EXTROPY.IO

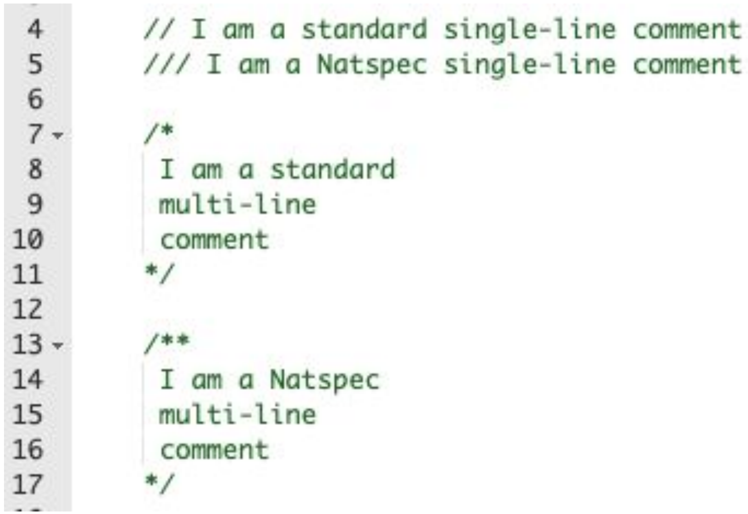
Moving to production

Lesson Plan

Documenting code
Testing / CI
Audits
Deployment / Monitoring

How to comment code in Solidity?

Comments in Solidity can be written in two different ways.



```
// I am a standard single-line comment
/// I am a Natspec single-line comment

/*
 I am a standard
 multi-line
 comment
*/

/**
 I am a Natspec
 multi-line
 comment
*/
```

What are Natspec Comments?

Special form of comments in Solidity contracts

⇒ **Machine Readable**

Used to documents variables, functions, contracts, etc...

Based on the **Ethereum Natural Language Specification Format (NatSpec)**

Single line Natspec comment: start with `///`

Multi line Natspec comment: start with `/**`, end with `*/`

What do Natspec comments do?

Document smart-contracts for developers

Generate documentation for the smart contracts automatically with third-party tools.

Annotate conditions for formal verification.

Using the `@dev` tag

Notify end-users when interacting with the contract

= more expressive

Show **relevant details** to end users at the time they will interact with the contract (= sign a transaction.

Using the `@notice` tag (only in *public* and *external* functions).

EXTROPY.IO

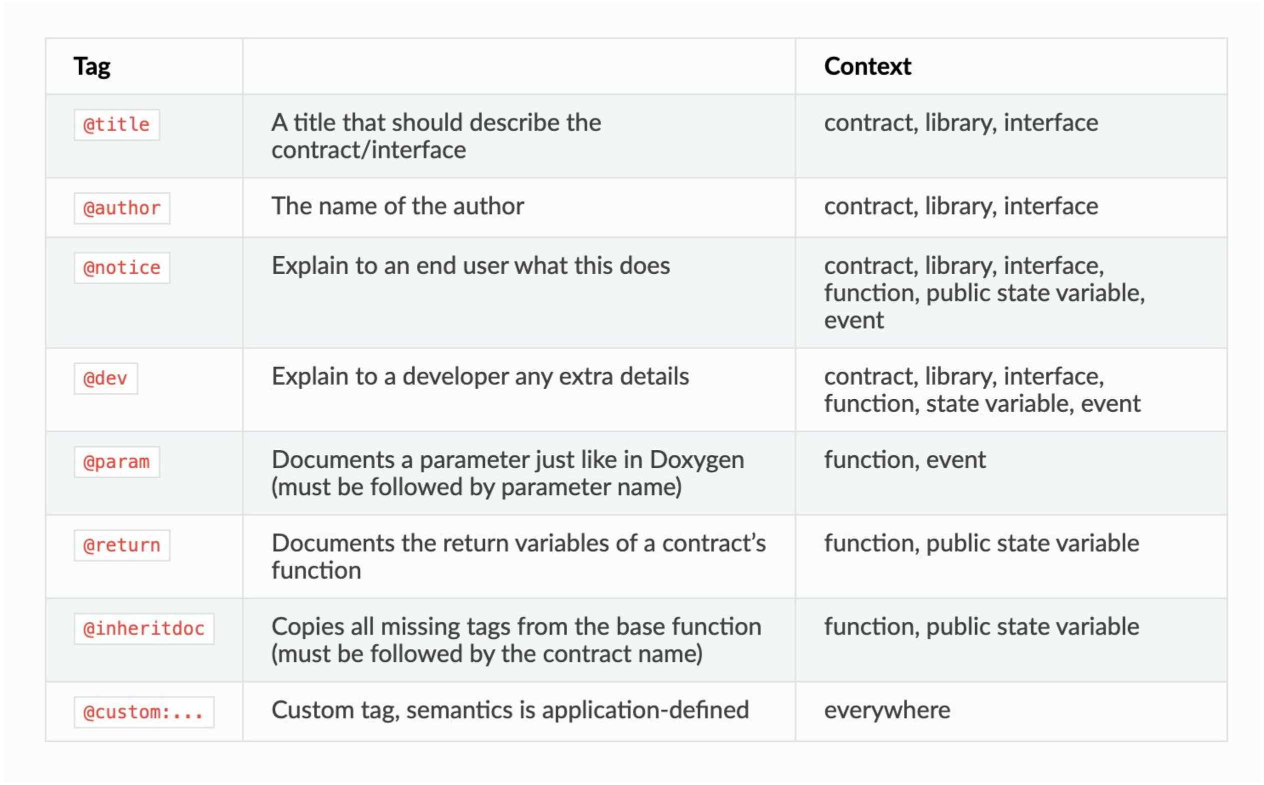
What to document in Natspec?

- **Contracts**
  - Including interfaces and libraries

- **Functions**,
  - Including constructors and public state variables (with automatic getter).

- **Events**

Supported Natspec tags



Tag		Context
`@title`	A title that should describe the contract/interface	contract, library, interface
`@author`	The name of the author	contract, library, interface
`@notice`	Explain to an end user what this does	contract, library, interface, function, public state variable, event
`@dev`	Explain to a developer any extra details	contract, library, interface, function, state variable, event
`@param`	Documents a parameter just like in Doxygen (must be followed by parameter name)	function, event
`@return`	Documents the return variables of a contract's function	function, public state variable
`@inheritdoc`	Copies all missing tags from the base function (must be followed by the contract name)	function, public state variable
`@custom:...`	Custom tag, semantics is application-defined	everywhere

Documenting **Functions** with Natspec

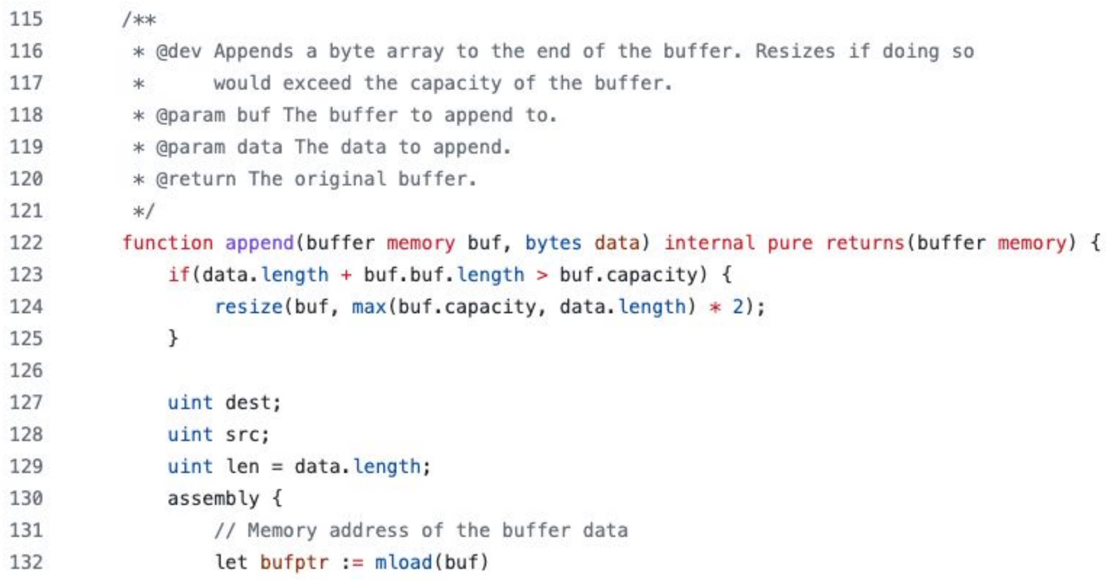
`@param` **must** be followed by the variable name passed as argument.

`@return` **good practice is to put return type** or the **name of the variable** returned.

`@notice` **only relevant in public + external** functions (only for userdocs).

Source: ***Buffer Library from Oraclize***

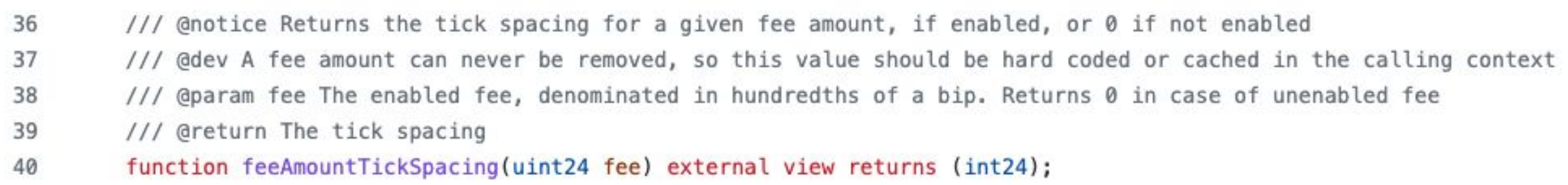
https://github.com/provable-things/ethereum-api/blob/ff29c6771a589b148ef01c0634b707e54793e7f6/oraclizeAPI_0.4.25.sol#L115-L122



```
115     /**
116      * @dev Appends a byte array to the end of the buffer. Resizes if doing so
117      *      would exceed the capacity of the buffer.
118      * @param buf The buffer to append to.
119      * @param data The data to append.
120      * @return The original buffer.
121      */
122     function append(buffer memory buf, bytes data) internal pure returns(buffer memory) {
123         if(data.length + buf.buf.length > buf.capacity) {
124             resize(buf, max(buf.capacity, data.length) * 2);
125         }
126 
127         uint dest;
128         uint src;
129         uint len = data.length;
130         assembly {
131             // Memory address of the buffer data
132             let bufptr := mload(buf)
```

