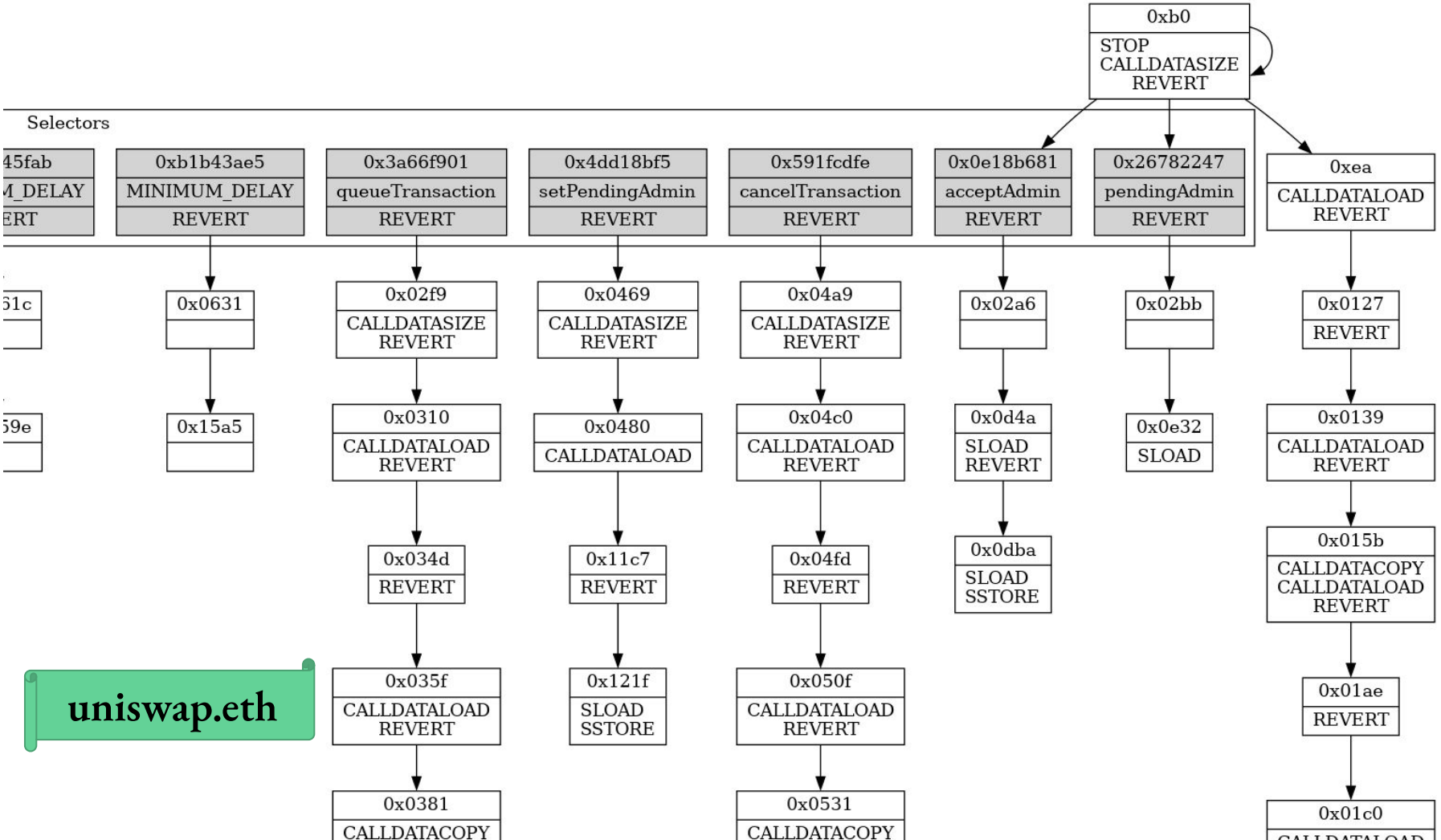
```
$ whatsabi/src.ts/bin/dot.ts uniswap.eth
```

