

---

# Mythril Documentation

*Release v0.23.9*

**ConsenSys Dilligence**

**Sep 05, 2022**



---

## Table of Contents:

---

<b>1</b>	<b>What is Mythril?</b>	<b>1</b>
<b>2</b>	<b>Installation and Setup</b>	<b>3</b>
<b>3</b>	<b>Tutorial</b>	<b>5</b>
<b>4</b>	<b>Security Analysis</b>	<b>17</b>
<b>5</b>	<b>Analysis Modules</b>	<b>19</b>
<b>6</b>	<b>mythril package</b>	<b>21</b>
<b>7</b>	<b>Indices and Tables</b>	<b>109</b>
	<b>Python Module Index</b>	<b>111</b>
	<b>Index</b>	<b>115</b>



# CHAPTER 1

---

## What is Mythril?

---

Mythril is a security analysis tool for Ethereum smart contracts. It was [introduced at HITBSecConf 2018](#).

Mythril detects a range of security issues, including integer underflows, owner-overwrite-to-Ether-withdrawal, and others. Note that Mythril is targeted at finding common vulnerabilities, and is not able to discover issues in the business logic of an application. Furthermore, Mythril and symbolic executors are generally unsound, as they are often unable to explore all possible states of a program.



---

## Installation and Setup

---

Mythril can be setup using different methods.

### 2.1 PyPI on Mac OS

```
brew update
brew upgrade
brew tap ethereum/ethereum
brew install solidity
pip3 install mythril
```

### 2.2 PyPI on Ubuntu

```
# Update
sudo apt update

# Install solc
sudo apt install software-properties-common
sudo add-apt-repository ppa:ethereum/ethereum
sudo apt install solc

# Install libssl-dev, python3-dev, and python3-pip
sudo apt install libssl-dev python3-dev python3-pip

# Install mythril
pip3 install mythril
myth version
```

## 2.3 Docker

All Mythril releases, starting from v0.18.3, are published to DockerHub as Docker images under the `mythril/myth` name.

After installing [Docker CE](#):

```
# Pull the latest release of mythril/myth
$ docker pull mythril/myth
```

Use `docker run mythril/myth` the same way you would use the `myth` command

```
docker run mythril/myth --help
docker run mythril/myth disassemble -c "0x6060"
```

To pass a file from your host machine to the dockerized Mythril, you must mount its containing folder to the container properly. For `contract.sol` in the current working directory, do:

```
docker run -v $(pwd):/tmp mythril/myth analyze /tmp/contract.sol
```



### 3.1 Executing Mythril on Simple Contracts

We consider a contract simple if it does not have any imports, like the following contract:

```
contract Exceptions {  
  
    uint256[8] myarray;  
    uint counter = 0;  
    function assert1() public pure {  
        uint256 i = 1;  
        assert(i == 0);  
    }  
    function counter_increase() public {  
        counter+=1;  
    }  
    function assert5(uint input_x) public view{  
        require(counter>2);  
        assert(input_x > 10);  
    }  
    function assert2() public pure {  
        uint256 i = 1;  
        assert(i > 0);  
    }  
  
    function assert3(uint256 input) public pure {  
        assert(input != 23);  
    }  
  
    function require_is_fine(uint256 input) public pure {  
        require(input != 23);  
    }  
  
    function this_is_fine(uint256 input) public pure {
```

(continues on next page)

(continued from previous page)

```

        if (input > 0) {
            uint256 i = 1/input;
        }
    }

    function this_is_find_2(uint256 index) public view {
        if (index < 8) {
            uint256 i = myarray[index];
        }
    }
}

```

We can execute such a contract by directly using the following command:

```
$ myth analyze <file_path>
```

This execution can give the following output:

```

==== Exception State ====
SWC ID: 110
Severity: Medium
Contract: Exceptions
Function name: assert1()
PC address: 708
Estimated Gas Usage: 207 - 492
An assertion violation was triggered.
It is possible to trigger an assertion violation. Note that Solidity
↳assert() statements should only be used to check invariants. Review the
↳transaction trace generated for this issue and either make sure your
↳program logic is correct, or use require() instead of assert() if your
↳goal is to constrain user inputs or enforce preconditions. Remember to
↳validate inputs from both callers (for instance, via passed arguments) and
↳callees (for instance, via return values).
-----
In file: solidity_examples/exceptions.sol:7

assert(i == 0)

-----
Initial State:

Account: [CREATOR], balance: 0x2, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], calldata: , value: 0x0
Caller: [ATTACKER], function: assert1(), txdata: 0xb34c3610, value: 0x0

==== Exception State ====
SWC ID: 110
Severity: Medium
Contract: Exceptions
Function name: assert3(uint256)
PC address: 708
Estimated Gas Usage: 482 - 767

```

(continues on next page)

[illegible]

```
$ myth_analyze <file_path> -t 4
```

```
==== Exception State ====
SWC ID: 110
Severity: Medium
Contract: Exceptions
Function name: assert1()
PC address: 731
Estimated Gas Usage: 207 - 492
An assertion violation was triggered.
It is possible to trigger an assertion violation. Note that Solidity_
→assert() statements should only be used to check invariants. Review the_
→transaction trace generated for this issue and either make sure your_
→program logic is correct, or use require() instead of assert() if your_
→goal is to constrain user inputs or enforce preconditions. Remember to_
→validate inputs from both callers (for instance, via passed arguments) and_
→callees (for instance, via return values).
-----
In file: solidity_examples/exceptions.sol:7

assert(i == 0)

-----
Initial State:
```

### 3.1. Executing Mythril on Simple Contracts 7

(continued from previous page)

```

Account: [CREATOR], balance: 0x2, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], calldata: , value: 0x0
Caller: [ATTACKER], function: assert1(), txdata: 0xb34c3610, value: 0x0

==== Exception State ====
SWC ID: 110
Severity: Medium
Contract: Exceptions
Function name: assert3(uint256)
PC address: 731
Estimated Gas Usage: 504 - 789
An assertion violation was triggered.
It is possible to trigger an assertion violation. Note that Solidity
↳assert() statements should only be used to check invariants. Review the
↳transaction trace generated for this issue and either make sure your
↳program logic is correct, or use require() instead of assert() if your
↳goal is to constrain user inputs or enforce preconditions. Remember to
↳validate inputs from both callers (for instance, via passed arguments) and
↳callees (for instance, via return values).
-----
In file: solidity_examples/exceptions.sol:22

assert(input != 23)

-----
Initial State:

Account: [CREATOR], balance: 0x3, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], calldata: , value: 0x0
Caller: [ATTACKER], function: assert3(uint256), txdata:
↳0x546455b50000000000000000000000000000000000000000000000000000000017,
↳ value: 0x0

==== Exception State ====
SWC ID: 110
Severity: Medium
Contract: Exceptions
Function name: assert5(uint256)
PC address: 731
Estimated Gas Usage: 1302 - 1587
An assertion violation was triggered.
It is possible to trigger an assertion violation. Note that Solidity
↳assert() statements should only be used to check invariants. Review the
↳transaction trace generated for this issue and either make sure your
↳program logic is correct, or use require() instead of assert() if your
↳goal is to constrain user inputs or enforce preconditions. Remember to
↳validate inputs from both callers (for instance, via passed arguments) and
↳callees (for instance, via return values).

```

(continues on next page)

[illegible]

(continued from previous page)

```

"swc-id": "110",
"title": "Exception State",
"tx_sequence": {
  "initialState": {
    "accounts": {
      "0xaffeaffeaffeaffeaffeaffeaffeaffeaffeaffe": {
        "balance": "0x2",
        "code": "",
        "nonce": 0,
        "storage": "{}"
      },
      "0xdeadbeefdeadbeefdeadbeefdeadbeefdeadbeef": {
        "balance": "0x0",
        "code": "",
        "nonce": 0,
        "storage": "{}"
      }
    }
  },
  "steps": [{
    "address": "",
    "calldata": "",
    "input":
      "0x6080604052600060085534801561001557600080fd5b506103f7806100256000396000f3fe60806040523480156",
    "name": "unknown",
    "origin": "0xaffeaffeaffeaffeaffeaffeaffeaffeaffeaffe",
    "value": "0x0"
  }, {
    "address": "0x901d12ebe1b195e5aa8748e62bd7734ae19b51f",
    "calldata": "0xb34c3610",
    "input": "0xb34c3610",
    "name": "assert1()",
    "origin": "0xdeadbeefdeadbeefdeadbeefdeadbeefdeadbeef",
    "resolved_input": null,
    "value": "0x0"
  }
]}
}, {
  "address": 731,
  "code": "assert(input != 23)",
  "contract": "Exceptions",
  "description": "An assertion violation was triggered.\nIt is
possible to trigger an assertion violation. Note that Solidity assert()
statements should only be used to check invariants. Review the transaction
trace generated for this issue and either make sure your program logic is
correct, or use require() instead of assert() if your goal is to constrain
user inputs or enforce preconditions. Remember to validate inputs from
both callers (for instance, via passed arguments) and callees (for
instance, via return values).",
  "filename": "solidity_examples/exceptions.sol",
  "function": "assert3(uint256)",
  "lineno": 22,
  "max_gas_used": 789,
  "min_gas_used": 504,
  "severity": "Medium",
  "sourceMap": ":::i",

```

(continues on next page)

### 3.1. Executing Mythril on Simple Contracts 11

(continued from previous page)

```

    "max_gas_used": 1587,
    "min_gas_used": 1302,
    "severity": "Medium",
    "sourceMap": "::-i",
    "swc-id": "110",
    "title": "Exception State",
    "tx_sequence": {
      "initialState": {
        "accounts": {
          "0xaaffeaffeaffeaffeaffeaffeaffeaffeaffeaffe": {
            "balance": "0x0",
            "code": "",
            "nonce": 0,
            "storage": "{}"
          },
          "0xdeadbeefdeadbeefdeadbeefdeadbeefdeadbeef": {
            "balance": "0x0",
            "code": "",
            "nonce": 0,
            "storage": "{}"
          }
        }
      },
      "steps": [{
        "address": "",
        "calldata": "",
        "input":
→ "0x6080604052600060085534801561001557600080fd5b506103f7806100256000396000f3fe60806040523480156
→ ",
        "name": "unknown",
        "origin": "0xaaffeaffeaffeaffeaffeaffeaffeaffeaffeaffe",
        "value": "0x0"
      }, {
        "address": "0x901d12ebe1b195e5aa8748e62bd7734ae19b51f",
        "calldata": "0xe47b0253",
        "input": "0xe47b0253",
        "name": "counter_increase()",
        "origin": "0xaaffeaffeaffeaffeaffeaffeaffeaffeaffeaffe",
        "resolved_input": null,
        "value": "0x0"
      }, {
        "address": "0x901d12ebe1b195e5aa8748e62bd7734ae19b51f",
        "calldata": "0xe47b0253",
        "input": "0xe47b0253",
        "name": "counter_increase()",
        "origin": "0xdeadbeefdeadbeefdeadbeefdeadbeefdeadbeef",
        "resolved_input": null,
        "value": "0x0"
      }, {
        "address": "0x901d12ebe1b195e5aa8748e62bd7734ae19b51f",
        "calldata": "0xe47b0253",
        "input": "0xe47b0253",
        "name": "counter_increase()",
        "origin": "0xaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa",
        "resolved_input": null,
        "value": "0x0"
      }, {

```

(continues on next page)



[illegible]

There are interesting options such as `--execution-timeout <seconds>` and `--solver-timeout <milliseconds>` which can be increased for better results. The default execution-timeout and solver-timeout are 86400 seconds and 25000 milliseconds respectively.

(continues on next page)

(continued from previous page)

```

        return false;
    } else {
        return keccak256(bytes(a)) == keccak256(bytes(b));
    }
}

function nothing(string memory g_0, bytes3 g_1, bytes5 g_2, bytes5 g_3,
↳bytes3 g_4, bytes3 g_5, bytes3 g_6, bytes3 g_7, bytes3 g_8, bytes3 g_9,
↳bytes3 g_10, bytes3 g_11) public view returns (bool){
    if (!stringCompare(g_0, x_0)) return false;

    if (g_1 != x_1) return false;

    if (g_2 != x_2) return false;

    if (g_3 != x_3) return false;

    if (g_4 != x_4) return false;

    if (g_5 != x_5) return false;

    if (g_6 != x_6) return false;

    if (g_7 != x_7) return false;

    if (g_8 != x_8) return false;

    if (g_9 != x_9) return false;

    if (g_10 != x_9) return false;

    if (g_11 != x_9) return false;

    return true;
}
}

```

When this contract is directly executed, by using the following command:

```
$ myth analyze <file_path>
```

We encounter the following error:

```

mythril.interfaces.cli [ERROR]: Solc experienced a fatal error.

ParserError: Source "@openzeppelin/contracts/token/ERC20/ERC20.sol" not
↳found: File not found. Searched the following locations: "".
--> <file_path>:1:1:
|
1 | import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
| ~~~~~

```

This is because Mythril uses Solidity to compile the program, to circumvent this issue we can use the following solc-json file:

```
{  
  "remappings": [ "@openzeppelin/contracts/token/ERC20/=node_modules/ERC20" ],  
}
```

Here we are mapping the import `@openzeppelin/contracts/token/ERC20/` to the path which contains `ERC20.sol` which in this case is `node_modules/ERC20`. This instructs the compiler to search for anything with the prefix `@openzeppelin/contracts/token/ERC20/` `` in the path `` `node_modules/ERC20` in our file system. We feed to file to Mythril using `--solc-json` argument.

```
$ myth analyze {file_path} --solc-json {json_file_path}
```

This can effectively execute the file since the Solidity compiler can locate *ERC20.sol*. For more information on remappings, you can refer to [Solc docs](#).



Run `myth analyze` with one of the input options described below will run the analysis modules in the `/analysis/modules` directory.

## 4.1 Analyzing Solidity Code

In order to work with Solidity source code files, the `solc command line compiler` needs to be installed and in `PATH`. You can then provide the source file(s) as positional arguments.

```
$ myth analyze ether_send.sol
==== Unprotected Ether Withdrawal ====
SWC ID: 105
Severity: High
Contract: Crowdfunding
Function name: withdrawfunds()
PC address: 730
Estimated Gas Usage: 1132 - 1743
Anyone can withdraw ETH from the contract account.
Arbitrary senders other than the contract creator can withdraw ETH from the contract_
↳account without previously having sent an equivalent amount of ETH to it. This is_
↳likely to be a vulnerability.
-----
In file: tests/testdata/input_contracts/ether_send.sol:21

msg.sender.transfer(address(this).balance)

-----
```

If an input file contains multiple contract definitions, Mythril analyzes the *last* bytecode output produced by `solc`. You can override this by specifying the contract name explicitly:

```
myth analyze OmiseGo.sol:OMGToken
```

### 4.1.1 Specifying Solc Versions

You can specify a version of the solidity compiler to be used with `--solc <version number>`. Please be aware that this uses `py-solc` and will only work on Linux and macOS. It will check the version of solc in your path, and if this is not what is specified, it will download binaries on Linux or try to compile from source on macOS.

### 4.1.2 Output Formats

By default, analysis results are printed to the terminal in text format. You can change the output format with the `-o` argument:

```
myth analyze underflow.sol -o jsonv2
```

Available formats are `text`, `markdown`, `json`, and `jsonv2`. For integration with other tools, `jsonv2` is generally preferred over `json` because it is consistent with other [MythX](#) tools.

## 4.2 Analyzing On-Chain Contracts

When analyzing contracts on the blockchain, Mythril will by default attempt to query INFURA. You can use the built-in INFURA support or manually configure the RPC settings with the `--rpc` argument.

<code>--rpc ganache</code>	Connect to local Ganache
<code>--rpc infura-[netname] --infura-id &lt;ID&gt;</code>	Connect to mainnet, rinkeby, kovan, or ropsten.
<code>--rpc host:port</code>	Connect to custom rpc
<code>--rpctls &lt;True/False&gt;</code>	RPC connection over TLS (default: False)

To specify a contract address, use `-a <address>`

Analyze mainnet contract via INFURA:

```
myth analyze -a 0x5c436ff914c458983414019195e0f4ecbef9e6dd --infura-id <ID>
```

You can also use the environment variable `INFURA_ID` instead of the cmd line argument or set it in `~/.mythril/config.ini`.

```
myth -v4 analyze -a 0xEbFD99838cb0c132016B9E117563CB41f2B02264 --infura-id <ID>
```

### 4.3 Speed vs. Coverage

The execution timeout can be specified with the `--execution-timeout <seconds>` argument. When the timeout is reached, mythril will stop analysis and print out all currently found issues.

The maximum recursion depth for the symbolic execution engine can be controlled with the `--max-depth` argument. The default value is 22. Lowering this value will decrease the number of explored states and analysis time, while increasing this number will increase the number of explored states and increase analysis time. For some contracts, it helps to fine tune this number to get the best analysis results. -

Mythril's detection capabilities are written in modules in the `/analysis/module/modules` directory.

### 5.1 Modules

#### 5.1.1 Delegate Call To Untrusted Contract

The `delegatecall` module detects SWC-112 (DELEGATECALL to Untrusted Callee).

#### 5.1.2 Dependence on Predictable Variables

The `predictable variables` module detects SWC-120 (Weak Randomness) and SWC-116 (Timestamp Dependence).

#### 5.1.3 Ether Thief

The `Ether Thief` module detects SWC-105 (Unprotected Ether Withdrawal).

#### 5.1.4 Exceptions

The `exceptions` module detects SWC-110 (Assert Violation).

#### 5.1.5 External Calls

The `external calls` module warns about SWC-107 (Reentrancy) by detecting calls to external contracts.

### 5.1.6 Integer

The `integer` module detects SWC-101 (Integer Overflow and Underflow).