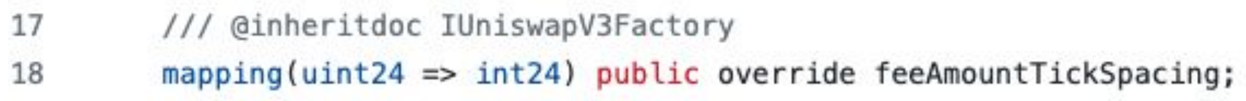
Documenting **Functions** with Natspec

Source: https://github.com/Uniswap/v3-core/blob/main/contracts/interfaces/IUniswapV3Factory.sol



```
    /// @notice Returns the tick spacing for a given fee amount, if enabled, or 0 if not enabled
    /// @dev A fee amount can never be removed, so this value should be hard coded or cached in the calling context
    /// @param fee The enabled fee, denominated in hundredths of a bip. Returns 0 in case of unenabled fee
    /// @return The tick spacing
    function feeAmountTickSpacing(uint24 fee) external view returns (int24);
```

Source: https://github.com/Uniswap/v3-core/blob/c05a0e2c8c08c460fb4d05cfdda30b3ad8deeaac/contracts/UniswapV3Factory.sol#L17-L18



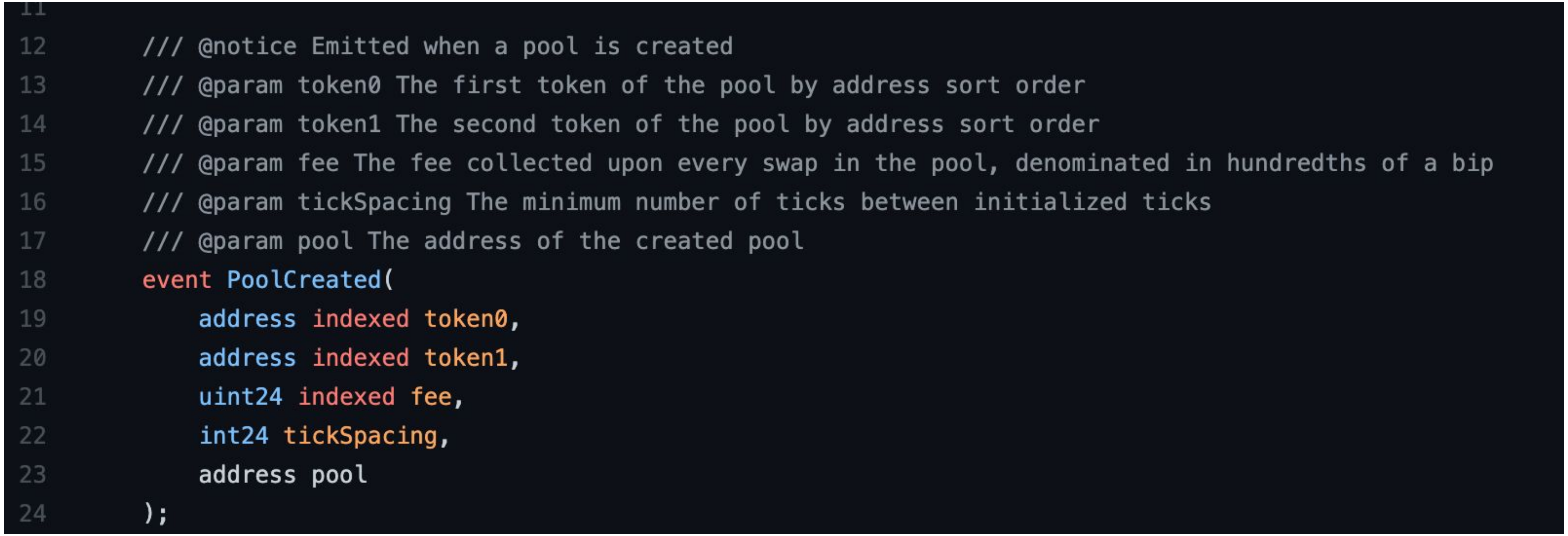
```
    /// @inheritdoc IUniswapV3Factory
    mapping(uint24 => int24) public override feeAmountTickSpacing;
```

Double example where:

- A public state variable (= automatic getter function)
- Inherit the docs of the parent / base contract (= here the interface)

Documenting **Events** with Natspec

Source: https://github.com/Uniswap/v3-core/blob/main/contracts/interfaces/IUniswapV3Factory.sol



```
    /// @notice Emitted when a pool is created
    /// @param token0 The first token of the pool by address sort order
    /// @param token1 The second token of the pool by address sort order
    /// @param fee The fee collected upon every swap in the pool, denominated in hundredths of a bip
    /// @param tickSpacing The minimum number of ticks between initialized ticks
    /// @param pool The address of the created pool
    event PoolCreated(
        address indexed token0,
        address indexed token1,
        uint24 indexed fee,
        int24 tickSpacing,
        address pool
    );
```

Documentation Generator

The Solidity compiler generates a JSON file

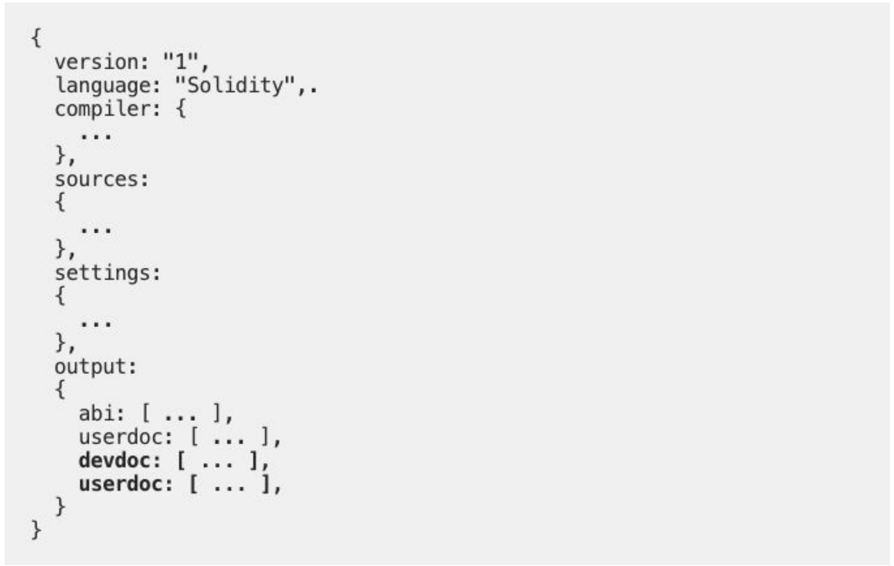
= artifacts with contract metadata, that contain:

- Compiler version
- ABI
- Contract bytecode
- …

But also the documentation generated by Natspec comments

(in the "output" section at the end of the file)

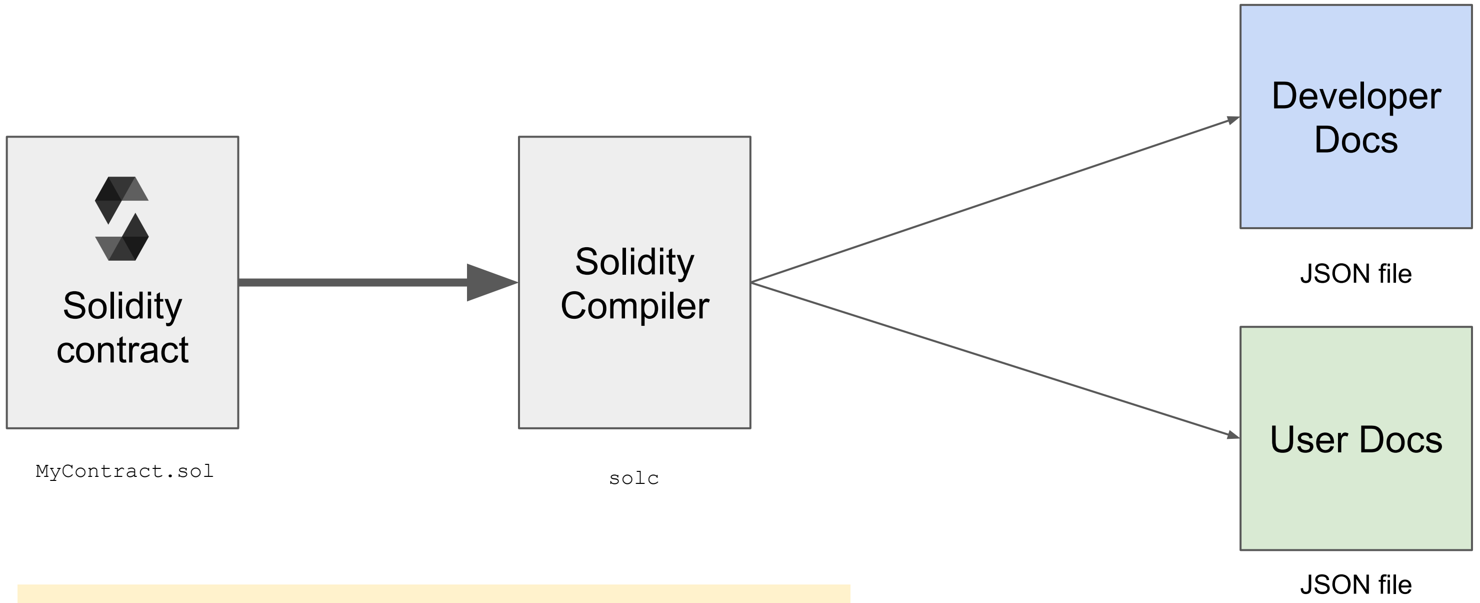
NB: when doing *truffle compile*, look at the JSON file under the */build* folder. If your contract had Natspec comments in it, you will see the **"devdoc"** and **"userdoc"** sections.



```
{
  version: "1",
  language: "Solidity",.
  compiler: {
    ...
  },
  sources:
  {
    ...
  },
  settings:
  {
    ...
  },
  output:
  {
    abi: [ ... ],
    userdoc: [ ... ],
    devdoc: [ ... ],
    userdoc: [ ... ],
  }
}
```

Documentation Generator

Generating the contract docs from the Natspec comments.



```
$ solc --userdoc --devdoc MyContract.sol
```

Let’s use our Volcano coin contract as an example !

We are going to produce its developer + user docs

Steps to reproduce

1. Install the **Solidity** Compiler: **solc**

NB: do not use the npm package *solcjs*, it does not have the option to parse Natpsec comments and generate user / dev docs.