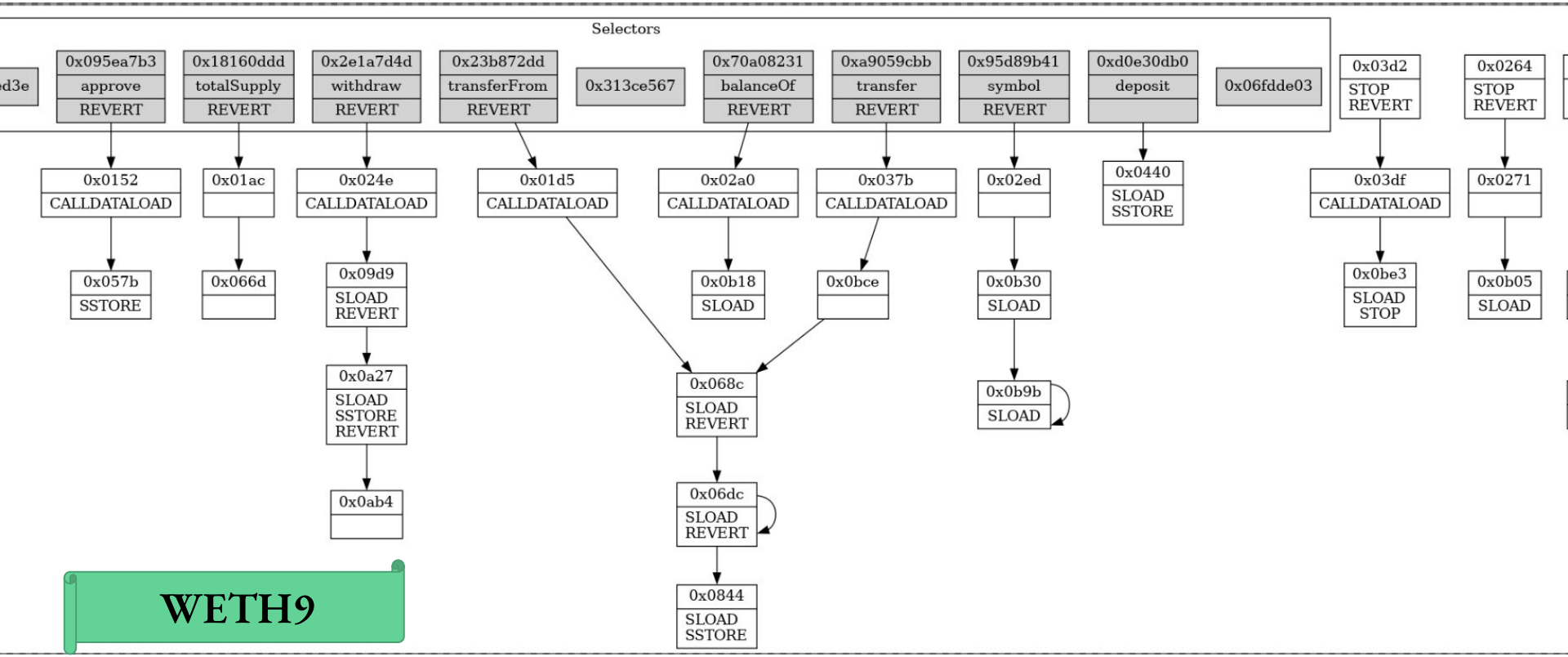
…

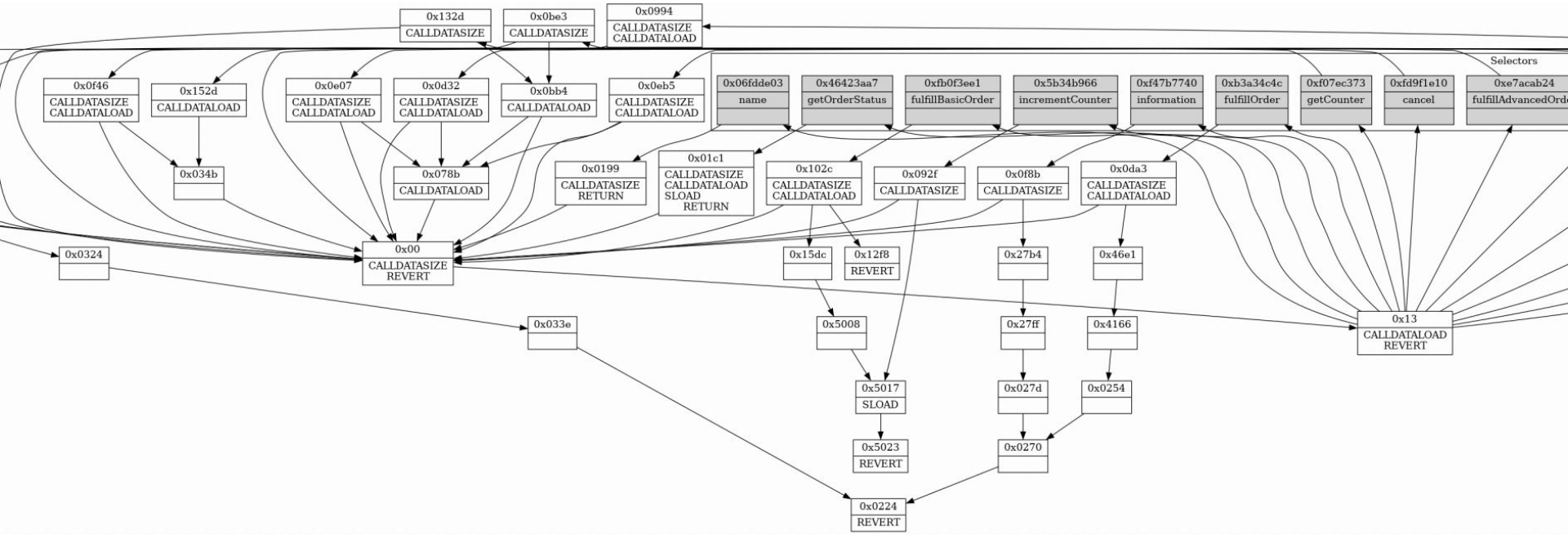
```
 ~/projects/whatsabi $ src.ts/bin/dot.ts 0x1a9C8182C09F50C8318d769245beA52c32BE35BC
withCache: Using cached value: .cache/0x1a9C8182C09F50C8318d769245beA52c32BE35BC_abi
digraph JUMPS {
        node [shape=record];

        subgraph cluster_0 {
                label = Selectors;
                node [style=filled];
                rankdir=LR;
                "0xc1a287e2" "0xe177246e" "0xf2b06537" "0xf851a440" "0x6a42b8f8" "0x7d645fab" "0xb1b43a
        }
        "0x26782247" [label="{ 0x26782247 | { REVERT } }"]
        "0x26782247" -> { "0x02bb" }
        "0x02bb" [label="{ 0x02bb | {  } }"]
        "0x02bb" -> { "0x0e32" }
        "0x0e32" [label="{ 0x0e32 | { SLOAD } }"]
        "0x0e32" -> {  }
        "0x0e18b681" [label="{ 0x0e18b681 | { REVERT } }"]
        "0x0e18b681" -> { "0x02a6" }