### 5.1.7 Multiple Sends

The `multiple sends` module detects SWC-113 (Denial of Service with Failed Call) by checking for multiple calls or sends in a single transaction.

### 5.1.8 Suicide

The `suicide` module detects SWC-106 (Unprotected SELFDESTRUCT).

### 5.1.9 State Change External Calls

The `state change external calls` module detects SWC-107 (Reentrancy) by detecting state change after calls to an external contract.

### 5.1.10 Unchecked Retval

The `unchecked retval` module detects SWC-104 (Unchecked Call Return Value).

### 5.1.11 User Supplied assertion

The `user supplied assertion` module detects SWC-110 (Assert Violation) for user-supplied assertions. User supplied assertions should be log messages of the form: `emit AssertionFailed(string)`.

### 5.1.12 Arbitrary Storage Write

The `arbitrary storage write` module detects SWC-124 (Write to Arbitrary Storage Location).

### 5.1.13 Arbitrary Jump

The `arbitrary jump` module detects SWC-127 (Arbitrary Jump with Function Type Variable).

## 5.2 Creating a Module

Create a module in the `analysis/modules` directory, and create an instance of a class that inherits `DetectionModule` named `detector`. Take a look at the `suicide` module as an example.



## 6.1 Subpackages

### 6.1.1 mythrill.analysis package

#### Subpackages

mythrill.analysis.module package

#### Subpackages

mythrill.analysis.module.modules package

#### Submodules

mythrill.analysis.module.modules.arbitrary\_jump module

This module contains the detection code for Arbitrary jumps.

```
class mythrill.analysis.module.modules.arbitrary_jump.ArbitraryJump
```

```
    Bases: mythrill.analysis.module.base.DetectionModule
```

```
    This module searches for JUMPs to a user-specified location.
```

```
    description = '\n\nSearch for jumps to arbitrary locations in the bytecode\n'
```

```
    entry_point = 2
```

```
    name = 'Caller can redirect execution to arbitrary bytecode locations'
```

```
    pre_hooks = ['JUMP', 'JUMPI']
```

```
    reset_module()
```

```
        Resets the module by clearing everything :return:
```

```
swc_id = '127'
```

```
mythril.analysis.module.modules.arbitrary_jump.is_unique_jumpdest (jump_dest:  
    mythril.laser.smt.bitvec.BitVec,  
    state:  
    mythril.laser.ethereum.state.global_  
    → bool
```

Handles cases where jump\_dest evaluates to a single concrete value

### mythril.analysis.module.modules.arbitrary\_write module

This module contains the detection code for arbitrary storage write.

```
class mythril.analysis.module.modules.arbitrary_write.ArbitraryStorage  
    Bases: mythril.analysis.module.base.DetectionModule  
    This module searches for a feasible write to an arbitrary storage slot.  
    description = '\n\nSearch for any writes to an arbitrary storage slot\n'  
    entry_point = 2  
    name = 'Caller can write to arbitrary storage locations'  
    pre_hooks = ['SSTORE']  
    reset_module()  
        Resets the module by clearing everything :return:  
    swc_id = '124'
```

### mythril.analysis.module.modules.delegatecall module

This module contains the detection code for insecure delegate call usage.

```
class mythril.analysis.module.modules.delegatecall.ArbitraryDelegateCall  
    Bases: mythril.analysis.module.base.DetectionModule  
    This module detects delegatecall to a user-supplied address.  
    description = 'Check for invocations of delegatecall to a user-supplied address.'  
    entry_point = 2  
    name = 'Delegatecall to a user-specified address'  
    pre_hooks = ['DELEGATECALL']  
    swc_id = '112'
```

### mythril.analysis.module.modules.dependence\_on\_origin module

This module contains the detection code for predictable variable dependence.

```
class mythril.analysis.module.modules.dependence_on_origin.TxOrigin  
    Bases: mythril.analysis.module.base.DetectionModule  
    This module detects whether control flow decisions are made based on the transaction origin.  
    description = 'Check whether control flow decisions are influenced by tx.origin'
```

```
entry_point = 2
name = 'Control flow depends on tx.origin'
post_hooks = ['ORIGIN']
pre_hooks = ['JUMPI']
swc_id = '115'

class mythrill.analysis.module.modules.dependence_on_origin.TxOriginAnnotation
  Bases: object

  Symbol annotation added to a variable that is initialized with a call to the ORIGIN instruction.
```

**mythril.analysis.module.modules.dependence\_on\_predictable\_vars module**

This module contains the detection code for predictable variable dependence.

```
class mythrill.analysis.module.modules.dependence_on_predictable_vars.OldBlockNumberUsedAnnotation
    Bases: mythrill.laser.ethereum.state.annotation.StateAnnotation

    Symbol annotation used if a variable is initialized from a predictable environment variable.


class mythrill.analysis.module.modules.dependence_on_predictable_vars.PredictableValueAnnotation
    Bases: object

    Symbol annotation used if a variable is initialized from a predictable environment variable.


class mythrill.analysis.module.modules.dependence_on_predictable_vars.PredictableVariablesModule
    Bases: mythrill.analysis.module.base.DetectionModule

    This module detects whether control flow decisions are made using predictable parameters.

    description = 'Check whether control flow decisions are influenced by block.coinbase,block.timestamp,block.number,block.gaslimit'
    entry_point = 2
    name = 'Control flow depends on a predictable environment variable'
    post_hooks = ['BLOCKHASH', 'COINBASE', 'GASLIMIT', 'TIMESTAMP', 'NUMBER']
    pre_hooks = ['JUMPI', 'BLOCKHASH']
    swc id = '116 120'
```

**mythril.analysis.module.modules.ether\_thief module**

This module contains the detection code for unauthorized ether withdrawal.

```
class mythril.analysis.module.modules.ether_thief.EtherThief
    Bases: mythril.analysis.module.base.DetectionModule

    This module search for cases where Ether can be withdrawn to a user- specified address.

    description = '\nSearch for cases where Ether can be withdrawn to a user-specified address'
    entry_point = 2
    name = 'Any sender can withdraw ETH from the contract account'
    post_hooks = ['CALL', 'STATICCALL']
```

```
reset_module()
    Resets the module by clearing everything :return:

swc_id = '105'
```

### mythril.analysis.module.modules.exceptions module

This module contains the detection code for reachable exceptions.

```
class mythril.analysis.module.modules.exceptions.Exceptions
    Bases: mythril.analysis.module.base.DetectionModule

    description = 'Checks whether any exception states are reachable.'
    entry_point = 2
    name = 'Assertion violation'
    pre_hooks = ['INVALID', 'JUMP', 'REVERT']
    swc_id = '110'

class mythril.analysis.module.modules.exceptions.LastJumpAnnotation(last_jump:
                                                                    Op-
                                                                    tional[int]
                                                                    = None)
    Bases: mythril.laser.ethereum.state.annotation.StateAnnotation
    State Annotation used if an overflow is both possible and used in the annotated path

mythril.analysis.module.modules.exceptions.is_assertion_failure(global_state)
```

### mythril.analysis.module.modules.external\_calls module

This module contains the detection code for potentially insecure low-level calls.

```
class mythril.analysis.module.modules.external_calls.ExternalCalls
    Bases: mythril.analysis.module.base.DetectionModule

    This module searches for low level calls (e.g. call.value()) that forward all gas to the callee.

    description = '\n\nSearch for external calls with unrestricted gas to a user-specified'
    entry_point = 2
    name = 'External call to another contract'
    pre_hooks = ['CALL']
    swc_id = '107'
```

### mythril.analysis.module.modules.integer module

This module contains the detection code for integer overflows and underflows.

```
class mythril.analysis.module.modules.integer.IntegerArithmetics
    Bases: mythril.analysis.module.base.DetectionModule

    This module searches for integer over- and underflows.

    description = "For every SUB instruction, check if there's a possible state where op1"
```

```

    entry_point = 2
    name = 'Integer overflow or underflow'
    pre_hooks = ['ADD', 'MUL', 'EXP', 'SUB', 'SSTORE', 'JUMPI', 'STOP', 'RETURN', 'CALL']
    reset_module()
        Resets the module :return:
    swc_id = '101'

class mythril.analysis.module.modules.integer.OverUnderflowAnnotation(overflowing_state:
                                                                    mythril.laser.ethereum.state.global_state.GlobalState,
                                                                    operator:
                                                                    str,
                                                                    constraint:
                                                                    mythril.laser.smt.bool.Bool)

    Bases: object

    Symbol Annotation used if a BitVector can overflow

class mythril.analysis.module.modules.integer.OverUnderflowStateAnnotation
    Bases: mythril.laser.ethereum.state.annotation.StateAnnotation

    State Annotation used if an overflow is both possible and used in the annotated path

```

### mythril.analysis.module.modules.multiple\_sends module

This module contains the detection code to find multiple sends occurring in a single transaction.

```

class mythril.analysis.module.modules.multiple_sends.MultipleSends
    Bases: mythril.analysis.module.base.DetectionModule

    This module checks for multiple sends in a single transaction.

    description = 'Check for multiple sends in a single transaction'
    entry_point = 2
    name = 'Multiple external calls in the same transaction'
    pre_hooks = ['CALL', 'DELEGATECALL', 'STATICCALL', 'CALLCODE', 'RETURN', 'STOP']
    swc_id = '113'

class mythril.analysis.module.modules.multiple_sends.MultipleSendsAnnotation
    Bases: mythril.laser.ethereum.state.annotation.StateAnnotation

```

### mythril.analysis.module.modules.state\_change\_external\_calls module

```

class mythril.analysis.module.modules.state_change_external_calls.StateChangeAfterCall
    Bases: mythril.analysis.module.base.DetectionModule

    This module searches for state change after low level calls (e.g. call.value()) that forward gas to the callee.

    description = '\n\nCheck whether the account state is accesses after the execution of a call'
    entry_point = 2
    name = 'State change after an external call'

```

```
pre_hooks = ['CALL', 'DELEGATECALL', 'CALLCODE', 'SSTORE', 'SLOAD', 'CREATE', 'CREATE2', 'CALLSTOCK']
swc_id = '107'

class mythril.analysis.module.modules.state_change_external_calls.StateChangeCallsAnnotation:
    Bases: mythril.laser.ethereum.state.annotation.StateAnnotation
    get_issue(global_state: mythril.laser.ethereum.state.global_state.GlobalState, detector: mythril.analysis.module.base.DetectionModule) → Optional[mythril.analysis.potential_issues.PotentialIssue]
```

### mythril.analysis.module.modules.suicide module

```
class mythril.analysis.module.modules.suicide.AccidentallyKillable:
    Bases: mythril.analysis.module.base.DetectionModule
    This module checks if the contract can be ‘accidentally’ killed by anyone.
    description = "\nCheck if the contract can be 'accidentally' killed by anyone.\nFor kill"
    entry_point = 2
    name = 'Contract can be accidentally killed by anyone'
    pre_hooks = ['SELFDESTRUCT']
    reset_module()
        Resets the module :return:
    swc_id = '106'
```

### mythril.analysis.module.modules.unchecked\_retval module

This module contains detection code to find occurrences of calls whose return value remains unchecked.

```
class mythril.analysis.module.modules.unchecked_retval.RetVal:
    Bases: dict

class mythril.analysis.module.modules.unchecked_retval.UncheckedRetVal:
    Bases: mythril.analysis.module.base.DetectionModule
    A detection module to test whether CALL return value is checked.
    description = 'Test whether CALL return value is checked. For direct calls, the Solidity'
    entry_point = 2
    name = 'Return value of an external call is not checked'
    post_hooks = ['CALL', 'DELEGATECALL', 'STATICCALL', 'CALLCODE']
    pre_hooks = ['STOP', 'RETURN']
    swc_id = '104'

class mythril.analysis.module.modules.unchecked_retval.UncheckedRetValAnnotation:
    Bases: mythril.laser.ethereum.state.annotation.StateAnnotation
```

## mythril.analysis.module.modules.user\_assertions module

This module contains the detection code for potentially insecure low-level calls.

```
class mythril.analysis.module.modules.user_assertions.UserAssertions
```

Bases: *mythril.analysis.module.base.DetectionModule*

This module searches for user supplied exceptions: emit AssertionFailed("Error").

```
description = "\n\nSearch for reachable user-supplied exceptions.\nReport a warning if
```

```
entry_point = 2
```

```
name = 'A user-defined assertion has been triggered'
```

```
pre_hooks = ['LOG1', 'MSTORE']
```

```
swc_id = '110'
```

## Module contents

### Submodules

## mythril.analysis.module.base module

Mythril Detection Modules

This module includes an definition of the DetectionModule interface. DetectionModules implement different analysis rules to find weaknesses and vulnerabilities.

```
class mythril.analysis.module.base.DetectionModule
```

Bases: abc.ABC

The base detection module.

All custom-built detection modules must inherit from this class.

There are several class properties that expose information about the detection modules

### Parameters

- **name** – The name of the detection module
- **swc\_id** – The SWC ID associated with the weakness that the module detects
- **description** – A description of the detection module, and what it detects
- **entry\_point** – Mythril can run callback style detection modules, or modules that search the statespace. [IMPORTANT] POST entry points severely slow down the analysis, try to always use callback style modules
- **pre\_hooks** – A list of instructions to hook the laser vm for (pre execution of the instruction)
- **post\_hooks** – A list of instructions to hook the laser vm for (post execution of the instruction)

```
description = 'Detection module description'
```

```
entry_point = 2
```

**execute** (*target:* *mythril.laser.ethereum.state.global\_state.GlobalState*) → **Optional**[List[mythril.analysis.report.Issue]]  
The entry point for execution, which is being called by Mythril.

**Parameters** **target** – The target of the analysis, either a global state (callback) or the entire statespace (post)

**Returns** List of encountered issues

**name** = 'Detection Module Name / Title'

**post\_hooks** = []

**pre\_hooks** = []

**reset\_module** ()

Resets the storage of this module

**swc\_id** = 'SWC-000'

**update\_cache** (*issues=None*)

Updates cache with param issues, updates against self.issues, if the param is None :param issues: The issues used to update the cache

**class** mythril.analysis.module.base.**EntryPoint**

Bases: enum.Enum

EntryPoint Enum

This enum is used to signify the entry\_point of detection modules. See also the class documentation of DetectionModule

**CALLBACK** = 2

**POST** = 1

## mythril.analysis.module.loader module

**class** mythril.analysis.module.loader.**ModuleLoader**

Bases: object

The module loader class implements a singleton loader for detection modules.

By default it will load the detection modules in the mythril package. Additional detection modules can be loaded using the register\_module function call implemented by the ModuleLoader

**get\_detection\_modules** (*entry\_point:* *Optional*[mythril.analysis.module.base.EntryPoint]  
= *None*, *white\_list:* *Optional*[List[str]] = *None*) →  
List[mythril.analysis.module.base.DetectionModule]

Gets registered detection modules

**Parameters**

- **entry\_point** – If specified: only return detection modules with this entry point
- **white\_list** – If specified: only return whitelisted detection modules

**Returns** The selected detection modules

**register\_module** (*detection\_module:* mythril.analysis.module.base.DetectionModule)

Registers a detection module with the module loader



## mythril.analysis.module.module\_helpers module

`mythril.analysis.module.module_helpers.is_prehook()` → bool

Check if we are in prehook. One of Bernhard's trademark hacks! Let's leave it to this for now, unless we need to check prehook for a lot more modules.

## mythril.analysis.module.util module

`mythril.analysis.module.util.get_detection_module_hooks` (*modules:*  
*List[mythril.analysis.module.base.DetectionModule]*  
*hook\_type='pre')* →  
*Dict[str, List[Callable]]*

Gets a dictionary with the hooks for the passed detection modules

### Parameters

- **modules** – The modules for which to retrieve hooks
- **hook\_type** – The type of hooks to retrieve (default: “pre”)

**Returns** Dictionary with discovered hooks

`mythril.analysis.module.util.reset_callback_modules` (*module\_names:*  
*Optional[List[str]] = None)* *Op-*

Clean the issue records of every callback-based module.

## Module contents

### Submodules

## mythril.analysis.analysis\_args module

## mythril.analysis.call\_helpers module

This module provides helper functions for the analysis modules to deal with call functionality.

`mythril.analysis.call_helpers.get_call_from_state` (*state:*  
*mythril.laser.ethereum.state.global\_state.GlobalState)*  
→ *Optional[mythril.analysis.ops.Call]* *Op-*

**Parameters** **state** –

**Returns**

## mythril.analysis.callgraph module

This module contains the configuration and functions to create call graphs.

`mythril.analysis.callgraph.extract_edges` (*statespace*)

**Parameters** **statespace** –

**Returns**

`mythril.analysis.callgraph.extract_nodes` (*statespace*)

**Parameters**

- **statespace** –
- **color\_map** –

**Returns**

`mythril.analysis.callgraph.generate_graph` (*statespace*, *title*='Mythril / Ethereum LASER Symbolic VM', *physics*=False, *phrackify*=False)

**Parameters**

- **statespace** –
- **title** –
- **physics** –
- **phrackify** –

**Returns****mythril.analysis.issue\_annotation module**

**class** `mythril.analysis.issue_annotation.IssueAnnotation` (*conditions*:  
*List*[*mythril.laser.smt.bool.Bool*],  
*issue*:  
*mythril.analysis.report.Issue*,  
*detector*)  
Bases: *mythril.laser.ethereum.state.annotation.StateAnnotation*

**persist\_over\_calls**

If this function returns true then laser will propagate the annotation between calls

The default is set to False

**persist\_to\_world\_state** () → bool

If this function returns true then laser will also annotate the world state.

If you want annotations to persist through different user initiated message call transactions then this should be enabled.

The default is set to False

**mythril.analysis.ops module**

This module contains various helper methods for dealing with EVM operations.

**class** `mythril.analysis.ops.Call` (*node*, *state*, *state\_index*, *\_type*, *to*, *gas*,  
*value*=<*mythril.analysis.ops.Variable object*>, *data*=None)

Bases: *mythril.analysis.ops.Op*

The representation of a CALL operation.