Source: *https://docs.soliditylang.org/en/v0.8.7/installing-solidity.html#linux-packages*

Mac

```
brew update
brew upgrade
brew tap ethereum/ethereum
brew install solidity
```

Linux

```
sudo add-apt-repository ppa:ethereum/ethereum
sudo add-apt-repository ppa:ethereum/ethereum-dev
sudo apt-get update
sudo apt-get install solc
```

2. Create a local file ***VolcanoCoin.sol*** and copy into it the code from this link:

https://gist.github.com/CJ42/158d76d19172eb40e6534ed5c41ec020

Developer Docs output

```
$ solc --devdoc VolcanoCoin.sol
```

```
{
  "author": "EncodeAcademy - {enter your name here}",
  "details": "This contract stores + keep track of all the tokens transfers made by each users via a custom Payment structure",
  "events":
  {
    "supplyEvent(uint256)":
    {
      "details": "event emitted when owmner increases the totalSupply",
      "params":
      {
        "newTotalSupply": "the newly updated totalSupply "
      }
    },
    "transferComplete(address,uint256)":
    {
      "details": "event emitted when a token transfer has been successful",
      "params":
      {
        "amount": "amount of tokens transfered",
        "recipient": "the beneficiary of the token transfer"
      }
    }
  },
  "kind": "dev",
  "methods":
  {
    "getTotalSupply()":
    {
      "details": "Return maximum number of tokens created initially + added by owner via updateTotalSupply() ",
      "returns":
      {
        "_0": "uint256 totalSupply"
      }
    },
    "transfer(address,uint256)":
    {
      "details": "Transfer `amount` of tokens",
      "params":
      {
        "amount": "amount of tokens to transfer",
        "dest": "receiving ethereum address of the tokens"
      }
    },
    "userBalance(address)":
    {
      "details": "Get available tokens balance for a user",
      "returns":
      {
        "_0": "uint256 user's balance"
      }
    },
    "userTransactions(address)":
    {
      "details": "Return an array of Payment structs, containing recipient and amount transfered",
      "params":
      {
        "sender": "The address to get the list of previously made transactions."
      },
      "returns":
      {
        "_0": "Payment[] list of previous transfers"
      }
    }
  },
  "stateVariables":
  {
    "balances":
    {
      "details": "Get available tokens balance for a user",
      "return": "uint256 user's balance",
      "returns":
      {
        "_0": "uint256 user's balance"
      }
    },
    "totalSupply":
    {
      "details": "Return maximum number of tokens created initially + added by owner via updateTotalSupply() ",
      "return": "uint256 totalSupply",
      "returns":
      {
        "_0": "uint256 totalSupply"
      }
    }
  },
  "title": "VolcanoCoin, your first Solidity smart contract",
  "version": 1
}
```

Developer Docs output

```
$ solc --devdoc VolcanoCoin.sol
```

Function signature

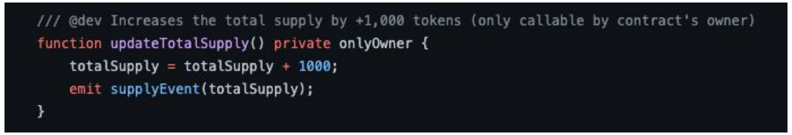
= function name + parameter types in parenthese (without spaces)

= **hashing this with `keccak256` gives you the bytes4 function selector**

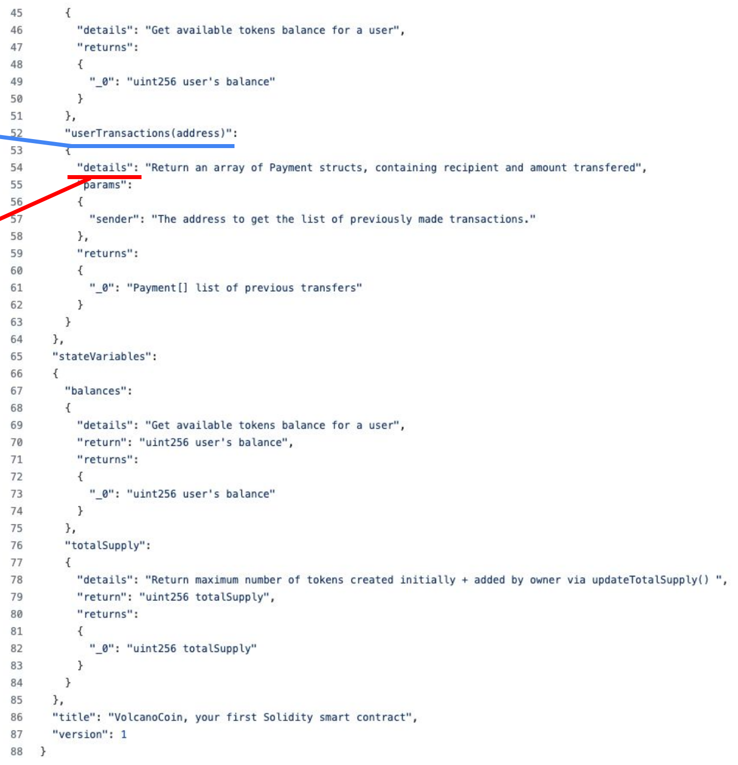
`@dev` tag

Only `public` and `external` functions are shown

(`private` and `internal` functions are not parsed)



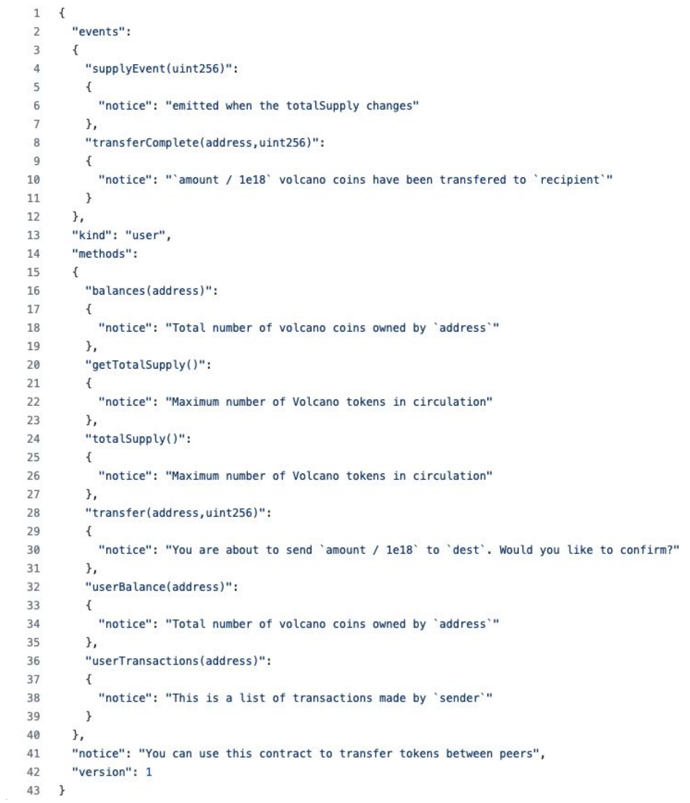
```
/// @dev Increases the total supply by +1,000 tokens (only callable by contract's owner)
function updateTotalSupply() private onlyOwner {
    totalSupply = totalSupply + 1000;
    emit supplyEvent(totalSupply);
}
```



```
      {
        "details": "Get available tokens balance for a user",
        "returns":
        {
          "_0": "uint256 user's balance"
        }
      },
      "userTransactions(address)":
      {
        "details": "Return an array of Payment structs, containing recipient and amount transfered",
        "params":
        {
          "sender": "The address to get the list of previously made transactions."
        },
        "returns":
        {
          "_0": "Payment[] list of previous transfers"
        }
      }
    },
    "stateVariables":
    {
      "balances":
      {
        "details": "Get available tokens balance for a user",
        "return": "uint256 user's balance",
        "returns":
        {
          "_0": "uint256 user's balance"
        }
      },
      "totalSupply":
      {
        "details": "Return maximum number of tokens created initially + added by owner via updateTotalSupply() ",
        "return": "uint256 totalSupply",
        "returns":
        {
          "_0": "uint256 totalSupply"
        }
      }
    },
    "title": "VolcanoCoin, your first Solidity smart contract",
    "version": 1
  }
```

User Docs output

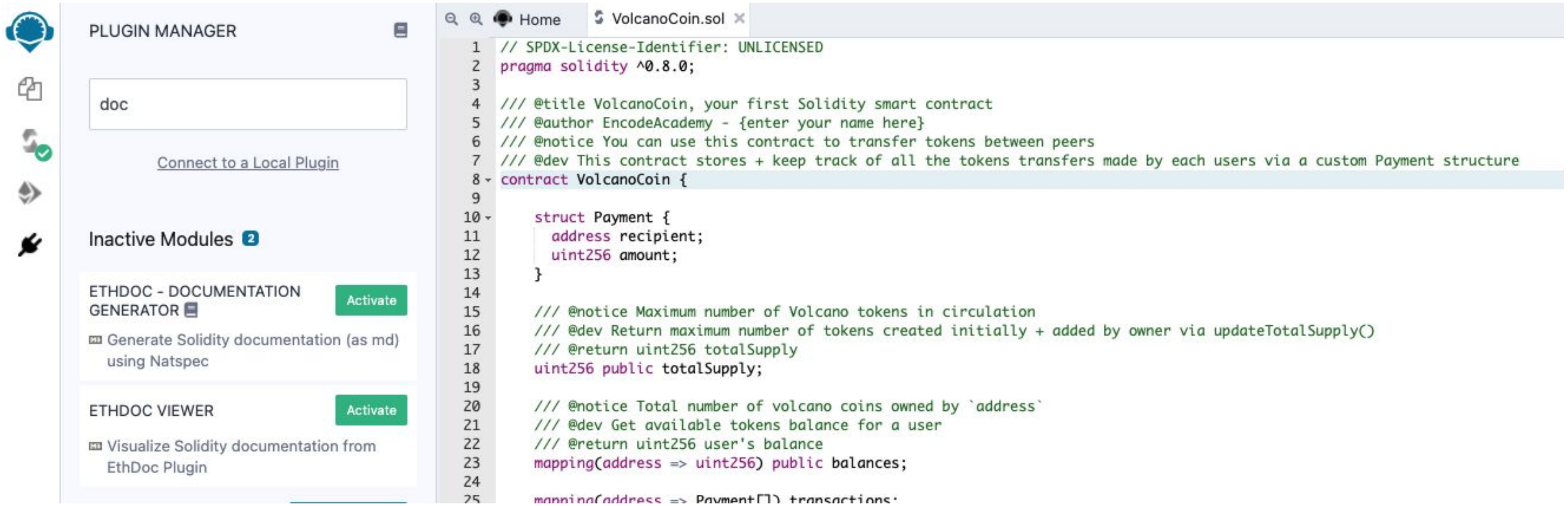
```
$ solc --userdoc VolcanoCoin.sol
```



```
{
  "events":
  {
    "supplyEvent(uint256)":
    {
      "notice": "emitted when the totalSupply changes"
    },
    "transferComplete(address,uint256)":
    {
      "notice": "`amount / 1e18` volcano coins have been transfered to `recipient`"
    }
  },
  "kind": "user",
  "methods":
  {
    "balances(address)":
    {
      "notice": "Total number of volcano coins owned by `address`"
    },
    "getTotalSupply()":
    {
      "notice": "Maximum number of Volcano tokens in circulation"
    },
    "totalSupply()":
    {
      "notice": "Maximum number of Volcano tokens in circulation"
    },
    "transfer(address,uint256)":
    {
      "notice": "You are about to send `amount / 1e18` to `dest`. Would you like to confirm?"
    },
    "userBalance(address)":
    {
      "notice": "Total number of volcano coins owned by `address`"
    },
    "userTransactions(address)":
    {
      "notice": "This is a list of transactions made by `sender`"
    }
  },
  "notice": "You can use this contract to transfer tokens between peers",
  "version": 1
}
```