        "0x02a6" [label="{ 0x02a6 | {  } }"]
        "0x02a6" -> { "0x0d4a" }
        "0x0d4a" [label="{ 0x0d4a | { SLOAD|REVERT } }"]
        "0x0d4a" -> { "0x0dba" }
        "0x0dba" [label="{ 0x0dba | { SLOAD|SSTORE } }"]
        "0x0dba" -> {  }
        "0xb0" [label="{ 0xb0 | { STOP|CALLDATASIZE|REVERT } }"]
        "0xb0" -> { "0xb0" "0xb0" "0x0e18b681" "0x0e18b681" "0x26782247" "0x26782247" "0xea" "0xea" }
        "0xea" [label="{ 0xea | { CALLDATALOAD|REVERT } }"]
        "0xea" -> { "0x0127" }
        "0x0127" [label="{ 0x0127 | { REVERT } }"]
```


0xb0
Selectors
b8f8
0x7d645fab
0xb1b43ae5
0x3a66f901
0x4dd18bf5
0x591fcdfe
0x0e18b681
0x26782247
0xea
07
0x061c
0x0631
0x02f9
0x0469
0x04a9
0x02a6
0x02bb
0x0127
08
0x159e
0x15a5
0x0310
0x0480
0x04c0
0x0d4a
0x0e32
0x0139
0x034d
0x11c7
0x04fd
0x0dba
0x015b
0x035f
0x121f
0x050f
0x01ae
0x0381
0x0531
0x01c0
0x03d4
0x0584
0x01e2