**class** `mythril.analysis.ops.Op` (*node*, *state*, *state\_index*)

Bases: *object*

The base type for operations referencing current node and state.

**class** `mythril.analysis.ops.VarType`

Bases: *enum.Enum*

An enum denoting whether a value is symbolic or concrete.

```
CONCRETE = 2
```

```
SYMBOLIC = 1
```

```
class mythril.analysis.ops.Variable(val, _type)
```

Bases: object

The representation of a variable with value and type.

```
mythril.analysis.ops.get_variable(i)
```

**Parameters** *i* –

**Returns**

## mythril.analysis.potential\_issues module

```
class mythril.analysis.potential_issues.PotentialIssue(contract, function_name,
                                                         address, swc_id, title,
                                                         bytecode, detector,
                                                         severity=None, description_head="",
                                                         description_tail="", constraints=None)
```

Bases: object

Representation of a potential issue

```
class mythril.analysis.potential_issues.PotentialIssuesAnnotation
```

Bases: *mythril.laser.ethereum.state.annotation.StateAnnotation*

**search\_importance**

Used in estimating the priority of a state annotated with the corresponding annotation. Default is 1

```
mythril.analysis.potential_issues.check_potential_issues(state:
                                                         mythril.laser.ethereum.state.global_state.GlobalState)
                                                         → None
```

Called at the end of a transaction, checks potential issues, and adds valid issues to the detector.

**Parameters** *state* – The final global state of a transaction

**Returns**

```
mythril.analysis.potential_issues.get_potential_issues_annotation(state:
                                                                    mythril.laser.ethereum.state.global_state.GlobalState)
                                                                    →
                                                                    mythril.analysis.potential_issues.PotentialIssuesAnnotation
```

Returns the potential issues annotation of the given global state, and creates one if one does not already exist.

**Parameters** *state* – The global state

**Returns**

## mythril.analysis.report module

This module provides classes that make up an issue report.

```
class mythril.analysis.report.Issue(contract: str, function_name: str, address: int, swc_id:
                                     str, title: str, bytecode: str, gas_used=(None, None),
                                     severity=None, description_head="", description_tail="",
                                     transaction_sequence=None, source_location=None)
```

Bases: object

Representation of an issue and its location.

**static add\_block\_data** (*transaction\_sequence: Dict[KT, VT]*)

Adds sane block data to a transaction\_sequence

**add\_code\_info** (*contract*)

Parameters **contract** –

**as\_dict**

Returns

**resolve\_function\_names** ()

Resolves function names for each step

**static resolve\_input** (*data, function\_name*)

Adds decoded calldata to the tx sequence.

**transaction\_sequence\_jsonv2**

Returns the transaction sequence as a json string with pre-generated block data

**transaction\_sequence\_users**

Returns the transaction sequence without pre-generated block data

**class** mythril.analysis.report.**Report** (*contracts=None, exceptions=None, execution\_info: Optional[List[mythril.laser.execution\_info.ExecutionInfo]] = None*)

Bases: object

A report containing the content of multiple issues.

**append\_issue** (*issue*)

Parameters **issue** –

**as\_json** ()

Returns

**as\_markdown** ()

Returns

**as\_swc\_standard\_format** ()

Format defined for integration and correlation.

Returns

**as\_text** ()

Returns

**environment** = <jinja2.environment.Environment object>

**sorted\_issues** ()

Returns

## mythril.analysis.security module

This module contains functionality for hooking in detection modules and executing them.

```
mythril.analysis.security.fire_lasers (statespace, white_list: Optional[List[str]] = None)
                                         → List[mythril.analysis.report.Issue]
```

Fire lasers at analysed statespace object

#### Parameters

- **statespace** – Symbolic statespace to analyze
- **white\_list** – Optionally whitelist modules to use for the analysis

**Returns** Discovered issues

```
mythril.analysis.security.retrieve_callback_issues (white_list: Optional[List[str]] = None) →
                                                         List[mythril.analysis.report.Issue]
```

Get the issues discovered by callback type detection modules

## mythril.analysis.solver module

This module contains analysis module helpers to solve path constraints.

```
mythril.analysis.solver.get_transaction_sequence (global_state:
                                                         mythril.laser.ethereum.state.global_state.GlobalState,
                                                         constraints:
                                                         mythril.laser.ethereum.state.constraints.Constraints)
                                                         → Dict[str, Any]
```

Generate concrete transaction sequence. Note: This function only considers the constraints in constraint argument, which in some cases is expected to differ from global\_state's constraints

#### Parameters

- **global\_state** – GlobalState to generate transaction sequence for
- **constraints** – list of constraints used to generate transaction sequence

```
mythril.analysis.solver.pretty_print_model (model)
```

Pretty prints a z3 model

**Parameters** **model** –

**Returns**

## mythril.analysis.swc\_data module

This module maps SWC IDs to their registry equivalents.

## mythril.analysis.symbolic module

This module contains a wrapper around LASER for extended analysis purposes.

```
class mythril.analysis.symbolic.SymExecWrapper(contract, address: Union[int, str,
mythril.laser.smt.bitvec.BitVec], strategy: str, dynloader=None, max_depth:
int = 22, execution_timeout: Optional[int] = None, loop_bound: int
= 3, create_timeout: Optional[int] = None, transaction_count: int = 2,
modules: Optional[List[str]] = None, compulsory_statespace: bool = True,
disable_dependency_pruning: bool = False, run_analysis_modules: bool =
True, custom_modules_directory: str =
")
```

Bases: object

Wrapper class for the LASER Symbolic virtual machine.

Symbolically executes the code and does a bit of pre-analysis for convenience.

**execution\_info**

## mythril.analysis.traceexplore module

This module provides a function to convert a state space into a set of state nodes and transition edges.

`mythril.analysis.traceexplore.get_serializable_statespace(statespace)`

**Parameters** `statespace` –

**Returns**

## Module contents

### 6.1.2 mythril.concolic package

#### Submodules

#### mythril.concolic.concolic\_execution module

```
mythril.concolic.concolic_execution.concolic_execution(concrete_data:
mythril.concolic.concrete_data.ConcreteData,
jump_addresses: List[T],
solver_timeout=100000)
→ List[Dict[str, Dict[str,
Any]]]
```

Executes codes and prints input required to cover the branch flips :param `input_file`: Input file :param `jump_addresses`: Jump addresses to flip :param `solver_timeout`: Solver timeout

```
mythril.concolic.concolic_execution.flip_branches(init_state:
mythril.laser.ethereum.state.world_state.WorldState,
concrete_data:
mythril.concolic.concrete_data.ConcreteData,
jump_addresses: List[str], trace:
List[T]) → List[Dict[str, Dict[str,
Any]]]
```

Flips branches and prints the input required for branch flip

**Parameters**

- **concrete\_data** – Concrete data
- **jump\_addresses** – Jump addresses to flip
- **trace** – trace to follow

**mythril.concolic.concrete\_data module**

**class** mythril.concolic.concrete\_data.**AccountData**  
Bases: dict

**class** mythril.concolic.concrete\_data.**ConcreteData**  
Bases: dict

**class** mythril.concolic.concrete\_data.**InitialState**  
Bases: dict

**class** mythril.concolic.concrete\_data.**TransactionData**  
Bases: dict

**mythril.concolic.find\_trace module**

mythril.concolic.find\_trace.**concrete\_execution** (*concrete\_data:*  
                                                   mythril.concolic.concrete\_data.*ConcreteData*)  
                                                   → Tu-  
                                                   ple[mythril.laser.ethereum.state.world\_state.WorldState,  
                                                   List[T]]  
 Executes code concretely to find the path to be followed by concolic executor :param concrete\_data: Concrete  
 data :return: path trace

mythril.concolic.find\_trace.**setup\_concrete\_initial\_state** (*concrete\_data:*  
                                                   mythril.concolic.concrete\_data.*ConcreteData*)  
                                                   →  
                                                   mythril.laser.ethereum.state.world\_state.WorldState  
 Sets up concrete initial state :param concrete\_data: Concrete data :return: initialised world state

**Module contents****6.1.3 mythril.disassembler package****Submodules****mythril.disassembler.asm module**

This module contains various helper classes and functions to deal with EVM code disassembly.

**class** mythril.disassembler.asm.**EvmInstruction** (*address, op\_code, argument=None*)  
Bases: object

Model to hold the information of the disassembly.

**to\_dict** () → dict

**Returns**

`mythril.disassembler.asm.disassemble (bytecode) → list`

Disassembles evm bytecode and returns a list of instructions.

**Parameters** `bytecode` –

**Returns**

`mythril.disassembler.asm.find_op_code_sequence (pattern: list, instruction_list: list) →`

`collections.abc.Generator`

Returns all indices in `instruction_list` that point to instruction sequences following a pattern.

**Parameters**

- **pattern** – The pattern to look for, e.g. `[["PUSH1", "PUSH2"], ["EQ"]]` where `[["PUSH1", "EQ"]]` satisfies pattern
- **instruction\_list** – List of instructions to look in

**Returns** Indices to the instruction sequences

`mythril.disassembler.asm.get_opcode_from_name (operation_name: str) → int`

Get an op code based on its name.

**Parameters** `operation_name` –

**Returns**

`mythril.disassembler.asm.instruction_list_to_easm (instruction_list: list) → str`

Convert a list of instructions into an easm op code string.

**Parameters** `instruction_list` –

**Returns**

`mythril.disassembler.asm.is_sequence_match (pattern: list, instruction_list: list, index: int)`

`→ bool`

Checks if the instructions starting at index follow a pattern.

**Parameters**

- **pattern** – List of lists describing a pattern, e.g. `[["PUSH1", "PUSH2"], ["EQ"]]` where `[["PUSH1", "EQ"]]` satisfies pattern
- **instruction\_list** – List of instructions
- **index** – Index to check for

**Returns** Pattern matched

## mythril.disassembler.disassembly module

This module contains the class used to represent disassembly code.

**class** `mythril.disassembler.disassembly.Disassembly (code: str, enable_online_lookup: bool = False)`

Bases: `object`

Disassembly class.

Stores bytecode, and its disassembly. Additionally it will gather the following information on the existing functions in the disassembled code: - function hashes - function name to entry point mapping - function entry point to function name mapping

**assign\_bytecode** (`bytecode`)

**get\_easm** ()



### Returns

```
mythril.disassembler.disassembly.get_function_info(index: int, instruction_list: list, signature_database: mythril.support.signatures.SignatureDB)
→ Tuple[str, int, str]
```

Finds the function information for a call table entry Solidity uses the first 4 bytes of the calldata to indicate which function the message call should execute The generated code that directs execution to the correct function looks like this:

- PUSH function\_hash
- EQ
- PUSH entry\_point
- JUMPI

This function takes an index that points to the first instruction, and from that finds out the function hash, function entry and the function name.

### Parameters

- **index** – Start of the entry pattern
- **instruction\_list** – Instruction list for the contract that is being analyzed
- **signature\_database** – Database used to map function hashes to their respective function names

**Returns** function hash, function entry point, function name

## Module contents

### 6.1.4 mythril.ethereum package

#### Subpackages

#### mythril.ethereum.interface package

#### Subpackages

#### mythril.ethereum.interface.rpc package

#### Submodules

#### mythril.ethereum.interface.rpc.base\_client module

This module provides a basic RPC interface client.

This code is adapted from: <https://github.com/ConsenSys/ethjsonrpc>

```
class mythril.ethereum.interface.rpc.base_client.BaseClient
```

Bases: object

The base RPC client class.

```
eth_blockNumber ()
    TODO: documentation
    https://github.com/ethereum/wiki/wiki/JSON-RPC#eth\_blocknumber
    TESTED

eth_coinbase ()
    TODO: documentation
    https://github.com/ethereum/wiki/wiki/JSON-RPC#eth\_coinbase
    TESTED

eth_getBalance (address=None, block='latest')
    TODO: documentation
    https://github.com/ethereum/wiki/wiki/JSON-RPC#eth\_getbalance
    TESTED

eth_getBlockByNumber (block='latest', tx_objects=True)
    TODO: documentation
    https://github.com/ethereum/wiki/wiki/JSON-RPC#eth\_getblockbynumber
    TESTED

eth_getCode (address, default_block='latest')
    TODO: documentation
    https://github.com/ethereum/wiki/wiki/JSON-RPC#eth\_getcode
    NEEDS TESTING

eth_getStorageAt (address=None, position=0, block='latest')
    TODO: documentation
    https://github.com/ethereum/wiki/wiki/JSON-RPC#eth\_getstorageat
    TESTED

eth_getTransactionReceipt (tx_hash)
    TODO: documentation
    https://github.com/ethereum/wiki/wiki/JSON-RPC#eth\_gettransactionreceipt
    TESTED
```

## mythril.ethereum.interface.rpc.client module

This module contains a basic Ethereum RPC client.

This code is adapted from: <https://github.com/ConsenSys/ethjsonrpc>

```
class mythril.ethereum.interface.rpc.client.EthJsonRpc (host='localhost',  
                                                    port=8545, tls=False)  
    Bases: mythril.ethereum.interface.rpc.base_client.BaseClient  
    Ethereum JSON-RPC client class.  
  
    close ()  
        Close the RPC client's session.
```

## mythril.ethereum.interface.rpc.constants module

This file contains constants used by the Ethereum JSON RPC interface.

## mythril.ethereum.interface.rpc.exceptions module

This module contains exceptions regarding JSON-RPC communication.

**exception** `mythril.ethereum.interface.rpc.exceptions.BadJsonError`  
Bases: `mythril.ethereum.interface.rpc.exceptions.EthJsonRpcError`

An RPC exception denoting that the RPC instance returned a bad JSON object.

**exception** `mythril.ethereum.interface.rpc.exceptions.BadResponseError`  
Bases: `mythril.ethereum.interface.rpc.exceptions.EthJsonRpcError`

An RPC exception denoting that the RPC instance returned a bad response.

**exception** `mythril.ethereum.interface.rpc.exceptions.BadStatusCodeError`  
Bases: `mythril.ethereum.interface.rpc.exceptions.EthJsonRpcError`

An RPC exception denoting a bad status code returned by the RPC instance.

**exception** `mythril.ethereum.interface.rpc.exceptions.ConnectionError`  
Bases: `mythril.ethereum.interface.rpc.exceptions.EthJsonRpcError`

An RPC exception denoting there was an error in connecting to the RPC instance.

**exception** `mythril.ethereum.interface.rpc.exceptions.EthJsonRpcError`  
Bases: `Exception`

The JSON-RPC base exception type.

## mythril.ethereum.interface.rpc.utils module

This module contains various utility functions regarding the RPC data format and validation.

`mythril.ethereum.interface.rpc.utils.clean_hex(d)`  
Convert decimal to hex and remove the “L” suffix that is appended to large numbers.

**Parameters** `d` –

**Returns**

`mythril.ethereum.interface.rpc.utils.ether_to_wei(ether)`  
Convert ether to wei.

**Parameters** `ether` –

**Returns**

`mythril.ethereum.interface.rpc.utils.hex_to_dec(x)`  
Convert hex to decimal.

**Parameters** `x` –

**Returns**

`mythril.ethereum.interface.rpc.utils.validate_block(block)`

**Parameters** `block` –

**Returns**

```
mythril.ethereum.interface.rpc.utils.wei_to_ether(wei)
```

Convert wei to ether.

**Parameters** `wei` –

**Returns**

## Module contents

## Module contents

## Submodules

### mythril.ethereum.evmcontract module

This module contains the class representing EVM contracts, aka Smart Contracts.

```
class mythril.ethereum.evmcontract.EVMContract (code="", creation_code="",
                                              name='Unknown',
                                              enable_online_lookup=False)
```

Bases: `persistent.Persistent`

This class represents an address with associated code (Smart Contract).