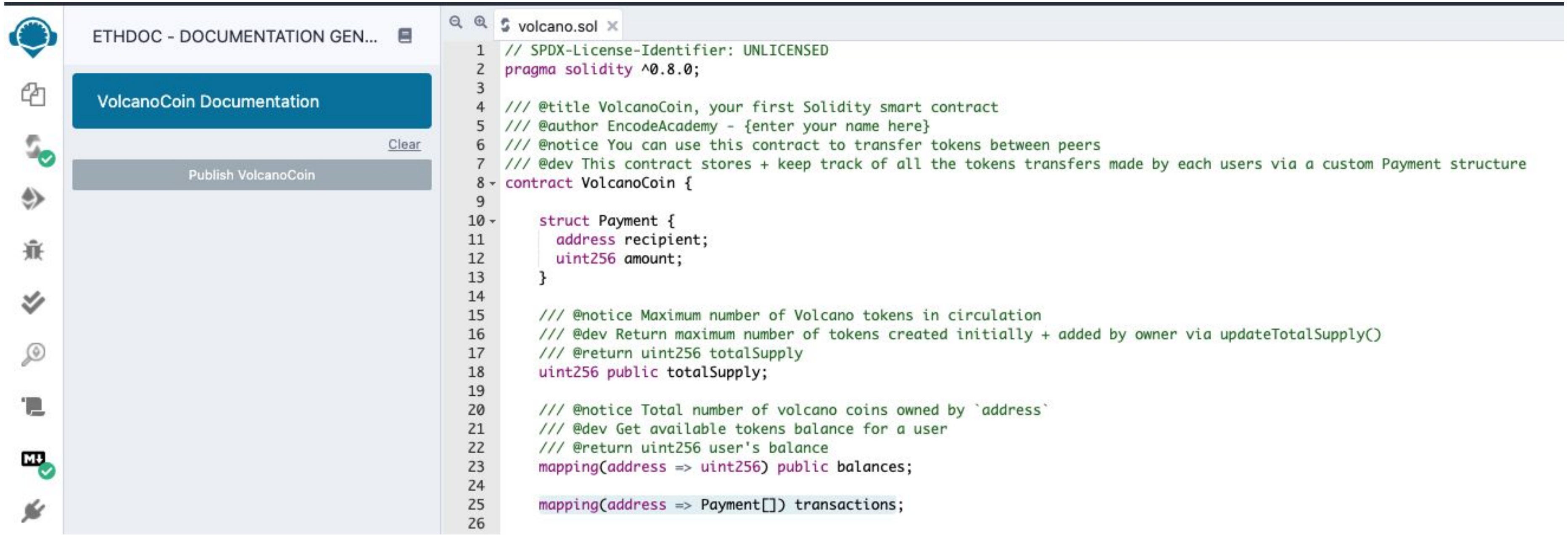
EXTROPY.IO

User Docs output - using Remix EthDoc plugin


PLUGIN MANAGER
doc
Connect to a Local Plugin
Inactive Modules 2
ETHDOC - DOCUMENTATION GENERATOR
Activate
Generate Solidity documentation (as md) using Natspec
ETHDOC VIEWER
Activate
Visualize Solidity documentation from EthDoc Plugin
Home
VolcanoCoin.sol

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

/// @title VolcanoCoin, your first Solidity smart contract
/// @author EncodeAcademy - {enter your name here}
/// @notice You can use this contract to transfer tokens between peers
/// @dev This contract stores + keep track of all the tokens transfers made by each users via a custom Payment structure
contract VolcanoCoin {

    struct Payment {
      address recipient;
      uint256 amount;
    }

    /// @notice Maximum number of Volcano tokens in circulation
    /// @dev Return maximum number of tokens created initially + added by owner via updateTotalSupply()
    /// @return uint256 totalSupply
    uint256 public totalSupply;

    /// @notice Total number of volcano coins owned by `address`
    /// @dev Get available tokens balance for a user
    /// @return uint256 user's balance
    mapping(address => uint256) public balances;

    mapping(address => Payment[]) transactions;
```

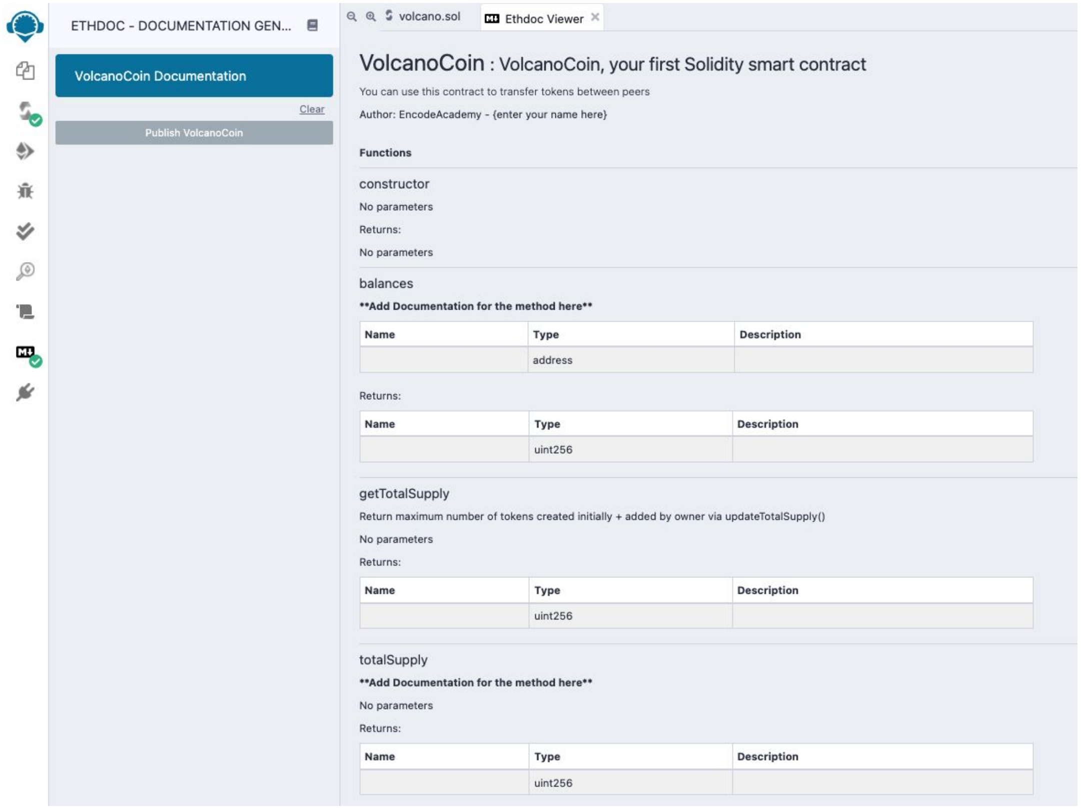
User Docs output - using Remix EthDoc plugin



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

/// @title VolcanoCoin, your first Solidity smart contract
/// @author EncodeAcademy - {enter your name here}
/// @notice You can use this contract to transfer tokens between peers
/// @dev This contract stores + keep track of all the tokens transfers made by each users via a custom Payment structure
contract VolcanoCoin {

    struct Payment {
      address recipient;
      uint256 amount;
    }

    /// @notice Maximum number of Volcano tokens in circulation
    /// @dev Return maximum number of tokens created initially + added by owner via updateTotalSupply()
    /// @return uint256 totalSupply
    uint256 public totalSupply;

    /// @notice Total number of volcano coins owned by `address`
    /// @dev Get available tokens balance for a user
    /// @return uint256 user's balance
    mapping(address => uint256) public balances;

    mapping(address => Payment[]) transactions;

```

EXTROPY.IO

User Docs output - using Remix EthDoc plugin



VolcanoCoin : VolcanoCoin, your first Solidity smart contract

You can use this contract to transfer tokens between peers

Author: EncodeAcademy - {enter your name here}

Functions

constructor

No parameters

Returns:

No parameters

balances

Add Documentation for the method here

Name	Type	Description
	address	

Returns:

Name	Type	Description
	uint256	

getTotalSupply

Return maximum number of tokens created initially + added by owner via updateTotalSupply()

No parameters

Returns:

Name	Type	Description
	uint256	

totalSupply

Add Documentation for the method here

No parameters

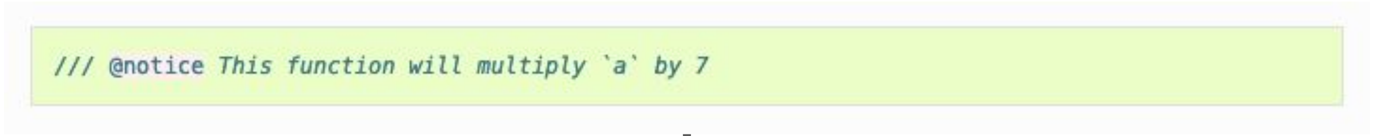
Returns:

Name	Type	Description
	uint256	

User Doc Output - Dynamic Expressions

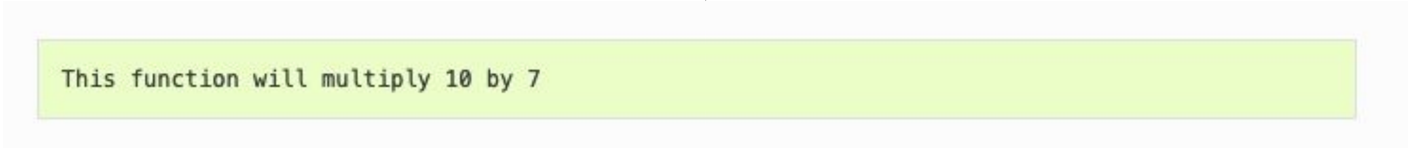
Solidity compiler ⇒ examine NatSpec comments ⇒ generate JSON.
End user software *(eg: Metamask*) can consume this document.