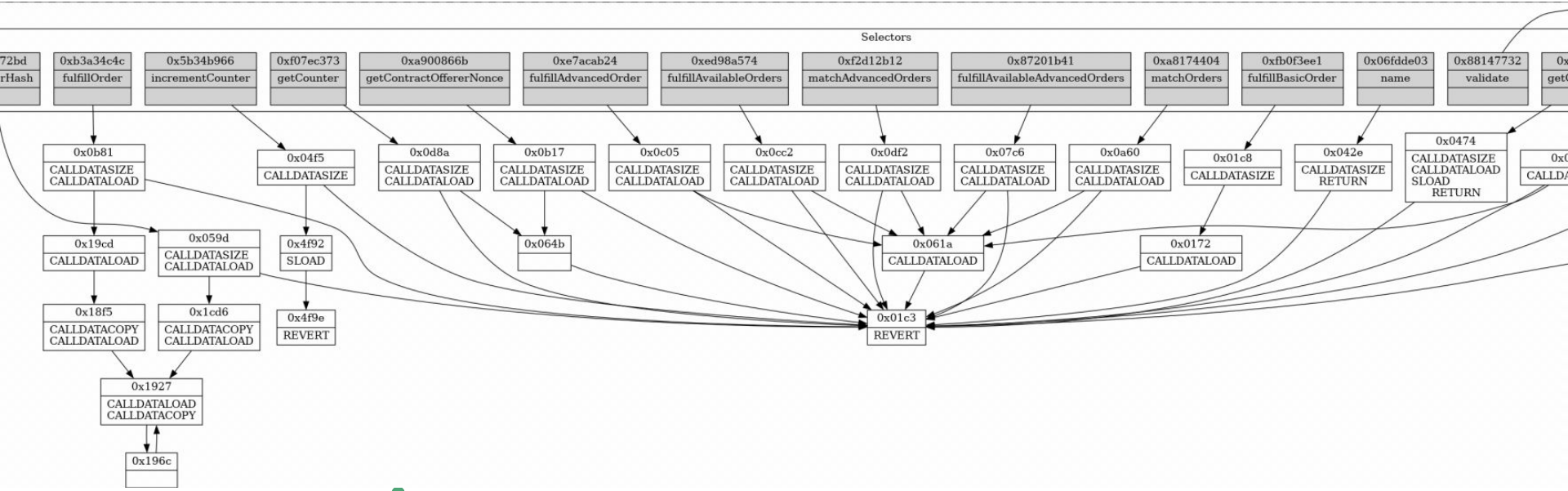

0xb0
electors
b
ELAY
0xb1b43ae5
MINIMUM_DELAY
0x3a66f901
queueTransaction
0x4dd18bf5
setPendingAdmin
0x591fcdfe
cancelTransaction
0x0e18b681
acceptAdmin
0x26782247
pendingAdmin
0xea
0x0631
0x15a5
0x02f9
0x0310
0x034d
0x035f
0x0381
0x03d4
0x0469
0x0480
0x11c7
0x121f
0x04a9
0x04c0
0x04fd
0x050f
0x0531
0x0584
0x02a6
0x0d4a
0x0dba
0x02bb
0x0e32
0x0127
0x0139
0x015b
0x01ae
0x01c0
0x01e2


0xb0
STOP
CALLDATASIZE
REVERT
Selectors
45fab
M_DELAY
ERT
0xb1b43ae5
MINIMUM_DELAY
REVERT
0x3a66f901
queueTransaction
REVERT
0x4dd18bf5
setPendingAdmin
REVERT
0x591fcdfe
cancelTransaction
REVERT
0x0e18b681
acceptAdmin
REVERT
0x26782247
pendingAdmin
REVERT
0xea
CALLDATALOAD
REVERT
51c
59e
0x0631
0x15a5
0x02f9
CALLDATASIZE
REVERT
0x0310
CALLDATALOAD
REVERT
0x034d
REVERT
0x035f
CALLDATALOAD
REVERT
0x0381
CALLDATACOPY
0x0469
CALLDATASIZE
REVERT
0x0480
CALLDATALOAD
0x11c7
REVERT
0x121f
SLOAD
SSTORE
0x04a9
CALLDATASIZE
REVERT
0x04c0
CALLDATALOAD
REVERT
0x04fd
REVERT
0x050f
CALLDATALOAD
REVERT
0x0531
CALLDATACOPY
0x02a6
0x0d4a
SLOAD
REVERT
0x0dba
SLOAD
SSTORE
0x02bb
0x0e32
SLOAD
0x0127
REVERT
0x0139
CALLDATALOAD
REVERT
0x015b
CALLDATACOPY
CALLDATALOAD
REVERT
0x01ae
REVERT
0x01c0
uniswap.eth