**as\_dict()**

**Returns**

**bytecode\_hash**

**Returns** runtime bytecode hash

**creation\_bytecode\_hash**

**Returns** Creation bytecode hash

**get\_creation\_easm()**

**Returns**

**get\_easm()**

**Returns**

**matches\_expression(expression)**

**Parameters** `expression` –

**Returns**

### mythril.ethereum.util module

This module contains various utility functions regarding unit conversion and solc integration.

```
mythril.ethereum.util.extract_binary(file: str) → str
```

```
mythril.ethereum.util.extract_version(file: str) → Optional[str]
```

```
mythril.ethereum.util.get_indexed_address(index)
```

**Parameters** `index` –

**Returns**

```
mythril.ethereum.util.get_random_address()
```

#### Returns

```
mythril.ethereum.util.get_solc_json(file, solc_binary='solc', solc_settings_json=None)
```

#### Parameters

- **file** –
- **solc\_binary** –
- **solc\_settings\_json** –

#### Returns

```
mythril.ethereum.util.safe_decode(hex_encoded_string)
```

#### Parameters **hex\_encoded\_string** –

#### Returns

```
mythril.ethereum.util.solc_exists(version)
```

#### Parameters **version** –

#### Returns

## Module contents

### 6.1.5 mythril.interfaces package

#### Submodules

#### mythril.interfaces.cli module

mythril.py: Bug hunting on the Ethereum blockchain

<http://www.github.com/ConsenSys/mythril>

```
mythril.interfaces.cli.add_analysis_args(options)
```

Adds arguments for analysis

#### Parameters **options** – Analysis Options

```
mythril.interfaces.cli.add_graph_commands(parser: argparse.ArgumentParser)
```

```
mythril.interfaces.cli.contract_hash_to_address(args: argparse.Namespace)
```

prints the hash from function signature :param args: :return:

```
mythril.interfaces.cli.create_analyzer_parser(analyzer_parser: arg-
                                              parse.ArgumentParser)
```

Modify parser to handle analyze command :param analyzer\_parser: :return:

```
mythril.interfaces.cli.create_concolic_parser(parser: argparse.ArgumentParser) →
                                              argparse.ArgumentParser
```

Get parser which handles arguments for concolic branch flipping

```
mythril.interfaces.cli.create_disassemble_parser(parser: argparse.ArgumentParser)
```

Modify parser to handle disassembly :param parser: :return:

```
mythril.interfaces.cli.create_func_to_hash_parser(parser: argparse.ArgumentParser)
```

Modify parser to handle func\_to\_hash command :param parser: :return:

`mythril.interfaces.cli.create_hash_to_addr_parser` (*hash\_parser:* `parse.ArgumentParser`) *arg-*  
 Modify parser to handle hash\_to\_addr command :param hash\_parser: :return:

`mythril.interfaces.cli.create_read_storage_parser` (*read\_storage\_parser:* `parse.ArgumentParser`) *arg-*  
 Modify parser to handle storage slots :param read\_storage\_parser: :return:

`mythril.interfaces.cli.create_safe_functions_parser` (*parser:* `parse.ArgumentParser`) *arg-*  
 The duplication exists between safe-functions and analyze as some of them have different default values. :param parser: Parser

`mythril.interfaces.cli.execute_command` (*disassembler:* `mythril.mythril.mythril_disassembler.MythrilDisassembler`,  
*address:* `str`, *parser:* `argparse.ArgumentParser`,  
*args:* `argparse.Namespace`)  
 Execute command :param disassembler: :param address: :param parser: :param args: :return:

`mythril.interfaces.cli.exit_with_error` (*format\_*, *message*)  
 Exits with error :param **format\_**: The format of the message :param message: message

`mythril.interfaces.cli.get_creation_input_parser` () → `argparse.ArgumentParser`  
 Returns Parser which handles input :return: Parser which handles input

`mythril.interfaces.cli.get_output_parser` () → `argparse.ArgumentParser`  
 Get parser which handles output :return: Parser which handles output

`mythril.interfaces.cli.get_rpc_parser` () → `argparse.ArgumentParser`  
 Get parser which handles RPC flags :return: Parser which handles rpc inputs

`mythril.interfaces.cli.get_runtime_input_parser` () → `argparse.ArgumentParser`  
 Returns Parser which handles input :return: Parser which handles input

`mythril.interfaces.cli.get_safe_functions_parser` () → `argparse.ArgumentParser`  
 Returns Parser which handles checking for safe functions :return: Parser which handles checking for safe functions

`mythril.interfaces.cli.get_utilities_parser` () → `argparse.ArgumentParser`  
 Get parser which handles utilities flags :return: Parser which handles utility flags

`mythril.interfaces.cli.load_code` (*disassembler:* `mythril.mythril.mythril_disassembler.MythrilDisassembler`,  
*args:* `argparse.Namespace`)  
 Loads code into disassembly and returns address :param disassembler: :param args: :return: Address

`mythril.interfaces.cli.main` () → None  
 The main CLI interface entry point.

`mythril.interfaces.cli.parse_args_and_execute` (*parser:* `argparse.ArgumentParser`, *args:* `argparse.Namespace`) → None  
 Parses the arguments :param parser: The parser :param args: The args

`mythril.interfaces.cli.print_function_report` (*myth\_disassembler:* `mythril.mythril.mythril_disassembler.MythrilDisassembler`,  
*report:* `mythril.analysis.report.Report`)  
 Prints the function report :param report: Mythril's report :return:

`mythril.interfaces.cli.set_config` (*args:* `argparse.Namespace`)  
 Set config based on args :param args: :return: modified config

`mythril.interfaces.cli.validate_args` (*args:* `argparse.Namespace`)  
 Validate cli args :param args: :return:

**mythril.interfaces.epic module**

Don't ask.

```
class mythril.interfaces.epic.LolCat (mode=256,                output=<_io.TextIOWrapper
                                name='<stdout>' mode='w' encoding='UTF-8'>)
```

Bases: object

Cats lel.

**ansi** (rgb)

Parameters rgb –

Returns

**cat** (fd, options)

Parameters

- fd –
- options –

**println** (s, options)

Parameters

- s –
- options –

**println\_ani** (s, options)

Parameters

- s –
- options –

Returns

**println\_plain** (s, options)

Parameters

- s –
- options –

**rainbow** (freq, i)

Parameters

- freq –
- i –

Returns

**wrap** (\*codes)

Parameters codes –

Returns

mythril.interfaces.epic.**detect\_mode** (term\_hint='xterm-256color')

Poor-mans color mode detection.

mythril.interfaces.epic.**reset** ()

```
mythril.interfaces.epic.run()
```

Main entry point.

## Module contents

### 6.1.6 mythril.laser package

#### Subpackages

#### mythril.laser.ethereum package

#### Subpackages

#### mythril.laser.ethereum.function\_managers package

#### Submodules

#### mythril.laser.ethereum.function\_managers.exponent\_function\_manager module

```
class mythril.laser.ethereum.function_managers.exponent_function_manager.ExponentFunctionManager
    Bases: object
```

Uses an uninterpreted function for exponentiation with the following properties: 1)  $\text{power}(a, b) > 0$  2) if  $a = 256$   
 $\Rightarrow \text{forall } i \text{ if } b = i \text{ then } \text{power}(a, b) = (256 \wedge i) \% (2^{256})$

Only these two properties are added as to handle indexing of boolean arrays. Caution should be exercised when increasing the conditions since it severely affects the solving time.

```
create_condition (base: mythril.laser.smt.bitvec.BitVec, exponent: mythril.laser.smt.bitvec.BitVec)
    → Tuple[mythril.laser.smt.bitvec.BitVec, mythril.laser.smt.bool.Bool]
```

Creates a condition for exponentiation :param base: The base of exponentiation :param exponent: The exponent of the exponentiation :return: Tuple of condition and the exponentiation result

#### mythril.laser.ethereum.function\_managers.keccak\_function\_manager module

```
class mythril.laser.ethereum.function_managers.keccak_function_manager.KeccakFunctionManager
    Bases: object
```

A bunch of uninterpreted functions are considered like keccak256\_160, ... where keccak256\_160 means the input of keccak256() is 160 bit number. the range of these functions are constrained to some mutually disjoint intervals All the hashes modulo 64 are 0 as we need a spread among hashes for array type data structures All the functions are kind of one to one due to constraint of the existence of inverse for each encountered input. For more info <https://files.sri.inf.ethz.ch/website/papers/sp20-verx.pdf>

```
create_conditions () → mythril.laser.smt.bool.Bool
```

```
create_keccak (data: mythril.laser.smt.bitvec.BitVec) → mythril.laser.smt.bitvec.BitVec
```

Creates Keccak of the data :param data: input :return: Tuple of keccak and the condition it should satisfy

```
static find_concrete_keccak (data: mythril.laser.smt.bitvec.BitVec) →
    mythril.laser.smt.bitvec.BitVec
```

Calculates concrete keccak :param data: input bitvecval :return: concrete keccak output



```

get_concrete_hash_data (model) → Dict[int, List[Optional[int]]]
    returns concrete values of hashes in the self.hash_result_store :param model: The z3 model to query
    for concrete values :return: A dictionary with concrete hashes { <hash_input_size> : [<concrete_hash>,
    <concrete_hash>]}

static get_empty_keccak_hash () → mythril.laser.smt.bitvec.BitVec
    returns sha3("") :return:

get_function (length: int) → Tuple[mythril.laser.smt.function.Function,
    mythril.laser.smt.function.Function]
    Returns the keccak functions for the corresponding length :param length: input size :return: tuple of keccak
    and it's inverse

hash_matcher = 'ffffff'

reset ()

```

## Module contents

### mythril.laser.ethereum.state package

#### Submodules

#### mythril.laser.ethereum.state.account module

This module contains account-related functionality.

This includes classes representing accounts and their storage.

```

class mythril.laser.ethereum.state.account.Account (address:
    Union[mythril.laser.smt.bitvec.BitVec,
    str], code=None, con-
    tract_name=None, balances:
    mythril.laser.smt.array.Array =
    None, concrete_storage=False,
    dynamic_loader=None,
    nonce=0)

```

Bases: object

Account class representing ethereum accounts.

```
add_balance (balance: Union[int, mythril.laser.smt.bitvec.BitVec]) → None
```

**Parameters** **balance** –

```
as_dict
```

**Returns**

```
serialised_code ()
```

```
set_balance (balance: Union[int, mythril.laser.smt.bitvec.BitVec]) → None
```

**Parameters** **balance** –

```
set_storage (storage: Dict[KT, VT])
```

Sets concrete storage

```

class mythril.laser.ethereum.state.account.Storage (concrete=False, address=None,
    dynamic_loader=None)

```

Bases: object

Storage class represents the storage of an Account.

### **mythril.laser.ethereum.state.annotation module**

This module includes classes used for annotating trace information.

This includes the base StateAnnotation class, as well as an adaption, which will not be copied on every new state.

**class** `mythril.laser.ethereum.state.annotation.MergeableStateAnnotation`  
Bases: `mythril.laser.ethereum.state.annotation.StateAnnotation`

This class allows a base annotation class for annotations that can be merged.

**check\_merge\_annotation** (*annotation*) → bool

**merge\_annotation** (*annotation*)

**class** `mythril.laser.ethereum.state.annotation.NoCopyAnnotation`  
Bases: `mythril.laser.ethereum.state.annotation.StateAnnotation`

This class provides a base annotation class for annotations that shouldn't be copied on every new state.

Rather the same object should be propagated. This is very useful if you are looking to analyze a property over multiple substates

**class** `mythril.laser.ethereum.state.annotation.StateAnnotation`  
Bases: `object`

The StateAnnotation class is used to persist information over traces.

This allows modules to reason about traces without the need to traverse the state space themselves.

**persist\_over\_calls**

If this function returns true then laser will propagate the annotation between calls

The default is set to False

**persist\_to\_world\_state**

If this function returns true then laser will also annotate the world state.

If you want annotations to persist through different user initiated message call transactions then this should be enabled.

The default is set to False

**search\_importance**

Used in estimating the priority of a state annotated with the corresponding annotation. Default is 1

### **mythril.laser.ethereum.state.calldata module**

This module declares classes to represent call data.

**class** `mythril.laser.ethereum.state.calldata.BaseCalldata` (*tx\_id: str*)  
Bases: `object`

Base calldata class This represents the calldata provided when sending a transaction to a contract.

**calldatasize**

**Returns** Calldata size for this calldata object

**concrete** (*model: z3.z3.Model*) → list

Returns a concrete version of the calldata using the provided model.

**Parameters model** –

**get\_word\_at** (*offset: int*) → `mythril.laser.smt.expression.Expression`  
Gets word at offset.

**Parameters offset** –

**Returns**

**size**

Returns the exact size of this calldata, this is not normalized.

**Returns** unnormalized call data size

```
class mythril.laser.ethereum.state.calldata.BasicConcreteCalldata (tx_id: str,  
                                                                calldata:  
                                                                list)
```

Bases: `mythril.laser.ethereum.state.calldata.BaseCalldata`

A base class to represent concrete call data.

**concrete** (*model: z3.z3.Model*) → list

**Parameters model** –

**Returns**

**size**

**Returns**

```
class mythril.laser.ethereum.state.calldata.BasicSymbolicCalldata (tx_id: str)
```

Bases: `mythril.laser.ethereum.state.calldata.BaseCalldata`

A basic class representing symbolic call data.

**concrete** (*model: z3.z3.Model*) → list

**Parameters model** –

**Returns**

**size**

**Returns**

```
class mythril.laser.ethereum.state.calldata.ConcreteCalldata (tx_id: str, calldata:  
                                                                list)
```

Bases: `mythril.laser.ethereum.state.calldata.BaseCalldata`

A concrete call data representation.

**concrete** (*model: z3.z3.Model*) → list

**Parameters model** –

**Returns**

**size**

**Returns**

```
class mythril.laser.ethereum.state.calldata.SymbolicCalldata (tx_id: str)
```

Bases: `mythril.laser.ethereum.state.calldata.BaseCalldata`

A class for representing symbolic call data.

**concrete** (*model: z3.z3.Model*) → list

**Parameters** `model` –

**Returns**

**size**

**Returns**

## mythril.laser.ethereum.state.constraints module

This module contains the class used to represent state-change constraints in the call graph.

```
class mythril.laser.ethereum.state.constraints.Constraints (constraint_list: Optional[List[mythril.laser.smt.bool.Bool]]  
                                                         = None)
```

Bases: list

This class should maintain a solver and its constraints, This class tries to make the Constraints() object as a simple list of constraints with some background processing.

```
append (constraint: Union[bool, mythril.laser.smt.bool.Bool]) → None
```

**Parameters** `constraint` – The constraint to be appended

**as\_list**

**Returns** returns the list of constraints

```
copy () → mythril.laser.ethereum.state.constraints.Constraints
```

Return a shallow copy of the list.

```
get_all_constraints ()
```

```
is_possible (solver_timeout=None) → bool
```

**Parameters** `solver_timeout` – The default timeout uses analysis timeout from `args.solver_timeout`

**Returns** True/False based on the existence of solution of constraints

## mythril.laser.ethereum.state.environment module

This module contains the representation for an execution state's environment.

```
class mythril.laser.ethereum.state.environment.Environment (active_account:  
                                                         mythril.laser.ethereum.state.account.Account,  
                                                         sender:  
                                                         z3.z3.ExprRef,  
                                                         calldata:  
                                                         mythril.laser.ethereum.state.calldata.BaseCallData,  
                                                         gasprice:  
                                                         z3.z3.ExprRef,  
                                                         callvalue:  
                                                         z3.z3.ExprRef, origin:  
                                                         z3.z3.ExprRef, base-  
                                                         fee: z3.z3.ExprRef,  
                                                         code=None,  
                                                         static=False)
```

Bases: object

The environment class represents the current execution environment for the symbolic executor.

**as\_dict**

**Returns**

## mythril.laser.ethereum.state.global\_state module

This module contains a representation of the global execution state.

```
class mythril.laser.ethereum.state.global_state.GlobalState(world_state: World-
    State, environment:
    mythril.laser.ethereum.state.environment.Envir-
    node:
    mythril.laser.ethereum.cfg.Node,
    ma-
    chine_state=None,
    transac-
    tion_stack=None,
    last_return_data=None,
    annotations=None)
```

Bases: object

GlobalState represents the current globalstate.

**accounts**

**Returns**

**add\_annotations** (annotations: List[mythril.laser.ethereum.state.annotation.StateAnnotation])

Function used to add annotations to global state :param annotations: :return:

**annotate** (annotation: mythril.laser.ethereum.state.annotation.StateAnnotation) → None

**Parameters annotation –**

**annotations**

**Returns**

**current\_transaction**

**Returns**

**get\_annotations** (annotation\_type: type) → Iterable[mythril.laser.ethereum.state.annotation.StateAnnotation]

Filters annotations for the queried annotation type. Designed particularly for modules with annotations:  
globalstate.get\_annotations(MySpecificModuleAnnotation)

**Parameters annotation\_type –** The type to filter annotations for

**Returns** filter of matching annotations

**get\_current\_instruction** () → Dict[KT, VT]

Gets the current instruction for this GlobalState.

**Returns**

**instruction**

**Returns**

**new\_bitvec** (name: str, size=256, annotations=None) → z3.z3.BitVec

**Parameters**

• **name –**

- **size** –

**Returns**

### `mythril.laser.ethereum.state.machine_state` module

This module contains a representation of the EVM's machine state and its stack.

**class** `mythril.laser.ethereum.state.machine_state.MachineStack` (*default\_list=None*)  
Bases: `list`

Defines EVM stack, overrides the default list to handle overflows.

**STACK\_LIMIT** = 1024

**append** (*element: Union[int, mythril.laser.smt.expression.Expression]*) → None

**This function ensures the following properties when appending to a list:**

- Element appended to this list should be a BitVec
- Ensures stack overflow bound

**Parameters** **element** – element to be appended to the list

**Function** appends the element to list if the size is less than **STACK\_LIMIT**, else throws an error

**pop** (*index=-1*) → `Union[int, mythril.laser.smt.expression.Expression]`

This function ensures stack underflow bound :param *index*:index to be popped, same as the `list()` class.  
:returns popped value :function: same as `list()` class but throws `StackUnderflowException` for popping from an empty list

**class** `mythril.laser.ethereum.state.machine_state.MachineState` (*gas\_limit:*  
*int,* *pc=0,*  
*stack=None,*  
*subrou-*  
*tine\_stack=None,*  
*memory: Op-*  
*tional[mythril.laser.ethereum.state.memory*  
*= None, con-*  
*straints=None,*  
*depth=0,*  
*max\_gas\_used=0,*  
*min\_gas\_used=0)*

Bases: `object`

`MachineState` represents current machine state also referenced to as `mu`.

**as\_dict**

**Returns**

**calculate\_extension\_size** (*start: int, size: int*) → `int`

**Parameters**

- **start** –
- **size** –

**Returns**

**calculate\_memory\_gas** (*start: int, size: int*)

**Parameters**

- **start** –
- **size** –

**Returns****check\_gas** ()

Check whether the machine is out of gas.

**mem\_extend** (*start*: Union[int, mythril.laser.smt.bitvec.BitVec], *size*: Union[int, mythril.laser.smt.bitvec.BitVec]) → None

Extends the memory of this machine state.

**Parameters**

- **start** – Start of memory extension
- **size** – Size of memory extension

**memory\_size****Returns****memory\_write** (*offset*: int, *data*: List[Union[int, mythril.laser.smt.bitvec.BitVec]]) → None

Writes data to memory starting at offset.

**Parameters**

- **offset** –
- **data** –

**pop** (*amount*=1) → Union[mythril.laser.smt.bitvec.BitVec, List[mythril.laser.smt.bitvec.BitVec]]

Pops amount elements from the stack.