Example with **@notice** tag.



```
/// @notice This function will multiply `a` by 7
```

End user call function with `` `a = 10` `` *as parameter*

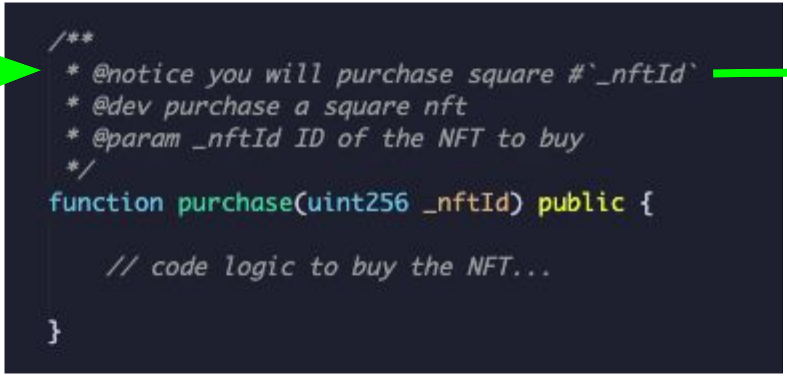


```
This function will multiply 10 by 7
```

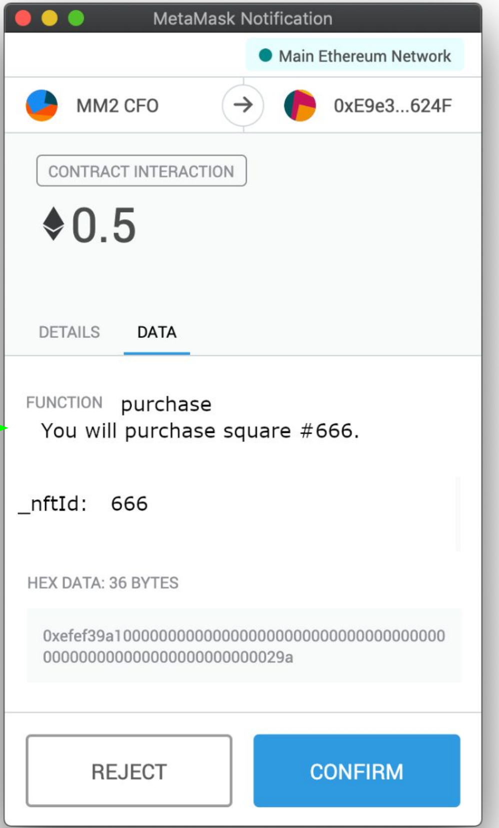
Parsing **Natspec comments** for end-user

Parsing @notice tag using Radspec interpreter

From **Aragon**



```
/**
 * @notice you will purchase square #`_nftId`
 * @dev purchase a square nft
 * @param _nftId ID of the NFT to buy
 */
function purchase(uint256 _nftId) public {

    // code logic to buy the NFT...

}
```



MetaMask Notification

Main Ethereum Network

MM2 CFO → 0xE9e3...624F

CONTRACT INTERACTION

0.5

DETAILS DATA

FUNCTION purchase
You will purchase square #666.

_nftId: 666

HEX DATA: 36 BYTES

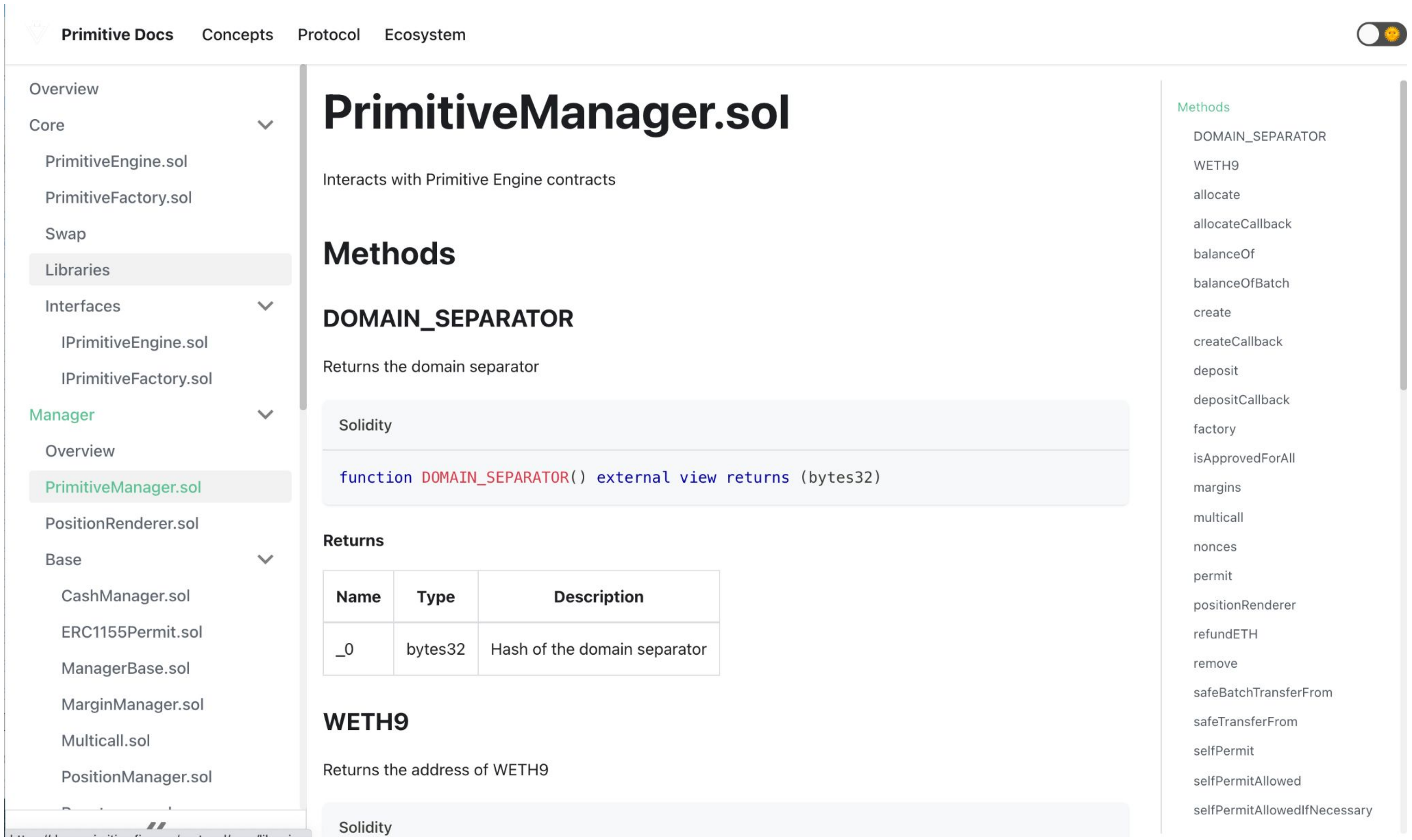
0xefef39a1000000000000000000000000000000000000000000000000000000000000029a

Dodoc: A zero-config Hardhat plugin to generate documentation

```
Install with
npm i @primitivefi/hardhat-dodoc
```

In hardhat.config.js

```
require('@primitivefi/hardhat-dodoc');
```



PrimitiveManager.sol

Interacts with Primitive Engine contracts

Methods

DOMAIN_SEPARATOR

Returns the domain separator

Solidity

```
function DOMAIN_SEPARATOR() external view returns (bytes32)
```

Returns

Name	Type	Description
_0	bytes32	Hash of the domain separator

WETH9

Returns the address of WETH9

Solidity

Testing / CI

Hardhat / Truffle - unit testing

e.g. npx hardhat test

Use ganache to fork the mainnet

```
npx ganache-cli --fork https://mainnet.infura.io/v3/<your project -id>
```

```
npx ganache-cli --fork https://eth-mainnet.alchemyapi.io/v2/<project-key>
```

or with hardhat

```
npx hardhat node --fork https://eth-mainnet.alchemyapi.io/v2/<key>
```

Ethereum Test Networks

The Ropsten test network is a Proof-of-Work testnet for Ethereum. To acquire ETH on Ropsten, one can mine on the network.

The Kovan test network is a Proof-of-Authority testnet for Ethereum, originally started by the Parity team. To acquire ETH on Kovan, one can request it from a faucet.

The Rinkeby test network is a Proof-of-Authority testnet for Ethereum, originally started by the Geth team. To acquire ETH on Rinkeby, one can request it from a faucet.

The Görli test network is a Proof-of-Authority testnet for Ethereum, originally proposed by Chainsafe and Afri Schoedon. To acquire ETH on Görli, one can use the one-way throttled bridge from any of the other three test networks.

EXTROPY.IO

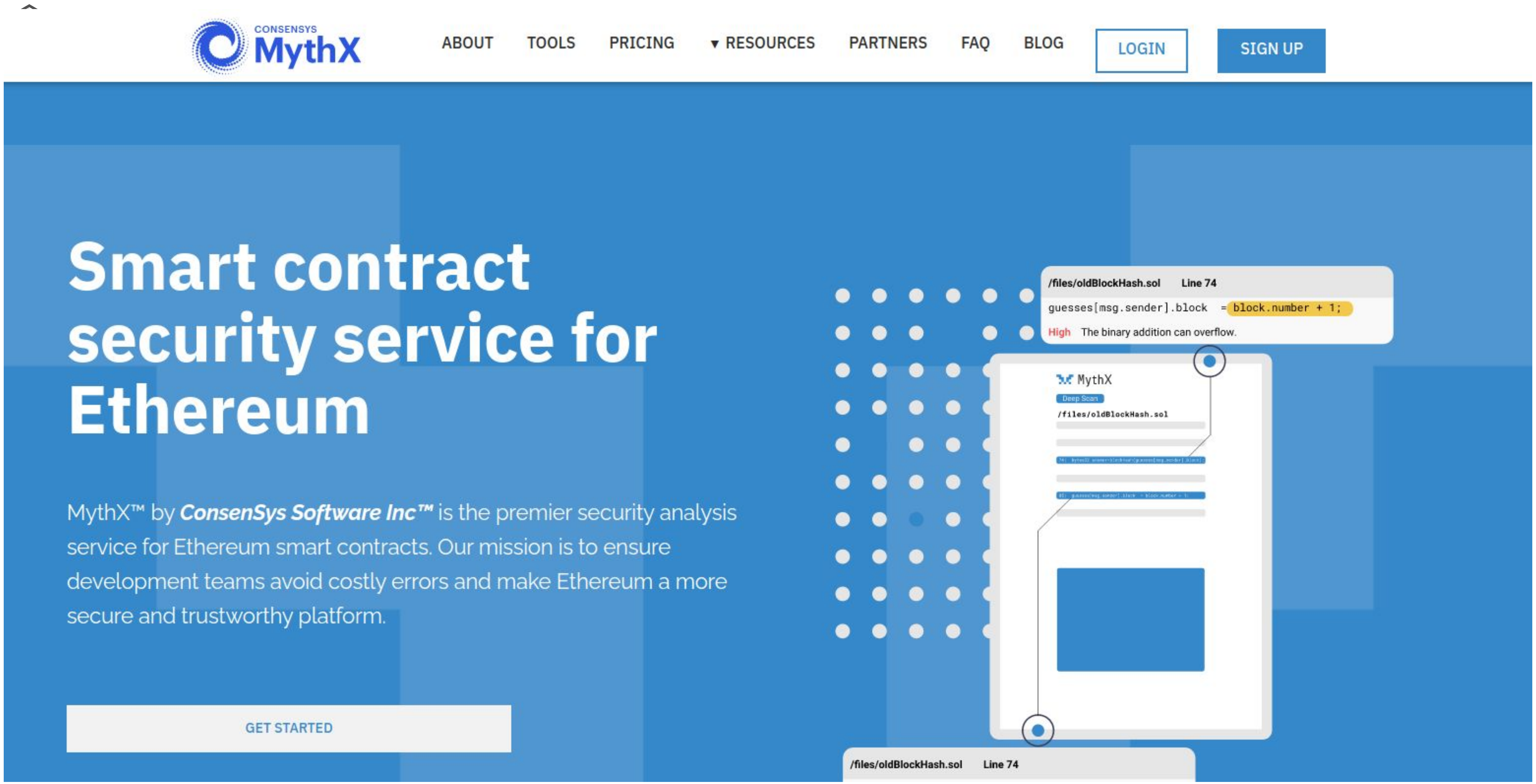
Continuous Integration

For example in Gitlab

```
test:Solidity Test:
  stage: test
  script:
    - echo "Testing Solidity"
    - cd blockchain
    - nvm use node
    - node -v
    - truffle console
    - truffle migrate --reset
    - truffle test
  retry:
    max: 2
```