Selectors
0x095ea7b3
approve
REVERT
0x18160ddd
totalSupply
REVERT
0x2e1a7d4d
withdraw
REVERT
0x23b872dd
transferFrom
REVERT
0x313ce567
0x70a08231
balanceOf
REVERT
0xa9059cbb
transfer
REVERT
0x95d89b41
symbol
REVERT
0xd0e30db0
deposit
0x06fdde03
0x03d2
STOP
REVERT
0x0264
STOP
REVERT
0x0152
CALLDATALOAD
0x01ac
0x024e
CALLDATALOAD
0x01d5
CALLDATALOAD
0x02a0
CALLDATALOAD
0x037b
CALLDATALOAD
0x02ed
0x0440
SLOAD
SSTORE
0x03df
CALLDATALOAD
0x0271
0x057b
SSTORE
0x066d
0x09d9
SLOAD
REVERT
0x0b18
SLOAD
0x0bce
0x0b30
SLOAD
0x0be3
SLOAD
STOP
0x0b05
SLOAD
0x0a27
SLOAD
SSTORE
REVERT
0x068c
SLOAD
REVERT
0x0b9b
SLOAD
0x0ab4
0x06dc
SLOAD
REVERT
0x0844
SLOAD
SSTORE
WETH9


0x132d
CALLDATASIZE
0x0be3
CALLDATASIZE
0x0994
CALLDATASIZE
CALLDATALOAD
Selectors
0x0f46
CALLDATASIZE
CALLDATALOAD
0x152d
CALLDATALOAD
0x0e07
CALLDATASIZE
CALLDATALOAD
0x0d32
CALLDATASIZE
CALLDATALOAD
0x0bb4
CALLDATALOAD
0x0eb5
CALLDATASIZE
CALLDATALOAD
0x06fdde03
name
0x46423aa7
getOrderStatus
0xfb0f3ee1
fulfillBasicOrder
0x5b34b966
incrementCounter
0xf47b7740
information
0xb3a34c4c
fulfillOrder
0xf07ec373
getCounter
0xfd9f1e10
cancel
0xe7acab24
fulfillAdvancedOrd
0x034b
0x078b
CALLDATALOAD
0x0199
CALLDATASIZE
RETURN
0x01c1
CALLDATASIZE
CALLDATALOAD
SLOAD
RETURN
0x102c
CALLDATASIZE
CALLDATALOAD
0x092f
CALLDATASIZE
0x0f8b
CALLDATASIZE
0x0da3
CALLDATASIZE
CALLDATALOAD
0x0324
0x00
CALLDATASIZE
REVERT
0x15dc
0x12f8
REVERT
0x27b4
0x46e1
0x033e
0x5008
0x27ff
0x4166
0x13
CALLDATALOAD
REVERT
0x5017
SLOAD
0x027d
0x0254
0x5023
REVERT
0x0270
0x0224
REVERT

Seaport v1.1


Selectors
0xb3a34c4c
fulfillOrder
0x5b34b966
incrementCounter
0xf07ec373
getCounter
0xa900866b
getContractOffererNonce
0xe7acab24
fulfillAdvancedOrder
0xed98a574
fulfillAvailableOrders
0xf2d12b12
matchAdvancedOrders
0x87201b41
fulfillAvailableAdvancedOrders
0xa8174404
matchOrders
0xfb0f3ee1
fulfillBasicOrder
0x06fdde03
name
0x88147732
validate
0x0b81
CALLDATASIZE
CALLDATALOAD
0x04f5
CALLDATASIZE
0x0d8a
CALLDATASIZE
CALLDATALOAD
0x0b17
CALLDATASIZE
CALLDATALOAD
0x0c05
CALLDATASIZE
CALLDATALOAD
0x0cc2
CALLDATASIZE
CALLDATALOAD
0x0df2
CALLDATASIZE
CALLDATALOAD
0x07c6
CALLDATASIZE
CALLDATALOAD
0x0a60
CALLDATASIZE
CALLDATALOAD
0x01c8
CALLDATASIZE
0x042e
CALLDATASIZE
RETURN
0x0474
CALLDATASIZE
CALLDATALOAD
SLOAD
RETURN
0x19cd
CALLDATALOAD
0x059d
CALLDATASIZE
CALLDATALOAD
0x4f92
SLOAD
0x064b
0x061a
CALLDATALOAD
0x0172
CALLDATALOAD
0x18f5
CALLDATACOPY
CALLDATALOAD
0x1cd6
CALLDATACOPY
CALLDATALOAD
0x4f9e
REVERT
0x01c3
REVERT
0x1927
CALLDATALOAD
CALLDATACOPY
0x196c

Seaport v1.4

Demo?

Future for WhatsABI

- Parse deploy init code
- Detect common interfaces (proxies, ERCs, etc)
- Abstract stack tracing
- Input type detection
- Better function window detection
  (impacts visualizing and input tracking)
- Interactive call visualizer?
- Optional semi-dynamic analysis?

0xF3

github.com/shazow/whatsabi

```
import { whatsabi } from "@shazow/whatsabi";
```

@shazow
shazow.net
shazow.eth
shazow.lens