**Parameters amount** –**Returns**

mythril.laser.ethereum.state.machine\_state.**ceil132** (*value*: int, \*, *ceiling*: int = 32) → int

**mythril.laser.ethereum.state.memory module**

This module contains a representation of a smart contract's memory.

**class** mythril.laser.ethereum.state.memory.**Memory**

Bases: object

A class representing contract memory with random access.

**extend** (*size*: int)**Parameters size** –**get\_word\_at** (*index*: int) → Union[int, mythril.laser.smt.bitvec.BitVec]

Access a word from a specified memory index.

**Parameters index** – integer representing the index to access**Returns** 32 byte word at the specified index

```
write_word_at (index: int, value: Union[int, mythril.laser.smt.bitvec.BitVec, bool,
mythril.laser.smt.bool.Bool]) → None
Writes a 32 byte word to memory at the specified index
```

#### Parameters

- **index** – index to write to
- **value** – the value to write to memory

```
mythril.laser.ethereum.state.memory.convert_bv (val: Union[int,
mythril.laser.smt.bitvec.BitVec]) →
mythril.laser.smt.bitvec.BitVec
```

### mythril.laser.ethereum.state.return\_data module

This module declares classes to represent call data.

```
class mythril.laser.ethereum.state.return_data.ReturnData (return_data:
List[mythril.laser.smt.bitvec.BitVec],
return_data_size:
mythril.laser.smt.bitvec.BitVec)
```

Bases: object

Base returndata class.

**size**

**Returns** Calldata size for this calldata object

### mythril.laser.ethereum.state.world\_state module

This module contains a representation of the EVM's world state.

```
class mythril.laser.ethereum.state.world_state.WorldState (transaction_sequence=None,
annotations:
List[mythril.laser.ethereum.state.annotation.StateAnnotation]
= None, constraints:
mythril.laser.ethereum.state.constraints.Constraints
= None)
```

Bases: object

The WorldState class represents the world state as described in the yellow paper.

**accounts**

```
accounts_exist_or_load (addr, dynamic_loader: mythril.support.loader.DynLoader) →
mythril.laser.ethereum.state.account.Account
returns account if it exists, else it loads from the dynamic loader :param addr: address :param dy-
namic_loader: Dynamic Loader :return: The code
```

```
annotate (annotation: mythril.laser.ethereum.state.annotation.StateAnnotation) → None
```

**Parameters** annotation –

**annotations**

**Returns**



**create\_account** (*balance=0, address=None, concrete\_storage=False, dynamic\_loader=None, creator=None, code=None, nonce=0*) → *mythril.laser.ethereum.state.account.Account*  
 Create non-contract account.

#### Parameters

- **address** – The account’s address
- **balance** – Initial balance for the account
- **concrete\_storage** – Interpret account storage as concrete
- **dynamic\_loader** – used for dynamically loading storage from the block chain
- **creator** – The address of the creator of the contract if it’s a contract
- **code** – The code of the contract, if it’s a contract
- **nonce** – Nonce of the account

**Returns** The new account

**create\_initialized\_contract\_account** (*contract\_code, storage*) → *None*  
 Creates a new contract account, based on the contract code and storage provided The contract code only includes the runtime contract bytecode.

#### Parameters

- **contract\_code** – Runtime bytecode for the contract
- **storage** – Initial storage for the contract

**Returns** The new account

**get\_annotations** (*annotation\_type: type*) → *Iterator[mythril.laser.ethereum.state.annotation.StateAnnotation]*  
 Filters annotations for the queried annotation type. Designed particularly for modules with annotations: *worldstate.get\_annotations(MySpecificModuleAnnotation)*

**Parameters** **annotation\_type** – The type to filter annotations for

**Returns** filter of matching annotations

**put\_account** (*account: mythril.laser.ethereum.state.account.Account*) → *None*

**Parameters** **account** –

## Module contents

**mythril.laser.ethereum.strategy package**

**Subpackages**

**mythril.laser.ethereum.strategy.extensions package**

**Submodules**

## mythril.laser.ethereum.strategy.extensions.bounded\_loops module

**class** `mythril.laser.ethereum.strategy.extensions.bounded_loops.BoundedLoopsStrategy` (*super\_strategy*, *mythril.laser.ethereum.strategy.extensions.bounded\_loops.BoundedLoopsStrategy*, *\*\*kwargs*)

Bases: `mythril.laser.ethereum.strategy.BasicSearchStrategy`

Adds loop pruning to the search strategy. Ignores JUMPI instruction if the destination was targeted >JUMPDEST\_LIMIT times.

**static** `calculate_hash(i: int, j: int, trace: List[int])` → int  
calculate hash(trace[i: j]) :param i: :param j: :param trace: :return: hash(trace[i: j])

**static** `count_key(trace: List[int], key: int, start: int, size: int)` → int  
Count continuous loops in the trace. :param trace: :param key: :param size: :return:

**static** `get_loop_count(trace: List[int])` → int  
Gets the loop count :param trace: :return:

**get\_strategic\_global\_state()** → `mythril.laser.ethereum.state.global_state.GlobalState`  
Returns the next state

**Returns** Global state

**class** `mythril.laser.ethereum.strategy.extensions.bounded_loops.JumpdestCountAnnotation`  
Bases: `mythril.laser.ethereum.state.annotation.StateAnnotation`

State annotation that counts the number of jumps per destination.

## Module contents

### Submodules

## mythril.laser.ethereum.strategy.basic module

This module implements basic symbolic execution search strategies.

**class** `mythril.laser.ethereum.strategy.basic.BreadthFirstSearchStrategy` (*work\_list*, *max\_depth*, *\*\*kwargs*)

Bases: `mythril.laser.ethereum.strategy.BasicSearchStrategy`

Implements a breadth first search strategy I.E.

Execute all states of a “level” before continuing

**get\_strategic\_global\_state()** → `mythril.laser.ethereum.state.global_state.GlobalState`

**Returns**

**class** `mythril.laser.ethereum.strategy.basic.DepthFirstSearchStrategy` (*work\_list*, *max\_depth*, *\*\*kwargs*)

Bases: `mythril.laser.ethereum.strategy.BasicSearchStrategy`

Implements a depth first search strategy I.E.

Follow one path to a leaf, and then continue to the next one

**get\_strategic\_global\_state()** → `mythril.laser.ethereum.state.global_state.GlobalState`

**Returns**

```
class mythril.laser.ethereum.strategy.basic.ReturnRandomNaivelyStrategy (work_list,  
                                                                    max_depth,  
                                                                    **kwargs)
```

Bases: *mythril.laser.ethereum.strategy.BasicSearchStrategy*

chooses a random state from the worklist with equal likelihood.

**get\_strategic\_global\_state** () → mythril.laser.ethereum.state.global\_state.GlobalState

**Returns**

```
class mythril.laser.ethereum.strategy.basic.ReturnWeightedRandomStrategy (work_list,  
                                                                    max_depth,  
                                                                    **kwargs)
```

Bases: *mythril.laser.ethereum.strategy.BasicSearchStrategy*

chooses a random state from the worklist with likelihood based on inverse proportion to depth.

**get\_strategic\_global\_state** () → mythril.laser.ethereum.state.global\_state.GlobalState

**Returns**

### mythril.laser.ethereum.strategy.beam module

```
class mythril.laser.ethereum.strategy.beam.BeamSearch (work_list,      max_depth,  
                                                         beam_width, **kwargs)
```

Bases: *mythril.laser.ethereum.strategy.BasicSearchStrategy*

chooses a random state from the worklist with equal likelihood.

**static beam\_priority** (*state*)

**get\_strategic\_global\_state** () → mythril.laser.ethereum.state.global\_state.GlobalState

**Returns**

**sort\_and\_eliminate\_states** ()

### mythril.laser.ethereum.strategy.concolic module

```
class mythril.laser.ethereum.strategy.concolic.ConcolicStrategy (work_list:  
                                                                    List[mythril.laser.ethereum.state.global  
                                                                    max_depth:  
                                                                    int,      trace:  
                                                                    List[List[Tuple[int,  
                                                                    str]]],  
                                                                    flip_branch_addresses:  
                                                                    List[str])
```

Bases: *mythril.laser.ethereum.strategy.CriterionSearchStrategy*

Executes program concolically using the input trace till a specific branch

**check\_completion\_criterion** ()

**get\_strategic\_global\_state** () → mythril.laser.ethereum.state.global\_state.GlobalState

This function does the following:- 1) Choose the states by following the concolic trace. 2) In case we have an executed JUMPI that is in flip\_branch\_addresses, flip that branch. :return:

```
class mythril.laser.ethereum.strategy.concolic.TraceAnnotation (trace=None)  
Bases: mythril.laser.ethereum.state.annotation.StateAnnotation
```

This is the annotation used by the ConcolicStrategy to store concolic traces.

**persist\_over\_calls**

If this function returns true then laser will propagate the annotation between calls

The default is set to False

**Module contents**

```
class mythril.laser.ethereum.strategy.BasicSearchStrategy(work_list, max_depth,  
                                                         **kwargs)
```

Bases: `abc.ABC`

A basic search strategy which halts based on depth

```
get_strategic_global_state()
```

```
class mythril.laser.ethereum.strategy.CriterionSearchStrategy(work_list,  
                                                             max_depth,  
                                                             **kwargs)
```

Bases: `mythril.laser.ethereum.strategy.BasicSearchStrategy`

If a criterion is satisfied, the search halts

```
get_strategic_global_state()
```

```
set_criterion_satisfied()
```

**mythril.laser.ethereum.transaction package****Submodules****mythril.laser.ethereum.transaction.concolic module**

This module contains functions to set up and execute concolic message calls.

```
mythril.laser.ethereum.transaction.concolic.execute_contract_creation(laser_evm,  
                                                                        callee_address,  
                                                                        caller_address,  
                                                                        ori-  
                                                                        gin_address,  
                                                                        data,  
                                                                        gas_limit,  
                                                                        gas_price,  
                                                                        value,  
                                                                        code=None,  
                                                                        track_gas=False,  
                                                                        con-  
                                                                        tract_name=None)
```

Executes a contract creation transaction concretely.

**Parameters**

- **laser\_evm** –
- **callee\_address** –
- **caller\_address** –

- **origin\_address** –
- **code** –
- **data** –
- **gas\_limit** –
- **gas\_price** –
- **value** –
- **track\_gas** –

#### Returns

```
mythril.laser.ethereum.transaction.concolic.execute_message_call(laser_evm,
                                                                    callee_address,
                                                                    caller_address,
                                                                    ori-
                                                                    gin_address,
                                                                    data,
                                                                    gas_limit,
                                                                    gas_price,
                                                                    value,
                                                                    code=None,
                                                                    track_gas=False)
→
Union[None,
List[mythril.laser.ethereum.state.global
```

Execute a message call transaction from all open states.

#### Parameters

- **laser\_evm** –
- **callee\_address** –
- **caller\_address** –
- **origin\_address** –
- **code** –
- **data** –
- **gas\_limit** –
- **gas\_price** –
- **value** –
- **track\_gas** –

#### Returns

```
mythril.laser.ethereum.transaction.concolic.execute_transaction(*args,
                                                                    **kwargs) →
Union[None,
List[mythril.laser.ethereum.state.global
```

Chooses the transaction type based on callee address and executes the transaction

## mythril.laser.ethereum.transaction.symbolic module

This module contains functions setting up and executing transactions with symbolic values.

```
class mythril.laser.ethereum.transaction.symbolic.Actors (creator=100475310549029526324481294656594  
                                                    at-  
                                                    tacker=1271270613000041655817448348132275  
                                                    someguy=97433442488726861213578988847752
```

Bases: object

**attacker**

**creator**

```
mythril.laser.ethereum.transaction.symbolic.execute_contract_creation (laser_evm,  
                                                                    con-  
                                                                    tract_initialization_code,  
                                                                    con-  
                                                                    tract_name=None,  
                                                                    world_state=None,  
                                                                    ori-  
                                                                    gin=100475310549029526324  
                                                                    caller=1004753105490295263  
                                                                    →  
                                                                    mythril.laser.ethereum.state.ac
```

Executes a contract creation transaction from all open states.

### Parameters

- **laser\_evm** –
- **contract\_initialization\_code** –
- **contract\_name** –

### Returns

```
mythril.laser.ethereum.transaction.symbolic.execute_message_call (laser_evm,  
                                                                    callee_address:  
                                                                    mythril.laser.smt.bitvec.BitVec,  
                                                                    func_hashes:  
                                                                    List[List[int]]  
                                                                    = None) →  
                                                                    None
```

Executes a message call transaction from all open states.

### Parameters

- **laser\_evm** –
- **callee\_address** –

```
mythril.laser.ethereum.transaction.symbolic.execute_transaction (*args,  
                                                                    **kwargs)
```

Chooses the transaction type based on callee address and executes the transaction

```
mythril.laser.ethereum.transaction.symbolic.generate_function_constraints (calldata:  
                                                                    mythril.laser.ethereum.st  
                                                                    func_hashes:  
                                                                    List[List[int]]  
                                                                    →  
                                                                    List[mythril.laser.smt.bo
```

This will generate constraints for fixing the function call part of calldata :param calldata: Calldata :param func\_hashes: The list of function hashes allowed for this transaction :return: Constraints List

## mythril.laser.ethereum.transaction.transaction\_models module

This module contains the transaction models used throughout LASER's symbolic execution.

```
class mythril.laser.ethereum.transaction.transaction_models.BaseTransaction (world_state:
    mythril.laser.ethereum.state.global_state.GlobalState,
    callee_account:
    mythril.laser.ethereum.account.Account,
    =
    None,
    caller:
    z3.z3.ExprRef,
    =
    None,
    call_data=None,
    identifier:
    Optional[str],
    =
    None,
    gas_price=None,
    gas_limit=None,
    origin:
    Optional[str],
    code=None,
    call_value=None,
    init_call_data=True,
    static=False,
    base_fee=None)
```

Bases: object

Basic transaction class holding common data.

**initial\_global\_state**() → mythril.laser.ethereum.state.global\_state.GlobalState

**initial\_global\_state\_from\_environment**(environment, active\_function)

### Parameters

- **environment** –
- **active\_function** –

### Returns

```
class mythril.laser.ethereum.transaction.transaction_models.ContractCreationTransaction (wor
```

```
myt
call
z3.2
=
Non
call
iden
ti-
fier
Op-
tion
=
Non
gas
gas
ori-
gin-
cod
call
com
trac
com
trac
bas
```

Bases: `mythril.laser.ethereum.transaction.transaction_models.BaseTransaction`

Transaction object models an transaction.

```
end (global_state:    mythril.laser.ethereum.state.global_state.GlobalState,  return_data=None,  re-
      revert=False)
```

#### Parameters

- `global_state` –
- `return_data` –
- `revert` –

```
initial_global_state () → mythril.laser.ethereum.state.global_state.GlobalState
Initialize the execution environment.
```

```
class mythril.laser.ethereum.transaction.transaction_models.MessageCallTransaction (*args,
                                                                                       **kwargs)
```

Bases: `mythril.laser.ethereum.transaction.transaction_models.BaseTransaction`

Transaction object models an transaction.

```
end (global_state:    mythril.laser.ethereum.state.global_state.GlobalState,  return_data=None,  re-
      revert=False) → None
```

#### Parameters

- `global_state` –
- `return_data` –
- `revert` –

```
initial_global_state () → mythril.laser.ethereum.state.global_state.GlobalState
Initialize the execution environment.
```



**exception** `mythril.laser.ethereum.transaction.transaction_models.TransactionEndSignal` (*global\_state: GlobalState, mythrilsmt.BitVec, re-vert=False*)

Bases: `Exception`

Exception raised when a transaction is finalized.

**exception** `mythril.laser.ethereum.transaction.transaction_models.TransactionStartSignal` (*transaction: Transaction, global\_state: GlobalState, mythrilsmt.BitVec, re-vert=False*)

Bases: `Exception`

Exception raised when a new transaction is started.

**class** `mythril.laser.ethereum.transaction.transaction_models.TxIdManager`

Bases: `object`

`get_next_tx_id()`

`restart_counter()`

## Module contents

### Submodules

#### `mythril.laser.ethereum.call` module

This module contains the business logic used by `Instruction` in `instructions.py` to get the necessary elements from the stack and determine the parameters for the new global state.

`mythril.laser.ethereum.call.get_call_data` (*global\_state: mythril.laser.ethereum.state.global\_state.GlobalState, memory\_start: Union[int, mythril.laser.smt.bitvec.BitVec], memory\_size: Union[int, mythril.laser.smt.bitvec.BitVec]*)

Gets `call_data` from the `global_state`.

#### Parameters

- **global\_state** – state to look in
- **memory\_start** – Start index
- **memory\_size** – Size

**Returns** Tuple containing: `call_data` array from memory or empty array if symbolic, type found

```
mythril.laser.ethereum.call.get_call_parameters (global_state:
                                                mythril.laser.ethereum.state.global_state.GlobalState,
                                                dynamic_loader:
                                                mythril.support.loader.DynLoader,
                                                with_value=False)
```

Gets call parameters from global state Pops the values from the stack and determines output parameters.

#### Parameters

- **global\_state** – state to look in
- **dynamic\_loader** – dynamic loader to use
- **with\_value** – whether to pop the value argument from the stack

**Returns** callee\_account, call\_data, value, call\_data\_type, gas

```
mythril.laser.ethereum.call.get_callee_account (global_state:
                                                mythril.laser.ethereum.state.global_state.GlobalState,
                                                callee_address: Union[str,
                                                mythril.laser.smt.bitvec.BitVec],
                                                dynamic_loader:
                                                mythril.support.loader.DynLoader)
```

Gets the callees account from the global\_state.