Audits

- Static Analysis
- Code Review
- Vulnerability Database



Smart contract security service for Ethereum

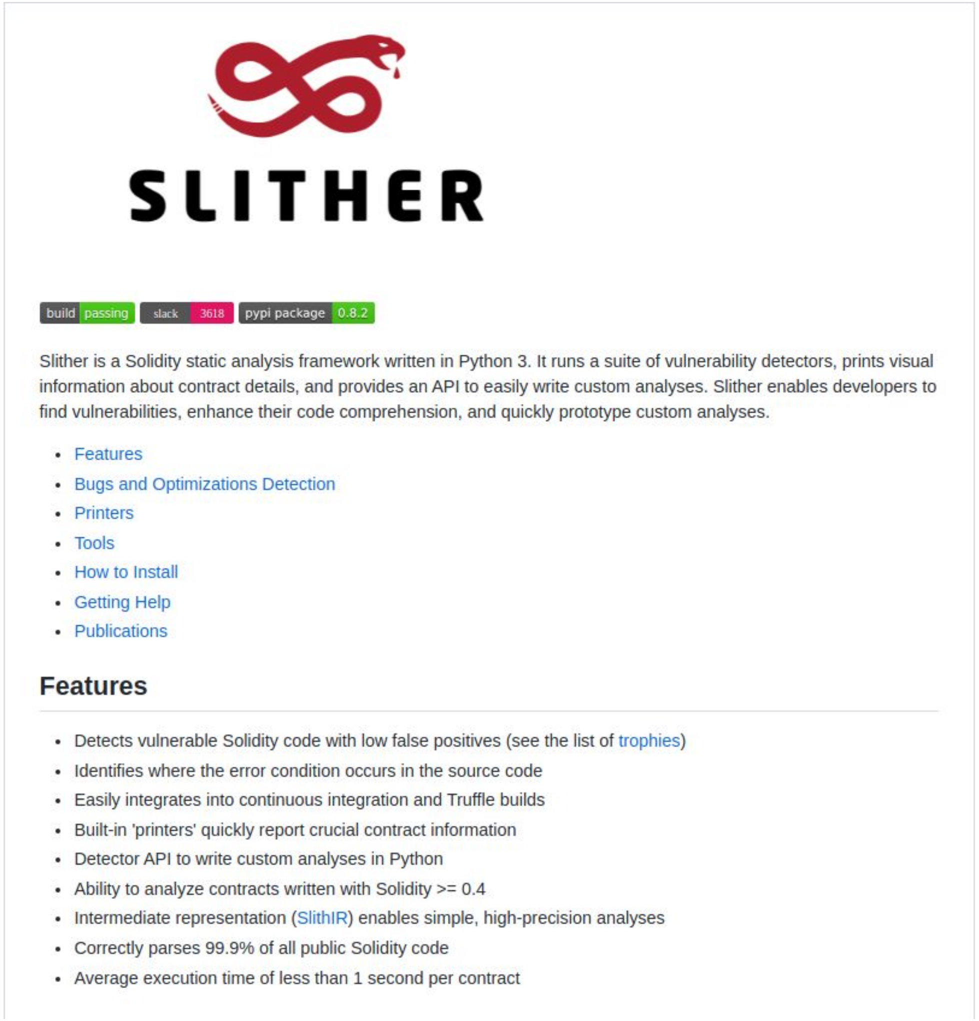
MythX™ by ***ConsenSys Software Inc™*** is the premier security analysis service for Ethereum smart contracts. Our mission is to ensure development teams avoid costly errors and make Ethereum a more secure and trustworthy platform.

/files/oldBlockHash.sol Line 74

```
guesses[msg.sender].block = block.number + 1;
```

High The binary addition can overflow.

/files/oldBlockHash.sol Line 74



SLITHER

build passing | slack 3618 | pypi package 0.8.2

Slither is a Solidity static analysis framework written in Python 3. It runs a suite of vulnerability detectors, prints visual information about contract details, and provides an API to easily write custom analyses. Slither enables developers to find vulnerabilities, enhance their code comprehension, and quickly prototype custom analyses.

- Features
- Bugs and Optimizations Detection
- Printers
- Tools
- How to Install
- Getting Help
- Publications

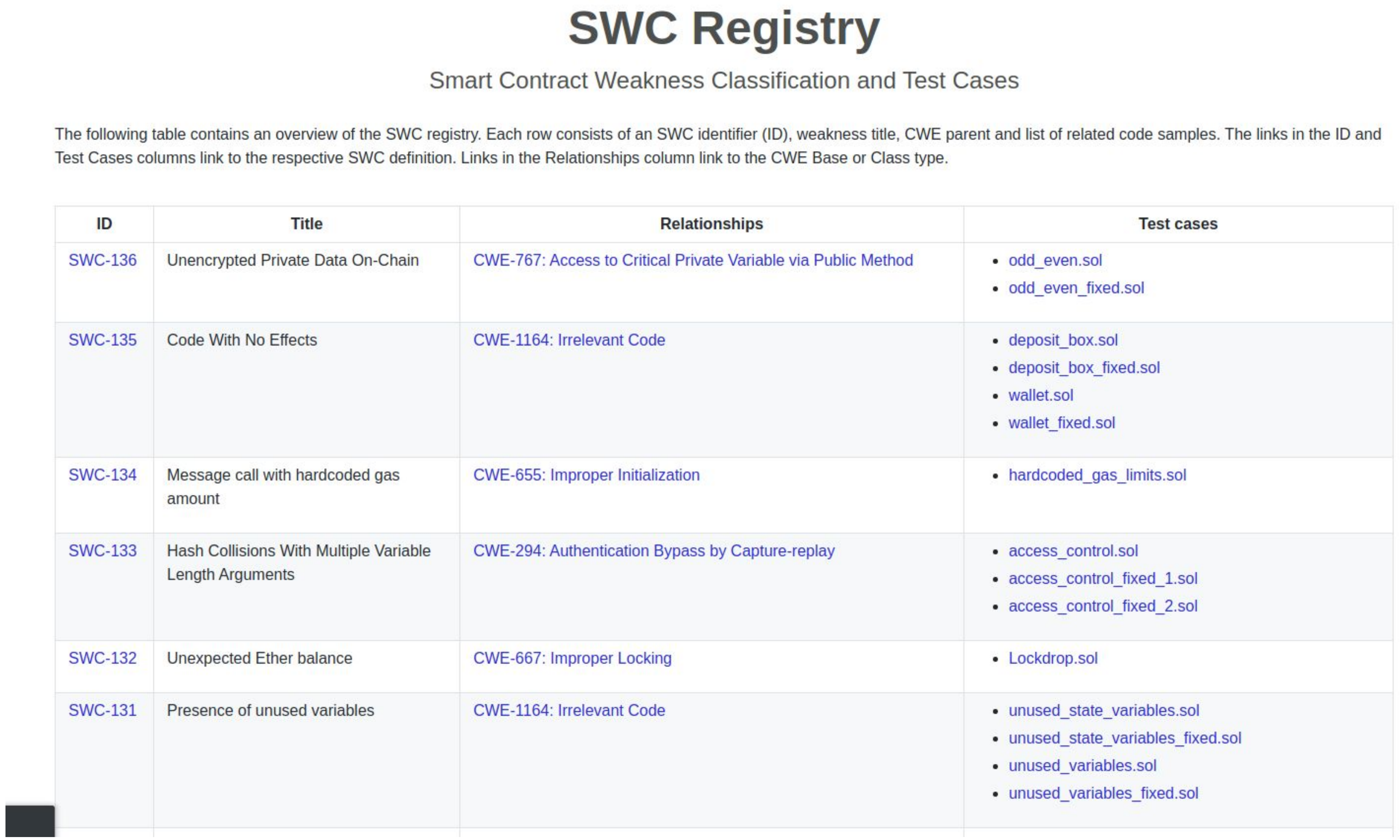
Features

- Detects vulnerable Solidity code with low false positives (see the list of trophies)
- Identifies where the error condition occurs in the source code
- Easily integrates into continuous integration and Truffle builds
- Built-in 'printers' quickly report crucial contract information
- Detector API to write custom analyses in Python
- Ability to analyze contracts written with Solidity >= 0.4
- Intermediate representation (SlithIR) enables simple, high-precision analyses
- Correctly parses 99.9% of all public Solidity code
- Average execution time of less than 1 second per contract

[🌐 📩 🔥]