#### Parameters

- **global\_state** – state to look in
- **callee\_address** – address of the callee
- **dynamic\_loader** – dynamic loader to use

**Returns** Account belonging to callee

```
mythril.laser.ethereum.call.get_callee_address (global_state:
                                                mythril.laser.ethereum.state.global_state.GlobalState,
                                                dynamic_loader:
                                                mythril.support.loader.DynLoader,
                                                symbolic_to_address:
                                                mythril.laser.smt.expression.Expression)
```

Gets the address of the callee.

#### Parameters

- **global\_state** – state to look in
- **dynamic\_loader** – dynamic loader to use
- **symbolic\_to\_address** – The (symbolic) callee address

**Returns** Address of the callee

```
mythril.laser.ethereum.call.insert_ret_val (global_state: mythril.laser.ethereum.state.global_state.GlobalState)
mythril.laser.ethereum.call.native_call (global_state: mythril.laser.ethereum.state.global_state.GlobalState,
                                           callee_address: Union[str,
                                           mythril.laser.smt.bitvec.BitVec],
                                           call_data:
                                           mythril.laser.ethereum.state.calldata.BaseCalldata,
                                           memory_out_offset: Union[int,
                                           mythril.laser.smt.expression.Expression],
                                           memory_out_size: Union[int,
                                           mythril.laser.smt.expression.Expression]) → Op-
                                           tional[List[mythril.laser.ethereum.state.global_state.GlobalState]]
```

## mythril.laser.ethereum.cfg module

This module.

```
class mythril.laser.ethereum.cfg.Edge (node_from: int, node_to: int,
                                         edge_type=<JumpType.UNCONDITIONAL: 2>,
                                         condition=None)
```

Bases: object

The representation of a call graph edge.

**as\_dict**

### Returns

```
class mythril.laser.ethereum.cfg.JumpType
```

Bases: enum.Enum

An enum to represent the types of possible JUMP scenarios.

**CALL** = 3

**CONDITIONAL** = 1

**RETURN** = 4

**Transaction** = 5

**UNCONDITIONAL** = 2

```
class mythril.laser.ethereum.cfg.Node (contract_name: str, start_addr=0, constraints=None,
                                         function_name='unknown')
```

Bases: object

The representation of a call graph node.

**get\_cfg\_dict** () → Dict[KT, VT]

### Returns

```
class mythril.laser.ethereum.cfg.NodeFlags
```

Bases: flags.Flags

A collection of flags to denote the type a call graph node can have.

## mythril.laser.ethereum.evm\_exceptions module

This module contains EVM exception types used by LASER.

```
exception mythril.laser.ethereum.evm_exceptions.InvalidInstruction
```

Bases: *mythril.laser.ethereum.evm\_exceptions.VmException*

A VM exception denoting an invalid op code has been encountered.

```
exception mythril.laser.ethereum.evm_exceptions.InvalidJumpDestination
```

Bases: *mythril.laser.ethereum.evm\_exceptions.VmException*

A VM exception regarding JUMPs to invalid destinations.

```
exception mythril.laser.ethereum.evm_exceptions.OutOfGasException
```

Bases: *mythril.laser.ethereum.evm\_exceptions.VmException*

A VM exception denoting the current execution has run out of gas.

**exception** `mythril.laser.ethereum.evm_exceptions.StackOverflowException`  
 Bases: `mythril.laser.ethereum.evm_exceptions.VmException`

A VM exception regarding stack overflows.

**exception** `mythril.laser.ethereum.evm_exceptions.StackUnderflowException`  
 Bases: `IndexError`, `mythril.laser.ethereum.evm_exceptions.VmException`

A VM exception regarding stack underflows.

**exception** `mythril.laser.ethereum.evm_exceptions.VmException`  
 Bases: `Exception`

The base VM exception type.

**exception** `mythril.laser.ethereum.evm_exceptions.WriteProtection`  
 Bases: `mythril.laser.ethereum.evm_exceptions.VmException`

A VM exception denoting that a write operation is executed on a write protected environment

## mythril.laser.ethereum.instruction\_data module

`mythril.laser.ethereum.instruction_data.calculate_native_gas` (*size: int, contract: str*)

### Parameters

- **size** –
- **contract** –

### Returns

`mythril.laser.ethereum.instruction_data.calculate_sha3_gas` (*length: int*)

### Parameters length –

### Returns

`mythril.laser.ethereum.instruction_data.ceil132` (*value: int, \*, ceiling: int = 32*) → `int`

`mythril.laser.ethereum.instruction_data.get_opcode_gas` (*opcode: str*) → `Tuple[int, int]`

`mythril.laser.ethereum.instruction_data.get_required_stack_elements` (*opcode: str*) → `int`

## mythril.laser.ethereum.instructions module

This module contains a representation class for EVM instructions and transitions between them.

**class** `mythril.laser.ethereum.instructions.Instruction` (*op\_code: str, dynamic\_loader: mythril.support.loader.DynLoader, pre\_hooks: List[Callable] = None, post\_hooks: List[Callable] = None*)

Bases: `object`

Instruction class is used to mutate a state according to the current instruction.

**add\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**addmod\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**address\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**and\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**assert\_fail\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**balance\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**basefee\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- `func_obj` –
- `global_state` –

**Returns**

**beginsub\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- `func_obj` –
- `global_state` –

**Returns**

**blockhash\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- `func_obj` –
- `global_state` –

**Returns**

**byte\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- `func_obj` –
- `global_state` –

**Returns**

**call\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- `func_obj` –
- `global_state` –

**Returns**

**call\_post** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- `func_obj` –
- `global_state` –

**Returns**

**callcode\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- `func_obj` –
- `global_state` –

**Returns**

**callcode\_post** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**calldatacopy\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**calldataload\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**calldatasize\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**caller\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**callvalue\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**chainid\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**codecopy\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**codesize\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**coinbase\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**create2\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**create2\_post** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**create\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**



**create\_post** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**delegatecall\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**delegatecall\_post** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**difficulty\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**div\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**dup\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**eq\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**evaluate** (*global\_state*: *mythril.laser.ethereum.state.global\_state.GlobalState*, *post=False*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]  
Performs the mutation for this instruction.

**Parameters**

- **global\_state** –
- **post** –

**Returns**

**exp\_** (*global\_state*: *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**extcodecopy\_** (*global\_state*: *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**extcodehash\_** (*global\_state*: *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**extcodesize\_** (*global\_state*: *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**gas\_** (*global\_state*: *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**gaslimit\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**gasprice\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**gt\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**invalid\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**iszero\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**jump\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**jumpdest\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**jumpi\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**jumpsb\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**log\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**lt\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**mload\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**mod\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**msize\_** (*global\_state*: *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**mstore8\_** (*global\_state*: *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**mstore\_** (*global\_state*: *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**mul\_** (*global\_state*: *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**mulmod\_** (*global\_state*: *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**not\_** (*global\_state*: *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**number\_** (*global\_state*: *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**or\_** (*global\_state*: *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**origin\_** (*global\_state*: *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**pc\_** (*global\_state*: *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**pop\_** (*global\_state*: *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**post\_handler** (*global\_state*, *function\_name*: str)

**push\_** (*global\_state*: *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**return\_** (*global\_state*: *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –

- `global_state` –

#### Returns

**returndatacopy\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

#### Parameters

- `func_obj` –
- `global_state` –

#### Returns

**returndatasize\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

#### Parameters

- `func_obj` –
- `global_state` –

#### Returns

**returnsub\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

#### Parameters

- `func_obj` –
- `global_state` –

#### Returns

**revert\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

#### Parameters

- `func_obj` –
- `global_state` –

#### Returns

**sar\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

#### Parameters

- `func_obj` –
- `global_state` –

#### Returns

**sdiv\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

#### Parameters

- `func_obj` –
- `global_state` –

#### Returns

**selfbalance\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**selfdestruct\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**sgt\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**sha3\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**shl\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**shr\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**signextend\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**



- **func\_obj** –
- **global\_state** –

**Returns**

**sload\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**slt\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**smod\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**sstore\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**staticcall\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**staticcall\_post** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**stop\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**sub\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**swap\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**timestamp\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**xor\_** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*) →  
List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters**

- **func\_obj** –
- **global\_state** –

**Returns**

**class** mythril.laser.ethereum.instructions.**StateTransition** (*increment\_pc=True*,  
*enable\_gas=True*,  
*is\_state\_mutation\_instruction=False*)

Bases: object

Decorator that handles global state copy and original return.

This decorator calls the decorated instruction mutator function on a copy of the state that is passed to it. After the call, the resulting new states' program counter is automatically incremented if *increment\_pc=True*.

**accumulate\_gas** (*global\_state:* *mythril.laser.ethereum.state.global\_state.GlobalState*)

**Parameters** **global\_state** –

**Returns**

**static call\_on\_state\_copy** (*func: Callable, func\_obj: mythril.laser.ethereum.instructions.Instruction, state: mythril.laser.ethereum.state.global\_state.GlobalState*)

**Parameters**

- **func** –
- **func\_obj** –
- **state** –

**Returns**

**static check\_gas\_usage\_limit** (*global\_state: mythril.laser.ethereum.state.global\_state.GlobalState*)

**Parameters global\_state** –

**Returns**

**increment\_states\_pc** (*states: List[mythril.laser.ethereum.state.global\_state.GlobalState]*) → List[mythril.laser.ethereum.state.global\_state.GlobalState]

**Parameters states** –

**Returns**

**mythril.laser.ethereum.instructions.transfer\_ether** (*global\_state: mythril.laser.ethereum.state.global\_state.GlobalState, sender: mythril.laser.smt.bitvec.BitVec, receiver: mythril.laser.smt.bitvec.BitVec, value: Union[int, mythril.laser.smt.bitvec.BitVec]*)

Perform an Ether transfer between two accounts

**Parameters**

- **global\_state** – The global state in which the Ether transfer occurs
- **sender** – The sender of the Ether
- **receiver** – The recipient of the Ether
- **value** – The amount of Ether to send

**Returns**

## mythril.laser.ethereum.natives module

This module defines helper functions to deal with native calls.

**exception** **mythril.laser.ethereum.natives.NativeContractException**

Bases: **Exception**

An exception denoting an error during a native call.

**mythril.laser.ethereum.natives.blake2b\_fcompress** (*data: List[int]*) → List[int]

blake2b hashing ;param data: ;return:

**mythril.laser.ethereum.natives.ec\_add** (*data: List[int]*) → List[int]

**mythril.laser.ethereum.natives.ec\_mul** (*data: List[int]*) → List[int]

**mythril.laser.ethereum.natives.ec\_pair** (*data: List[int]*) → List[int]

`mythril.laser.ethereum.natives.ecrecover` (*data: List[int]*) → List[int]

**Parameters** *data* –

**Returns**

`mythril.laser.ethereum.natives.ecrecover_to_pub` (*rawhash, v, r, s*)

`mythril.laser.ethereum.natives.encode_int32` (*v*)

`mythril.laser.ethereum.natives.identity` (*data: List[int]*) → List[int]

**Parameters** *data* –

**Returns**

`mythril.laser.ethereum.natives.int_to_32bytearray` (*i*)

`mythril.laser.ethereum.natives.mod_exp` (*data: List[int]*) → List[int]

TODO: Some symbolic parts can be handled here Modular Exponentiation :param data: Data with <length\_of\_BASE> <length\_of\_EXPONENT> <length\_of\_MODULUS> <BASE> <EXPONENT> <MODULUS> :return: modular exponentiation

`mythril.laser.ethereum.natives.native_contracts` (*address: int, data: mythril.laser.ethereum.state.calldata.BaseCalldata*) → List[int]

Takes integer address 1, 2, 3, 4.

**Parameters**

- **address** –
- **data** –

**Returns**

`mythril.laser.ethereum.natives.ripemd160` (*data: List[int]*) → List[int]

**Parameters** *data* –

**Returns**

`mythril.laser.ethereum.natives.safe_ord` (*value*)

`mythril.laser.ethereum.natives.sha256` (*data: List[int]*) → List[int]

**Parameters** *data* –

**Returns**

## mythril.laser.ethereum.svm module

This module implements the main symbolic execution engine.

```
class mythril.laser.ethereum.svm.LaserEVM (dynamic_loader=None,          max_depth=inf,
                                           execution_timeout=60,          create_timeout=10,
                                           strategy=<class
'mythril.laser.ethereum.strategy.basic.DepthFirstSearchStrategy'>,
                                           transaction_count=2,          requires_statespace=True,
                                           use_reachability_check=True,    iprof=None,
                                           beam_width=None)
```

Bases: object

The LASER EVM.

Just as Mithril had to be mined at great efforts to provide the Dwarves with their exceptional armour, LASER stands at the heart of Mythril, digging deep in the depths of call graphs, unearthing the most precious symbolic call data, that is then hand-forged into beautiful and strong security issues by the experienced smiths we call detection modules. It is truly a magnificent symbiosis.

**exec** (*create=False, track\_gas=False*) → Optional[List[mythril.laser.ethereum.state.global\_state.GlobalState]]

**Parameters**

- **create** –
- **track\_gas** –

**Returns**

**execute\_state** (*global\_state: mythril.laser.ethereum.state.global\_state.GlobalState*) → Tuple[List[mythril.laser.ethereum.state.global\_state.GlobalState], Optional[str]]  
Execute a single instruction in *global\_state*.

**Parameters** *global\_state* –

**Returns** A list of successor states.

**extend\_strategy** (*extension: abc.ABCMeta, \*\*kwargs*) → None

**handle\_vm\_exception** (*global\_state: mythril.laser.ethereum.state.global\_state.GlobalState, op\_code: str, error\_msg: str*) → List[mythril.laser.ethereum.state.global\_state.GlobalState]

**instr\_hook** (*hook\_type, opcode*) → Callable  
Registers the annotated function with `register_instr_hooks`

**Parameters**

- **hook\_type** – Type of hook pre/post
- **opcode** – The opcode related to the function

**laser\_hook** (*hook\_type: str*) → Callable  
Registers the annotated function with `register_laser_hooks`

**Parameters** *hook\_type* –

**Returns** hook decorator

**manage\_cfg** (*opcode: str, new\_states: List[mythril.laser.ethereum.state.global\_state.GlobalState]*) → None

**Parameters**

- **opcode** –
- **new\_states** –

**post\_hook** (*op\_code: str*) → Callable

**Parameters** *op\_code* –

**Returns**

**pre\_hook** (*op\_code: str*) → Callable

**Parameters** *op\_code* –

**Returns**

**register\_hooks** (*hook\_type: str, hook\_dict: Dict[str, List[Callable]]*)

**Parameters**

- **hook\_type** –
- **hook\_dict** –

**register\_instr\_hooks** (*hook\_type: str, opcode: str, hook: Callable*)

Registers instructions hooks from plugins

**register\_laser\_hooks** (*hook\_type: str, hook: Callable*)

registers the hook with this Laser VM

**sym\_exec** (*world\_state: mythril.laser.ethereum.state.world\_state.WorldState = None, target\_address: int = None, creation\_code: str = None, contract\_name: str = None*) → None

Starts symbolic execution There are two modes of execution. Either we analyze a preconfigured configuration, in which case the `world_state` and `target_address` variables must be supplied. Or we execute the creation code of a contract, in which case the creation code and desired name of that contract should be provided.

:param world\_state The world state configuration from which to perform analysis :param target\_address The address of the contract account in the world state which analysis should target :param creation\_code The creation code to create the target contract in the symbolic environment :param contract\_name The name that the created account should be associated with

**exception** `mythril.laser.ethereum.svm.SVMError`

Bases: `Exception`

An exception denoting an unexpected state in symbolic execution.

### **mythril.laser.ethereum.time\_handler module**

**class** `mythril.laser.ethereum.time_handler.TimeHandler`

Bases: `object`

**start\_execution** (*execution\_time*)

**time\_remaining** ()

### **mythril.laser.ethereum.util module**

This module contains various utility conversion functions and constants for LASER.

`mythril.laser.ethereum.util.bytearray_to_int` (*arr*)

**Parameters** *arr* –

**Returns**

`mythril.laser.ethereum.util.concrete_int_from_bytes` (*concrete\_bytes: Union[List[Union[mythril.laser.smt.bitvec.BitVec, int]], bytes], start\_index: int*) → int

**Parameters**

- **concrete\_bytes** –
- **start\_index** –

**Returns**

`mythril.laser.ethereum.util.concrete_int_to_bytes` (*val*)

**Parameters** *val* –

**Returns**

`mythril.laser.ethereum.util.extract32` (*data: bytearray, i: int*) → int

**Parameters**

- **data** –
- **i** –

**Returns**

`mythril.laser.ethereum.util.extract_copy` (*data: bytearray, mem: bytearray, memstart: int, datastart: int, size: int*)

`mythril.laser.ethereum.util.get_concrete_int` (*item: Union[int, mythril.laser.smt.expression.Expression]*) → int

**Parameters** **item** –**Returns**

`mythril.laser.ethereum.util.get_instruction_index` (*instruction\_list: List[Dict[KT, VT]], address: int*) → Optional[int]

**Parameters**

- **instruction\_list** –
- **address** –

**Returns**

`mythril.laser.ethereum.util.get_trace_line` (*instr: Dict[KT, VT], state: MachineState*) → str

**Parameters**

- **instr** –
- **state** –

**Returns**

`mythril.laser.ethereum.util.pop_bitvec` (*state: MachineState*) → `mythril.laser.smt.bitvec.BitVec`

**Parameters** **state** –**Returns**

`mythril.laser.ethereum.util.safe_decode` (*hex\_encoded\_string: str*) → bytes

**Parameters** **hex\_encoded\_string** –**Returns**

`mythril.laser.ethereum.util.to_signed` (*i: int*) → int

**Parameters** **i** –**Returns**

## Module contents

### mythril.laser.plugin package

#### Subpackages

### mythril.laser.plugin.plugins package

#### Subpackages

### mythril.laser.plugin.plugins.coverage package

#### Submodules

### mythril.laser.plugin.plugins.coverage.coverage\_plugin module

```
class mythril.laser.plugin.plugins.coverage.coverage_plugin.CoveragePluginBuilder
    Bases: mythril.laser.plugin.builder.PluginBuilder

    name = 'coverage'

class mythril.laser.plugin.plugins.coverage.coverage_plugin.InstructionCoveragePlugin
    Bases: mythril.laser.plugin.interface.LaserPlugin

    This plugin measures the instruction coverage of mythril. The instruction coverage is the ratio between the
    instructions that have been executed and the total amount of instructions.

    Note that with lazy constraint solving enabled that this metric will be “unsound” as reachability will not be
    considered for the calculation of instruction coverage.

    initialize (symbolic_vm: mythril.laser.ethereum.svm.LaserEVM)
        Initializes the instruction coverage plugin

        Introduces hooks for each instruction :param symbolic_vm: :return:

    is_instruction_covered (bytecode, index)
```

### mythril.laser.plugin.plugins.coverage.coverage\_strategy module

```
class mythril.laser.plugin.plugins.coverage.coverage_strategy.CoverageStrategy (super_strategy:
mythril.laser.ethereum.strategy.BasicSearchStrategy)
    Bases: mythril.laser.ethereum.strategy.BasicSearchStrategy
    Implements a instruction coverage based search strategy

    This strategy is quite simple and effective, it prioritizes the execution of instructions that have previously been
    uncovered. Once there is no such global state left in the work list, it will resort to using the super_strategy.

    This strategy is intended to be used “on top of” another one
```



**get\_strategic\_global\_state()** → `mythril.laser.ethereum.state.global_state.GlobalState`  
 Returns the first uncovered global state in the work list if it exists, otherwise `super_strategy.get_strategic_global_state()` is returned.