Solidity Metrics for Academy

Table of contents



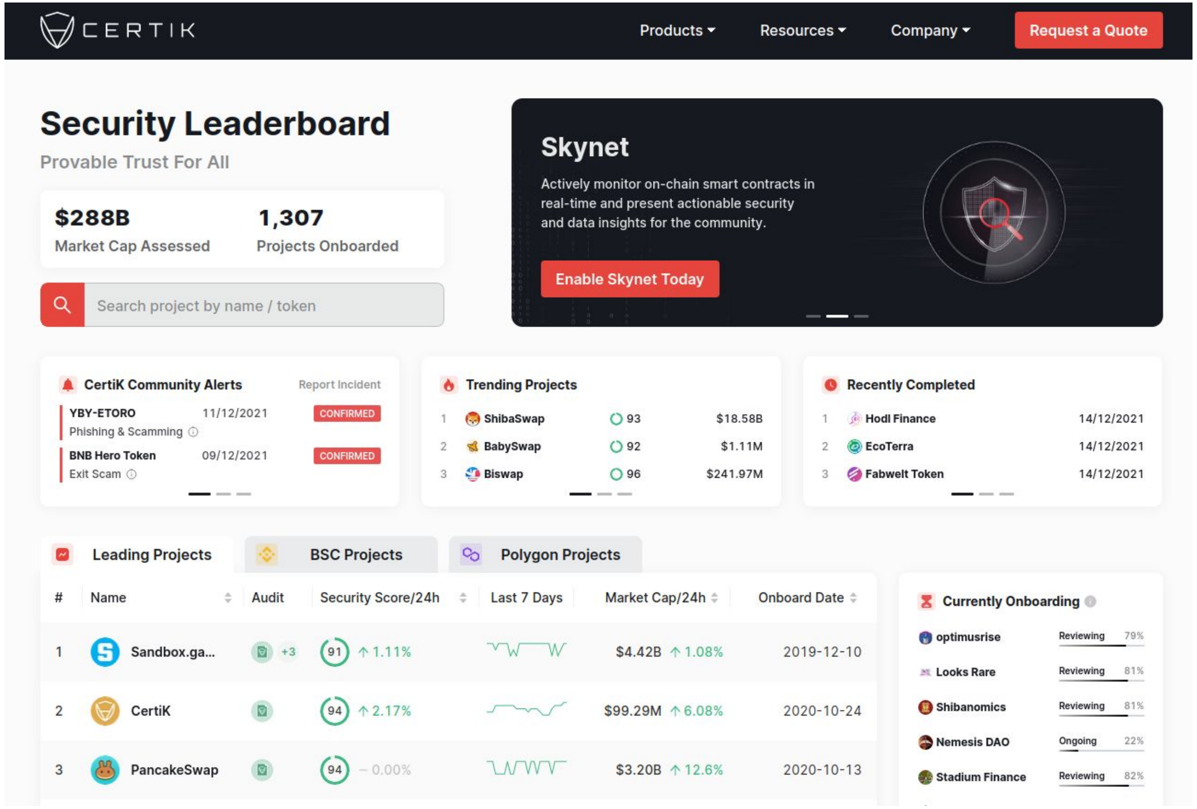
SWC Registry

Smart Contract Weakness Classification and Test Cases

The following table contains an overview of the SWC registry. Each row consists of an SWC identifier (ID), weakness title, CWE parent and list of related code samples. The links in the ID and Test Cases columns link to the respective SWC definition. Links in the Relationships column link to the CWE Base or Class type.

ID	Title	Relationships	Test cases
SWC-136	Unencrypted Private Data On-Chain	CWE-767: Access to Critical Private Variable via Public Method	• odd_even.sol • odd_even_fixed.sol
SWC-135	Code With No Effects	CWE-1164: Irrelevant Code	• deposit_box.sol • deposit_box_fixed.sol • wallet.sol • wallet_fixed.sol
SWC-134	Message call with hardcoded gas amount	CWE-655: Improper Initialization	• hardcoded_gas_limits.sol
SWC-133	Hash Collisions With Multiple Variable Length Arguments	CWE-294: Authentication Bypass by Capture-replay	• access_control.sol • access_control_fixed_1.sol • access_control_fixed_2.sol
SWC-132	Unexpected Ether balance	CWE-667: Improper Locking	• Lockdrop.sol
SWC-131	Presence of unused variables	CWE-1164: Irrelevant Code	• unused_state_variables.sol • unused_state_variables_fixed.sol • unused_variables.sol • unused_variables_fixed.sol

Notable Audit Companies



CERTIK

Products ▾ Resources ▾ Company ▾ Request a Quote

Security Leaderboard

Provable Trust For All

$288B Market Cap Assessed

1,307 Projects Onboarded

Search project by name / token

Skynet

Actively monitor on-chain smart contracts in real-time and present actionable security and data insights for the community.

Enable Skynet Today

CertiK Community Alerts — Report Incident

YBY-ETORO Phishing & Scamming	11/12/2021	CONFIRMED
BNB Hero Token Exit Scam	09/12/2021	CONFIRMED

Trending Projects

1	ShibaSwap	93	$18.58B
2	BabySwap	92	$1.11M
3	Biswap	96	$241.97M

Recently Completed

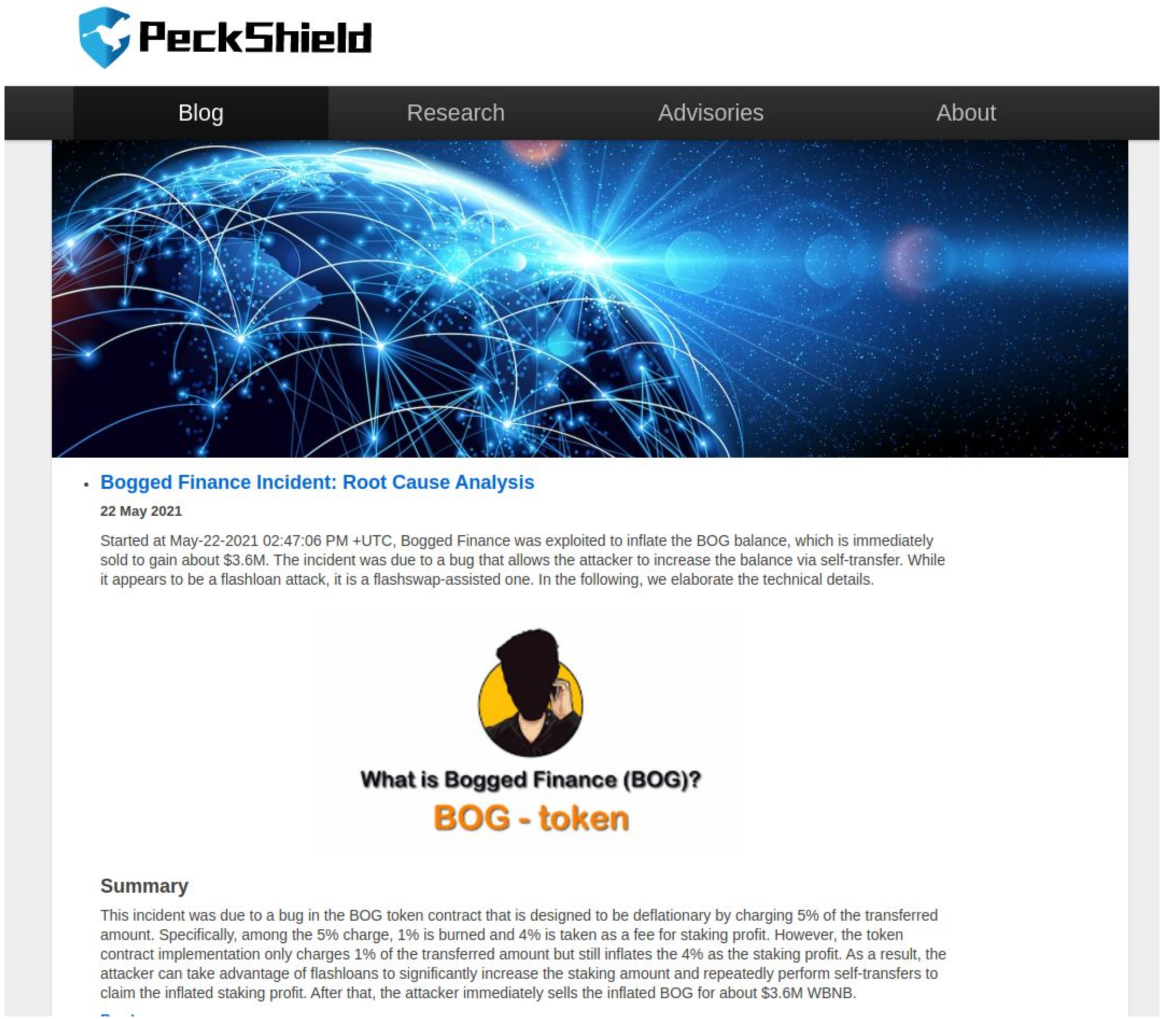
1	Hodl Finance	14/12/2021
2	EcoTerra	14/12/2021
3	Fabwelt Token	14/12/2021

Leading Projects | BSC Projects | Polygon Projects

#	Name	Audit	Security Score/24h	Last 7 Days	Market Cap/24h	Onboard Date
1	Sandbox.ga...	+3	91 ↑ 1.11%		$4.42B ↑ 1.08%	2019-12-10
2	CertiK		94 ↑ 2.17%		$99.29M ↑ 6.08%	2020-10-24
3	PancakeSwap		94 – 0.00%		$3.20B ↑ 12.6%	2020-10-13

Currently Onboarding

optimusrise	Reviewing	79%
Looks Rare	Reviewing	81%
Shibanomics	Reviewing	81%
Nemesis DAO	Ongoing	22%
Stadium Finance	Reviewing	82%



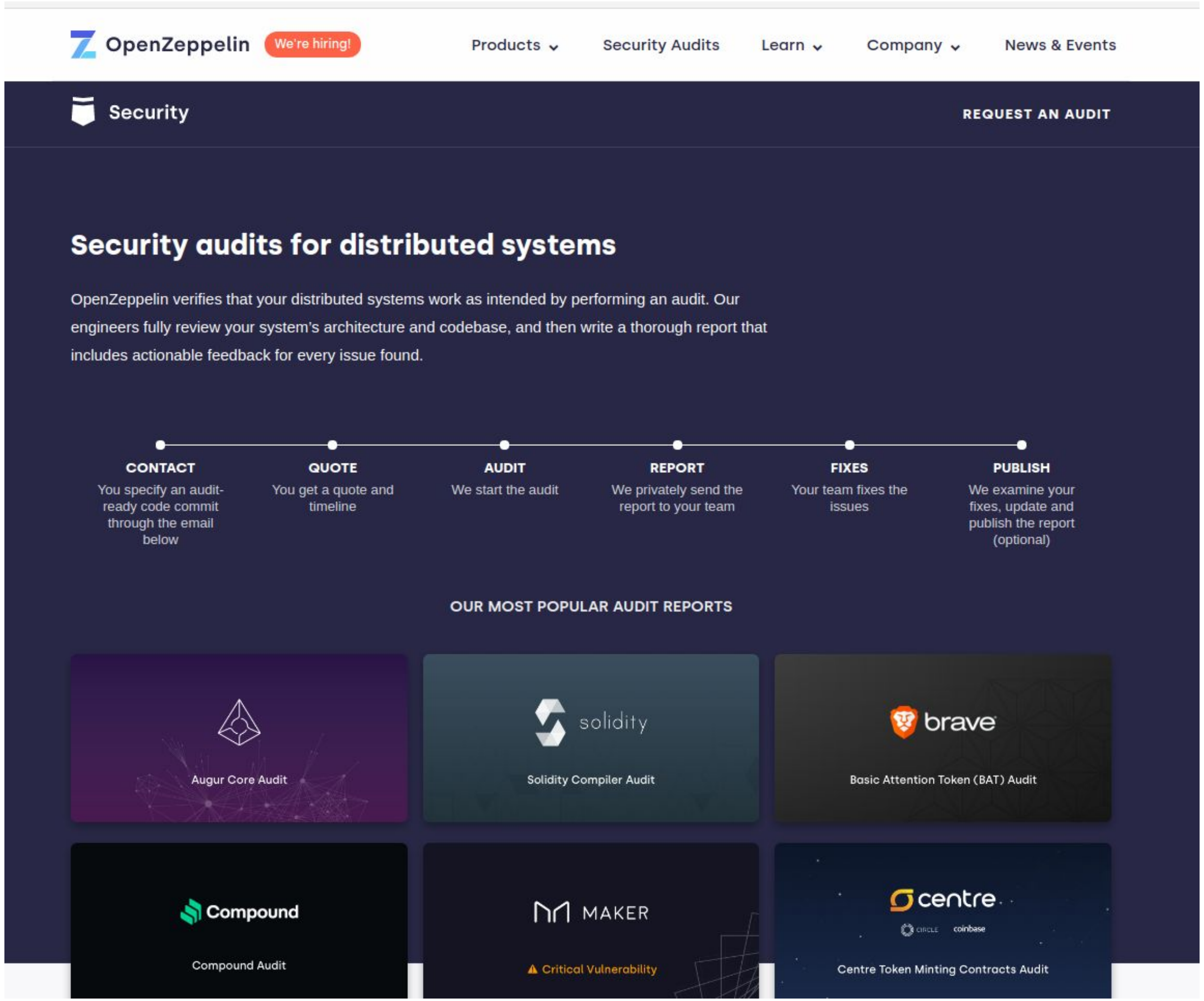
- **Bogged Finance Incident: Root Cause Analysis**

  22 May 2021

  Started at May-22-2021 02:47:06 PM +UTC, Bogged Finance was exploited to inflate the BOG balance, which is immediately sold to gain about $3.6M. The incident was due to a bug that allows the attacker to increase the balance via self-transfer. While it appears to be a flashloan attack, it is a flashswap-assisted one. In the following, we elaborate the technical details.