## Module contents

### `mythril.laser.plugin.plugins.summary_backup` package

## Module contents

## Submodules

### `mythril.laser.plugin.plugins.benchmark` module

**class** `mythril.laser.plugin.plugins.benchmark.BenchmarkPlugin` (*name=None*)

Bases: `mythril.laser.plugin.interface.LaserPlugin`

Benchmark Plugin

This plugin aggregates the following information: - duration - code coverage over time - final code coverage - total number of executed instructions

**initialize** (*symbolic\_vm: mythril.laser.ethereum.svm.LaserEVM*)

Initializes the BenchmarkPlugin

Introduces hooks in `symbolic_vm` to track the desired values :param `symbolic_vm`: Symbolic virtual machine to analyze

**class** `mythril.laser.plugin.plugins.benchmark.BenchmarkPluginBuilder`

Bases: `mythril.laser.plugin.builder.PluginBuilder`

**name** = 'benchmark'

### `mythril.laser.plugin.plugins.call_depth_limiter` module

**class** `mythril.laser.plugin.plugins.call_depth_limiter.CallDepthLimit` (*call\_depth\_limit: int*)

Bases: `mythril.laser.plugin.interface.LaserPlugin`

**initialize** (*symbolic\_vm: mythril.laser.ethereum.svm.LaserEVM*)

Initializes the mutation pruner

Introduces hooks for SSTORE operations :param `symbolic_vm`: :return:

**class** `mythril.laser.plugin.plugins.call_depth_limiter.CallDepthLimitBuilder`

Bases: `mythril.laser.plugin.builder.PluginBuilder`

**name** = 'call-depth-limit'

### `mythril.laser.plugin.plugins.dependency_pruner` module

**class** `mythril.laser.plugin.plugins.dependency_pruner.DependencyPruner`

Bases: `mythril.laser.plugin.interface.LaserPlugin`

Dependency Pruner Plugin

For every basic block, this plugin keeps a list of storage locations that are accessed (read) in the execution path containing that block. This map is built up over the whole symbolic execution run.

After the initial build up of the map in the first transaction, blocks are executed only if any of the storage locations written to in the previous transaction can have an effect on that block or any of its successors.

**initialize** (*symbolic\_vm*: *mythril.laser.ethereum.svm.LaserEVM*) → None

Initializes the DependencyPruner

:param symbolic\_vm

**update\_calls** (*path*: *List[int]*) → None

Update the dependency map for the block offsets on the given path.

:param path :param target\_location

**update\_sloads** (*path*: *List[int]*, *target\_location*: *object*) → None

Update the dependency map for the block offsets on the given path.

:param path :param target\_location

**update\_sstores** (*path*: *List[int]*, *target\_location*: *object*) → None

Update the dependency map for the block offsets on the given path.

:param path :param target\_location

**wanna\_execute** (*address*: *int*, *annotation*: *mythril.laser.plugin.plugins.plugin\_annotations.DependencyAnnotation*) → bool

Decide whether the basic block starting at ‘address’ should be executed.

:param address :param storage\_write\_cache

**class** *mythril.laser.plugin.plugins.dependency\_pruner.DependencyPrunerBuilder*

Bases: *mythril.laser.plugin.builder.PluginBuilder*

**name** = 'dependency-pruner'

*mythril.laser.plugin.plugins.dependency\_pruner.get\_dependency\_annotation* (*state*:

*mythril.laser.ethereum.sta*

→

*mythril.laser.plugin.plugin*

Returns a dependency annotation

**Parameters** *state* – A global state object

*mythril.laser.plugin.plugins.dependency\_pruner.get\_ws\_dependency\_annotation* (*state*:

*mythril.laser.ethereum*

→

*mythril.laser.plugin.p*

Returns the world state annotation

**Parameters** *state* – A global state object

## mythril.laser.plugin.plugins.instruction\_profiler module

**class** *mythril.laser.plugin.plugins.instruction\_profiler.InstructionProfiler*

Bases: *mythril.laser.plugin.interface.LaserPlugin*

Performance profile for the execution of each instruction.

**initialize** (*symbolic\_vm*: *mythril.laser.ethereum.svm.LaserEVM*) → None

Initializes this plugin on the symbolic virtual machine

**Parameters** *symbolic\_vm* – symbolic virtual machine to initialize the laser plugin on

```
class mythril.laser.plugin.plugins.instruction_profiler.InstructionProfilerBuilder
    Bases: mythril.laser.plugin.builder.PluginBuilder

    name = 'instruction-profiler'
```

### mythril.laser.plugin.plugins.mutation\_pruner module

```
class mythril.laser.plugin.plugins.mutation_pruner.MutationPruner
    Bases: mythril.laser.plugin.interface.LaserPlugin

    Mutation pruner plugin
```

Let S be a world state from which T is a symbolic transaction, and S' is the resulting world state. In a situation where T does not execute any mutating instructions we can safely abandon further analysis on top of state S'. This is for the reason that we already performed analysis on S, which is equivalent.

This optimization inhibits path explosion caused by “clean” behaviour

**The basic operation of this plugin is as follows:**

- Hook all mutating operations and introduce a MutationAnnotation to the global state on execution of the hook
- Hook the svm EndTransaction on execution filter the states that do not have a mutation annotation

```
initialize (symbolic_vm: mythril.laser.ethereum.svm.LaserEVM)
    Initializes the mutation pruner
```

```
    Introduces hooks for SSTORE operations :param symbolic_vm: :return:
```

```
class mythril.laser.plugin.plugins.mutation_pruner.MutationPrunerBuilder
    Bases: mythril.laser.plugin.builder.PluginBuilder

    name = 'mutation-pruner'
```

### mythril.laser.plugin.plugins.plugin\_annotations module

```
class mythril.laser.plugin.plugins.plugin_annotations.DependencyAnnotation
    Bases: mythril.laser.ethereum.state.annotation.MergeableStateAnnotation

    Dependency Annotation
```

This annotation tracks read and write access to the state during each transaction.

```
check_merge_annotation (other: mythril.laser.plugin.plugins.plugin_annotations.DependencyAnnotation)
```

```
extend_storage_write_cache (iteration: int, value: object)
```

```
get_storage_write_cache (iteration: int)
```

```
merge_annotation (other: mythril.laser.plugin.plugins.plugin_annotations.DependencyAnnotation)
```

```
class mythril.laser.plugin.plugins.plugin_annotations.MutationAnnotation
    Bases: mythril.laser.ethereum.state.annotation.StateAnnotation

    Mutation Annotation
```

This is the annotation used by the MutationPruner plugin to record mutations

```
persist_over_calls
```

If this function returns true then laser will propagate the annotation between calls

The default is set to False

```
class mythril.laser.plugin.plugins.plugin_annotations.WSDependencyAnnotation
```

Bases: *mythril.laser.ethereum.state.annotation.MergeableStateAnnotation*

Dependency Annotation for World state

This world state annotation maintains a stack of state annotations. It is used to transfer individual state annotations from one transaction to the next.

```
check_merge_annotation (annotation: mythril.laser.plugin.plugins.plugin_annotations.WSDependencyAnnotation)  
→ bool
```

```
merge_annotation (annotation: mythril.laser.plugin.plugins.plugin_annotations.WSDependencyAnnotation)  
→ mythril.laser.plugin.plugins.plugin_annotations.WSDependencyAnnotation
```

## Module contents

Plugin implementations

This module contains the implementation of some features

- benchmarking
- pruning

## Submodules

### mythril.laser.plugin.builder module

```
class mythril.laser.plugin.builder.PluginBuilder
```

Bases: *abc.ABC*

The plugin builder interface enables construction of Laser plugins

```
name = 'Default Plugin Name'
```

### mythril.laser.plugin.interface module

```
class mythril.laser.plugin.interface.LaserPlugin
```

Bases: *object*

Base class for laser plugins

Functionality in laser that the symbolic execution process does not need to depend on can be implemented in the form of a laser plugin.

Laser plugins implement the function `initialize(symbolic_vm)` which is called with the laser virtual machine when they are loaded. Regularly a plugin will introduce several hooks into laser in this function

Plugins can direct actions by raising Signals defined in `mythril.laser.ethereum.plugins.signals` For example, a pruning plugin might raise the `PluginSkipWorldState` signal.

```
initialize (symbolic_vm: mythril.laser.ethereum.svm.LaserEVM) → None
```

Initializes this plugin on the symbolic virtual machine

**Parameters** `symbolic_vm` – symbolic virtual machine to initialize the laser plugin on

## mythril.laser.plugin.loader module

**class** `mythril.laser.plugin.loader.LaserPluginLoader`

Bases: `object`

The LaserPluginLoader is used to abstract the logic relating to plugins. Components outside of laser thus don't have to be aware of the interface that plugins provide

**add\_args** (*plugin\_name*, *\*\*kwargs*)

**enable** (*plugin\_name*: *str*)

**instrument\_virtual\_machine** (*symbolic\_vm*: *mythril.laser.ethereum.svm.LaserEVM*,  
*with\_plugins*: *Optional[List[str]]*)

Load enabled plugins into the passed symbolic virtual machine :param symbolic\_vm: The virtual machine to instrument the plugins with :param with\_plugins: Override the globally enabled/disabled builders and load all plugins in the list

**is\_enabled** (*plugin\_name*: *str*) → *bool*

Returns whether the plugin is loaded in the symbolic\_vm

**Parameters** *plugin\_name* – Name of the plugin to check

**load** (*plugin\_builder*: *mythril.laser.plugin.builder.PluginBuilder*) → *None*

Enables a Laser Plugin

**Parameters** *plugin\_builder* – Builder that constructs the plugin

## mythril.laser.plugin.signals module

**exception** `mythril.laser.plugin.signals.PluginSignal`

Bases: `Exception`

Base plugin signal

These signals are used by the laser plugins to create intent for certain actions in the symbolic virtual machine

**exception** `mythril.laser.plugin.signals.PluginSkipState`

Bases: `mythril.laser.plugin.signals.PluginSignal`

Plugin to skip world state

Plugins that raise this signal while the `add_world_state` hook is being executed will force laser to abandon that world state.

**exception** `mythril.laser.plugin.signals.PluginSkipWorldState`

Bases: `mythril.laser.plugin.signals.PluginSignal`

Plugin to skip world state

Plugins that raise this signal while the `add_world_state` hook is being executed will force laser to abandon that world state.

## Module contents

Laser plugins

This module contains everything to do with laser plugins

Laser plugins are a way of extending laser's functionality without complicating the core business logic. Different features that have been implemented in the form of plugins are: - benchmarking - path pruning

Plugins also provide a way to implement optimisations outside of the mythril code base and to inject them. The api that laser currently provides is still unstable and will probably change to suit our needs as more plugins get developed.

For the implementation of plugins the following modules are of interest: - laser.plugins.plugin - laser.plugins.signals - laser.svm

Which show the basic interfaces with which plugins are able to interact

## **mythril.laser.smt package**

### **Subpackages**

## **mythril.laser.smt.solver package**

### **Submodules**

## **mythril.laser.smt.solver.independence\_solver module**

```
class mythril.laser.smt.solver.independence_solver.DependenceBucket (variables=None, conditions=None)
```

Bases: object

Bucket object to contain a set of conditions that are dependent on each other

```
class mythril.laser.smt.solver.independence_solver.DependenceMap
```

Bases: object

DependenceMap object that maintains a set of dependence buckets, used to separate independent smt queries

```
add_condition (condition: z3.z3.BoolRef) → None
```

Add condition to the dependence map :param condition: The condition that is to be added to the dependence map

```
class mythril.laser.smt.solver.independence_solver.IndependenceSolver
```

Bases: object

An SMT solver object that uses independence optimization

```
add (*constraints) → None
```

Adds the constraints to this solver.

**Parameters constraints** – constraints to add

```
append (*constraints) → None
```

Adds the constraints to this solver.

**Parameters constraints** – constraints to add

```
check (**kwargs)
```

```
model () → mythril.laser.smt.model.Model
```

Returns z3 model for a solution.

```
pop (num) → None
```

Pop num constraints from this solver.

```
reset () → None
```

Reset this solver.

**set\_timeout** (*timeout: int*) → None  
 Sets the timeout that will be used by this solver, timeout is in milliseconds.

**Parameters timeout** –

## mythril.laser.smt.solver.solver module

This module contains an abstract SMT representation of an SMT solver.

**class** mythril.laser.smt.solver.solver.**BaseSolver** (*raw: T*)  
 Bases: typing.Generic

**add** (*\*constraints*) → None  
 Adds the constraints to this solver.

**Parameters constraints** –

**Returns**

**append** (*\*constraints*) → None  
 Adds the constraints to this solver.

**Parameters constraints** –

**Returns**

**check** (*\*\*kwargs*)

**model** () → mythril.laser.smt.model.Model  
 Returns z3 model for a solution.

**Returns**

**set\_timeout** (*timeout: int*) → None  
 Sets the timeout that will be used by this solver, timeout is in milliseconds.

**Parameters timeout** –

**sexpr** ()

**class** mythril.laser.smt.solver.solver.**Optimize**  
 Bases: *mythril.laser.smt.solver.solver.BaseSolver*

An optimizing smt solver.

**maximize** (*element: mythril.laser.smt.expression.Expression[z3.z3.ExprRef][z3.z3.ExprRef]*) → None  
 In solving this solver will try to maximize the passed expression.

**Parameters element** –

**minimize** (*element: mythril.laser.smt.expression.Expression[z3.z3.ExprRef][z3.z3.ExprRef]*) → None  
 In solving this solver will try to minimize the passed expression.

**Parameters element** –

**class** mythril.laser.smt.solver.solver.**Solver**  
 Bases: *mythril.laser.smt.solver.solver.BaseSolver*

An SMT solver object.

**pop** (*num: int*) → None  
 Pop num constraints from this solver.

**Parameters num** –

**reset** () → None  
Reset this solver.

## mythril.laser.smt.solver.solver\_statistics module

**class** mythril.laser.smt.solver.solver\_statistics.**SolverStatistics**  
Bases: object  
Solver Statistics Class  
Keeps track of the important statistics around smt queries

mythril.laser.smt.solver.solver\_statistics.**stat\_smt\_query** (*func: Callable*)  
Measures statistics for annotated smt query check function

## Module contents

### Submodules

## mythril.laser.smt.array module

This module contains an SMT abstraction of arrays.

This includes an Array class to implement basic store and set operations, as well as as a K-array, which can be initialized with default values over a certain range.

**class** mythril.laser.smt.array.**Array** (*name: str, domain: int, value\_range: int*)  
Bases: *mythril.laser.smt.array.BaseArray*  
A basic symbolic array.

**class** mythril.laser.smt.array.**BaseArray** (*raw*)  
Bases: object  
Base array type, which implements basic store and set operations.

**substitute** (*original\_expression, new\_expression*)

### Parameters

- **original\_expression** –
- **new\_expression** –

**class** mythril.laser.smt.array.**K** (*domain: int, value\_range: int, value: int*)  
Bases: *mythril.laser.smt.array.BaseArray*  
A basic symbolic array, which can be initialized with a default value.

## mythril.laser.smt.bitvec module

This module provides classes for an SMT abstraction of bit vectors.

**class** mythril.laser.smt.bitvec.**BitVec** (*raw: z3.z3.BitVecRef, annotations: Optional[Set[Any]] = None*)  
Bases: *mythril.laser.smt.expression.Expression*  
A bit vector symbol.



**size()** → int

TODO: documentation

**Returns**

**symbolic**

Returns whether this symbol doesn't have a concrete value.

**Returns**

**value**

Returns the value of this symbol if concrete, otherwise None.

**Returns**

## mythril.laser.smt.bitvec\_helper module

`mythril.laser.smt.bitvec_helper.BVAddNoOverflow` (*a: Union[mythril.laser.smt.bitvec.BitVec, int], b: Union[mythril.laser.smt.bitvec.BitVec, int], signed: bool*) → `mythril.laser.smt.bool.Bool`

Creates predicate that verifies that the addition doesn't overflow.