  ### Summary

  This incident was due to a bug in the BOG token contract that is designed to be deflationary by charging 5% of the transferred amount. Specifically, among the 5% charge, 1% is burned and 4% is taken as a fee for staking profit. However, the token contract implementation only charges 1% of the transferred amount but still inflates the 4% as the staking profit. As a result, the attacker can take advantage of flashloans to significantly increase the staking amount and repeatedly perform self-transfers to claim the inflated staking profit. After that, the attacker immediately sells the inflated BOG for about $3.6M WBNB.



Security audits for distributed systems

OpenZeppelin verifies that your distributed systems work as intended by performing an audit. Our engineers fully review your system's architecture and codebase, and then write a thorough report that includes actionable feedback for every issue found.

CONTACT You specify an audit-ready code commit through the email below

QUOTE You get a quote and timeline

AUDIT We start the audit

REPORT We privately send the report to your team

FIXES Your team fixes the issues

PUBLISH We examine your fixes, update and publish the report (optional)

OUR MOST POPULAR AUDIT REPORTS

Augur Core Audit

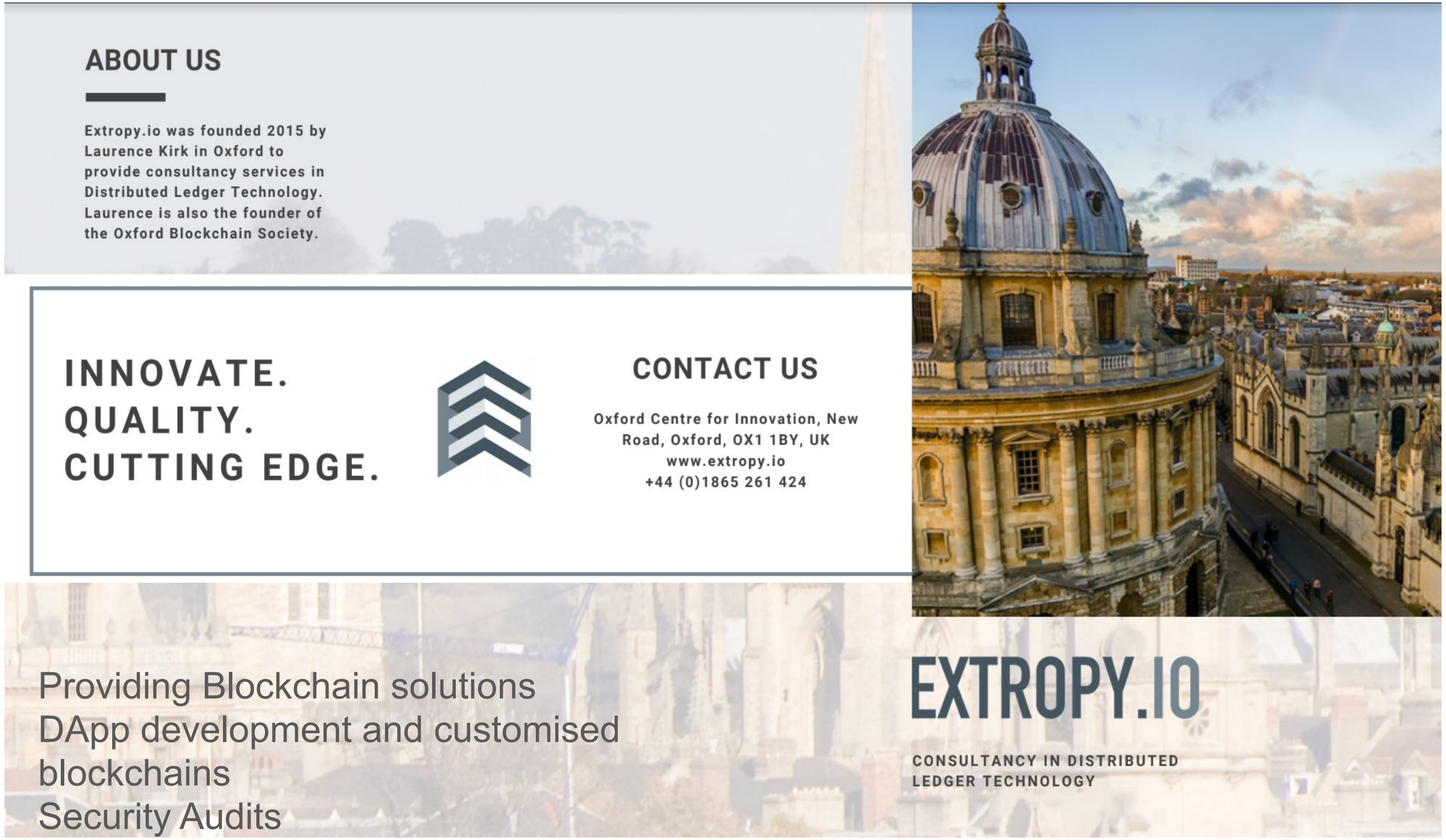
Solidity Compiler Audit

Basic Attention Token (BAT) Audit

Compound Audit

Critical Vulnerability

Centre Token Minting Contracts Audit



ABOUT US

Extropy.io was founded 2015 by Laurence Kirk in Oxford to provide consultancy services in Distributed Ledger Technology. Laurence is also the founder of the Oxford Blockchain Society.

INNOVATE.
QUALITY.
CUTTING EDGE.

CONTACT US

Oxford Centre for Innovation, New Road, Oxford, OX1 1BY, UK
www.extropy.io
+44 (0)1865 261 424

Providing Blockchain solutions
DApp development and customised blockchains
Security Audits

EXTROPY.IO

CONSULTANCY IN DISTRIBUTED LEDGER TECHNOLOGY

Free Developer Workshops

- Basic
- Enterprise
- Advanced EVM
- Zero Knowledge Proofs

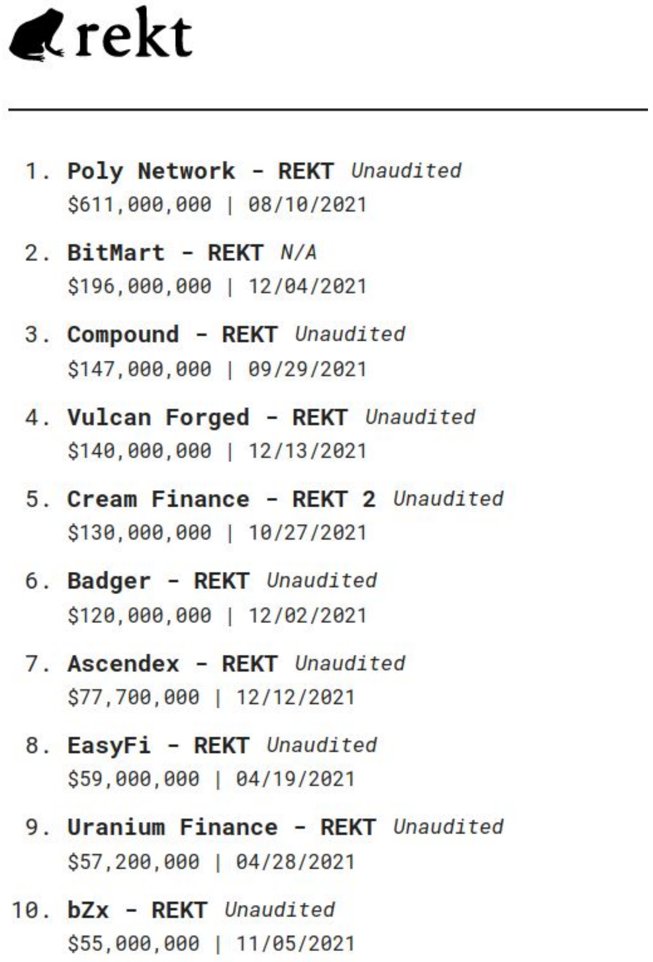
Business Workshops

Website : https://extropy.io

Email : info@extropy.io

Twitter : @extropy

EXTROPY.IO



rekt

1. **Poly Network - REKT** *Unaudited*
   $611,000,000 | 08/10/2021
2. **BitMart - REKT** *N/A*
   $196,000,000 | 12/04/2021
3. **Compound - REKT** *Unaudited*
   $147,000,000 | 09/29/2021
4. **Vulcan Forged - REKT** *Unaudited*
   $140,000,000 | 12/13/2021
5. **Cream Finance - REKT 2** *Unaudited*
   $130,000,000 | 10/27/2021
6. **Badger - REKT** *Unaudited*
   $120,000,000 | 12/02/2021
7. **Ascendex - REKT** *Unaudited*
   $77,700,000 | 12/12/2021
8. **EasyFi - REKT** *Unaudited*
   $59,000,000 | 04/19/2021
9. **Uranium Finance - REKT** *Unaudited*
   $57,200,000 | 04/28/2021
10. **bZx - REKT** *Unaudited*
    $55,000,000 | 11/05/2021

Deployment / Monitoring

Tools

- Hardhat / Truffle scripts
- Etherscan contract verification
- Tenderley / OZ Defender

References

https://docs.soliditylang.org/en/v0.8.7/natspec-format.html

https://jeancvllr.medium.com/solidity-tutorial-all-about-comments-bc31c729975a

https://www.bitdegree.org/learn/solidity-syntax#natspec

https://github.com/aragon/radspec

Slither

Dodoc

Tenderly

Defender