**Parameters**

- **a** –
- **b** –
- **signed** –

**Returns**

`mythril.laser.smt.bitvec_helper.BVMulNoOverflow` (*a: Union[mythril.laser.smt.bitvec.BitVec, int], b: Union[mythril.laser.smt.bitvec.BitVec, int], signed: bool*) → `mythril.laser.smt.bool.Bool`

Creates predicate that verifies that the multiplication doesn't overflow.

**Parameters**

- **a** –
- **b** –
- **signed** –

**Returns**

`mythril.laser.smt.bitvec_helper.BVSubNoUnderflow` (*a: Union[mythril.laser.smt.bitvec.BitVec, int], b: Union[mythril.laser.smt.bitvec.BitVec, int], signed: bool*) → `mythril.laser.smt.bool.Bool`

Creates predicate that verifies that the subtraction doesn't overflow.

**Parameters**

- **a** –
- **b** –

- **signed** –

**Returns**

`mythril.laser.smt.bitvec_helper.Concat (*args) → mythril.laser.smt.bitvec.BitVec`

Create a concatenation expression.

**Parameters** `args` –**Returns**

`mythril.laser.smt.bitvec_helper.Extract (high: int, low: int, bv: mythril.laser.smt.bitvec.BitVec) → mythril.laser.smt.bitvec.BitVec`

Create an extract expression.

**Parameters**

- **high** –
- **low** –
- **bv** –

**Returns**

`mythril.laser.smt.bitvec_helper.If (a: Union[mythril.laser.smt.bool.Bool, bool], b: Union[mythril.laser.smt.array.BaseArray, mythril.laser.smt.bitvec.BitVec, int], c: Union[mythril.laser.smt.array.BaseArray, mythril.laser.smt.bitvec.BitVec, int]) → Union[mythril.laser.smt.bitvec.BitVec, mythril.laser.smt.array.BaseArray]`

Create an if-then-else expression.

**Parameters**

- **a** –
- **b** –
- **c** –

**Returns**

`mythril.laser.smt.bitvec_helper.LShR (a: mythril.laser.smt.bitvec.BitVec, b: mythril.laser.smt.bitvec.BitVec)`

`mythril.laser.smt.bitvec_helper.SRem (a: mythril.laser.smt.bitvec.BitVec, b: mythril.laser.smt.bitvec.BitVec) → mythril.laser.smt.bitvec.BitVec`

Create a signed remainder expression.

**Parameters**

- **a** –
- **b** –

**Returns**

`mythril.laser.smt.bitvec_helper.Sum (*args) → mythril.laser.smt.bitvec.BitVec`

Create sum expression.

**Returns**

```
mythril.laser.smt.bitvec_helper.UDiv (a: mythril.laser.smt.bitvec.BitVec,
                                         b: mythril.laser.smt.bitvec.BitVec) →
                                         mythril.laser.smt.bitvec.BitVec
```

Create an unsigned division expression.

#### Parameters

- **a** –
- **b** –

#### Returns

```
mythril.laser.smt.bitvec_helper.UGE (a: mythril.laser.smt.bitvec.BitVec,
                                         b: mythril.laser.smt.bitvec.BitVec) →
                                         mythril.laser.smt.bool.Bool
```

Create an unsigned greater than or equal to expression.

#### Parameters

- **a** –
- **b** –

#### Returns

```
mythril.laser.smt.bitvec_helper.UGT (a: mythril.laser.smt.bitvec.BitVec,
                                         b: mythril.laser.smt.bitvec.BitVec) →
                                         mythril.laser.smt.bool.Bool
```

Create an unsigned greater than expression.

#### Parameters

- **a** –
- **b** –

#### Returns

```
mythril.laser.smt.bitvec_helper.ULE (a: mythril.laser.smt.bitvec.BitVec,
                                         b: mythril.laser.smt.bitvec.BitVec) →
                                         mythril.laser.smt.bool.Bool
```

Create an unsigned less than or equal to expression.

#### Parameters

- **a** –
- **b** –

#### Returns

```
mythril.laser.smt.bitvec_helper.ULT (a: mythril.laser.smt.bitvec.BitVec,
                                         b: mythril.laser.smt.bitvec.BitVec) →
                                         mythril.laser.smt.bool.Bool
```

Create an unsigned less than expression.

#### Parameters

- **a** –
- **b** –

#### Returns

```
mythril.laser.smt.bitvec_helper.URem(a: mythril.laser.smt.bitvec.BitVec,  
                                     b: mythril.laser.smt.bitvec.BitVec) →  
                                     mythril.laser.smt.bitvec.BitVec
```

Create an unsigned remainder expression.

**Parameters**

- **a** –
- **b** –

**Returns**

## mythril.laser.smt.bool module

This module provides classes for an SMT abstraction of boolean expressions.

```
mythril.laser.smt.bool.And(*args) → mythril.laser.smt.bool.Bool
```

Create an And expression.

```
class mythril.laser.smt.bool.Bool(raw: T, annotations: Optional[Set[Any]] = None)  
    Bases: mythril.laser.smt.expression.Expression
```

This is a Bool expression.

**is\_false**

Specifies whether this variable can be simplified to false.

**Returns**

**is\_true**

Specifies whether this variable can be simplified to true.

**Returns**

```
substitute (original_expression, new_expression)
```

**Parameters**

- **original\_expression** –
- **new\_expression** –

**value**

Returns the concrete value of this bool if concrete, otherwise None.

**Returns** Concrete value or None

```
mythril.laser.smt.bool.Not(a: mythril.laser.smt.bool.Bool) → mythril.laser.smt.bool.Bool
```

Create a Not expression.

**Parameters** **a** –

**Returns**

```
mythril.laser.smt.bool.Or(*args) → mythril.laser.smt.bool.Bool
```

Create an or expression.

**Parameters**

- **a** –
- **b** –

**Returns**

`mythril.laser.smt.bool.Xor(a: mythril.laser.smt.bool.Bool, b: mythril.laser.smt.bool.Bool) → mythril.laser.smt.bool.Bool`

Create an And expression.

`mythril.laser.smt.bool.is_false(a: mythril.laser.smt.bool.Bool) → bool`  
Returns whether the provided bool can be simplified to false.

**Parameters** `a` –

**Returns**

`mythril.laser.smt.bool.is_true(a: mythril.laser.smt.bool.Bool) → bool`  
Returns whether the provided bool can be simplified to true.

**Parameters** `a` –

**Returns**

## mythril.laser.smt.expression module

This module contains the SMT abstraction for a basic symbol expression.

**class** `mythril.laser.smt.expression.Expression` (`raw: T, annotations: Optional[Set[Any]] = None`)

Bases: `typing.Generic`

This is the base symbol class and maintains functionality for simplification and annotations.

**annotate** (`annotation: Any`) → `None`  
Annotates this expression with the given annotation.

**Parameters** `annotation` –

**annotations**  
Gets the annotations for this expression.

**Returns**

**get\_annotations** (`annotation: Any`)

**simplify** () → `None`  
Simplify this expression.

**size** ()

`mythril.laser.smt.expression.simplify(expression: G) → G`  
Simplify the expression .

**Parameters** `expression` –

**Returns**

## mythril.laser.smt.function module

**class** `mythril.laser.smt.function.Function` (`name: str, domain: List[int], value_range: int`)  
Bases: `object`

An uninterpreted function.

## mythril.laser.smt.model module

**class** `mythril.laser.smt.model.Model` (*models: List[z3.z3.ModelRef] = None*)

Bases: `object`

The model class wraps a z3 model

This implementation allows for multiple internal models, this is required for the use of an independence solver which has models for multiple queries which need an uniform output.

**decls** () → List[z3.z3.ExprRef]

Get the declarations for this model

**eval** (*expression: z3.z3.ExprRef, model\_completion: bool = False*) → Union[None, z3.z3.ExprRef]

Evaluate the expression using this model

### Parameters

- **expression** – The expression to evaluate
- **model\_completion** – Use the default value if the model has no interpretation of the given expression

**Returns** The evaluated expression

## Module contents

**class** `mythril.laser.smt.SymbolFactory`

Bases: `typing.Generic`

A symbol factory provides a default interface for all the components of mythril to create symbols

**static BitVecSym** (*name: str, size: int, annotations: Optional[Set[Any]] = None*) → U

Creates a new bit vector with a symbolic value.

### Parameters

- **name** – The name of the symbolic bit vector
- **size** – The size of the bit vector
- **annotations** – The annotations to initialize the bit vector with

**Returns** The freshly created bit vector

**static BitVecVal** (*value: int, size: int, annotations: Optional[Set[Any]] = None*) → U

Creates a new bit vector with a concrete value.

### Parameters

- **value** – The concrete value to set the bit vector to
- **size** – The size of the bit vector
- **annotations** – The annotations to initialize the bit vector with

**Returns** The freshly created bit vector

**static Bool** (*value: \_\_builtins\_\_.bool, annotations: Optional[Set[Any]] = None*) → T

Creates a Bool with concrete value :param value: The boolean value :param annotations: The annotations to initialize the bool with :return: The freshly created Bool()

**static BoolSym** (*name: str, annotations: Optional[Set[Any]] = None*) → T

Creates a boolean symbol :param name: The name of the Bool variable :param annotations: The annotations to initialize the bool with :return: The freshly created Bool()

## Submodules

### mythril.laser.execution\_info module

**class** mythril.laser.execution\_info.**ExecutionInfo**

Bases: abc.ABC

**as\_dict** ()

Returns a dictionary with the execution info contained in this object

The returned dictionary only uses primitive types.

## Module contents

### 6.1.7 mythril.mythril package

## Submodules

### mythril.mythril.mythril\_analyzer module

**class** mythril.mythril.mythril\_analyzer.**MythrilAnalyzer** (*disassembler: mythril.mythril.mythril\_disassembler.MythrilDisassembler, cmd\_args: argparse.Namespace, strategy: str = 'dfs', address: Optional[str] = None*)

Bases: object

The Mythril Analyzer class Responsible for the analysis of the smart contracts

**dump\_statespace** (*contract: mythril.ethereum.evmcontract.EVMContract = None*) → str

Returns serializable statespace of the contract :param contract: The Contract on which the analysis should be done :return: The serialized state space

**fire\_lasers** (*modules: Optional[List[str]] = None, transaction\_count: Optional[int] = None*) → mythril.analysis.report.Report

#### Parameters

- **modules** – The analysis modules which should be executed
- **transaction\_count** – The amount of transactions to be executed

**Returns** The Report class which contains the all the issues/vulnerabilities

**graph\_html** (*contract: mythril.ethereum.evmcontract.EVMContract = None, enable\_physics: bool = False, phrackify: bool = False, transaction\_count: Optional[int] = None*) → str

#### Parameters

- **contract** – The Contract on which the analysis should be done
- **enable\_physics** – If true then enables the graph physics simulation
- **phrackify** – If true generates Phrack-style call graph

- **transaction\_count** – The amount of transactions to be executed

**Returns** The generated graph in html format

### mythril.mythril.mythril\_config module

**class** mythril.mythril.mythril\_config.**MythrilConfig**

Bases: object

The Mythril Analyzer class Responsible for setup of the mythril environment

**static init\_mythril\_dir**() → str

Initializes the mythril dir and config.ini file :return: The mythril dir's path

**set\_api\_from\_config\_path**() → None

Set the RPC mode based on a given config file.

**set\_api\_infura\_id**(id)

**set\_api\_rpc**(rpc: str = None, rpctl: bool = False) → None

Sets the RPC mode to either of ganache or infura :param rpc: either of the strings - ganache, infura-mainnet, infura-rinkeby, infura-kovan, infura-ropsten

**set\_api\_rpc\_infura**() → None

Set the RPC mode to INFURA on Mainnet.

**set\_api\_rpc\_localhost**() → None

Set the RPC mode to a local instance.

### mythril.mythril.mythril\_disassembler module

**class** mythril.mythril.mythril\_disassembler.**MythrilDisassembler**(eth: Optional[mythril.ethereum.interface.rpc.client.EthereumRPCClient] = None, solc\_version: str = None, solc\_settings\_json: str = None, enable\_online\_lookup: bool = False)

Bases: object

The Mythril Disassembler class Responsible for generating disassembly of smart contracts:

- Compiles solc code from file/onchain
- Can also be used to access onchain storage data

**get\_state\_variable\_from\_storage**(address: str, params: Optional[List[str]] = None) → str

Get variables from the storage :param address: The contract address :param params: The list of parameters  
param types: [position, length] or ["mapping", position, key1, key2, ... ]

or [position, length, array]

**Returns** The corresponding storage slot and its value

**static hash\_for\_function\_signature**(func: str) → str

Return function names corresponding signature hash :param func: function name :return: Its hash signature



**load\_from\_address** (*address: str*) → Tuple[str, mythril.ethereum.evmcontract.EVMContract]

Returns the contract given it's on chain address :param address: The on chain address of a contract :return: tuple(address, contract)

**load\_from\_bytecode** (*code: str, bin\_runtime: bool = False, address: Optional[str] = None*) → Tuple[str, mythril.ethereum.evmcontract.EVMContract]

Returns the address and the contract class for the given bytecode :param code: Bytecode :param bin\_runtime: Whether the code is runtime code or creation code :param address: address of contract :return: tuple(address, Contract class)

**load\_from\_solidity** (*solidity\_files: List[str]*) → Tuple[str, List[mythril.solidity.soliditycontract.SolidityContract]]

**Parameters solidity\_files** – List of solidity\_files

**Returns** tuple of address, contract class list

## Module contents

### 6.1.8 mythril.plugin package

#### Submodules

#### mythril.plugin.discovery module

**class** mythril.plugin.discovery.**PluginDiscovery**

Bases: object

PluginDiscovery class

This plugin implements the logic to discover and build plugins in installed python packages

**build\_plugin** (*plugin\_name: str, plugin\_args: Dict[KT, VT]*) → mythril.plugin.interface.MythrilPlugin

Returns the plugin for the given plugin\_name if it is installed

**get\_plugins** (*default\_enabled=None*) → List[str]

Gets a list of installed mythril plugins

**Parameters default\_enabled** – Select plugins that are enabled by default

**Returns** List of plugin names

**init\_installed\_plugins** ()

**installed\_plugins**

**is\_installed** (*plugin\_name: str*) → bool

Returns whether there is python package with a plugin with plugin\_name

#### mythril.plugin.interface module

**class** mythril.plugin.interface.**MythrilCLIPlugin** (*\*\*kwargs*)

Bases: *mythril.plugin.interface.MythrilPlugin*

MythrilCLIPlugin interface

This interface should be implemented by mythril plugins that aim to add commands to the mythril cli

```
class mythril.plugin.interface.MythrilLaserPlugin (**kwargs)
    Bases: mythril.plugin.interface.MythrilPlugin, mythril.laser.plugin.builder.
    PluginBuilder, abc.ABC
```

Mythril Laser Plugin interface

Plugins of this type are used to instrument the laser EVM

```
class mythril.plugin.interface.MythrilPlugin (**kwargs)
    Bases: object
```

MythrilPlugin interface

Mythril Plugins can be used to extend Mythril in different ways: 1. Extend Laser, in which case the LaserPlugin interface must also be extended 2. Extend Laser with a new search strategy in which case the SearchStrategy needs to be implemented 3. Add an analysis module, in this case the AnalysisModule interface needs to be implemented 4. Add new commands to the Mythril cli, using the MythrilCLIPlugin Interface

```
author = 'Default Author'
name = 'Plugin Name'
plugin_description = 'This is an example plugin description'
plugin_license = 'All rights reserved.'
plugin_type = 'Mythril Plugin'
plugin_version = '0.0.1 '
```

## mythril.plugin.loader module

```
class mythril.plugin.loader.MythrilPluginLoader
    Bases: object
```

MythrilPluginLoader singleton

This object permits loading MythrilPlugin's

```
load (plugin: mythril.plugin.interface.MythrilPlugin)
    Loads the passed plugin
```

This function handles input validation and dispatches loading to type specific loaders. Supported plugin types:

- laser plugins
- detection modules

```
set_args (plugin_name: str, **kwargs)
```

```
exception mythril.plugin.loader.UnsupportedPluginType
    Bases: Exception
```

Raised when a plugin with an unsupported type is loaded

## Module contents

### 6.1.9 mythril.solidity package

#### Submodules

## mythril.solidity.soliditycontract module

This module contains representation classes for Solidity files, contracts and source mappings.

```
class mythril.solidity.soliditycontract.SolidityContract (input_file, name=None,
                                                         solc_settings_json=None,
                                                         solc_binary='solc')
```

Bases: *mythril.ethereum.evmcontract.EVMContract*

Representation of a Solidity contract.

```
static get_full_contract_src_maps (ast: Dict[KT, VT]) → Set[str]
    Takes a solc AST and gets the src mappings for all the contracts defined in the top level of the ast :param
    ast: AST of the contract :return: The source maps
```

```
static get_solc_indices (data: Dict[KT, VT]) → Dict[KT, VT]
    Returns solc file indices
```

```
get_source_info (address, constructor=False)
```

**Parameters**

- **address** –
- **constructor** –

**Returns**

```
static get_sources (indices_data: Dict[KT, VT], source_data: Dict[KT, VT]) → None
    Get source indices mapping
```

```
class mythril.solidity.soliditycontract.SolidityFile (filename: str, data: str,
                                                         full_contract_src_maps:
                                                         Set[str])
```

Bases: object

Representation of a file containing Solidity code.

```
class mythril.solidity.soliditycontract.SourceCodeInfo (filename, lineno, code, map-
                                                         ping)
```

Bases: object

```
class mythril.solidity.soliditycontract.SourceMapping (solidity_file_idx, offset,
                                                         length, lineno, mapping)
```

Bases: object

```
mythril.solidity.soliditycontract.get_contracts_from_file (input_file,
                                                         solc_settings_json=None,
                                                         solc_binary='solc')
```

**Parameters**

- **input\_file** –
- **solc\_settings\_json** –
- **solc\_binary** –

## Module contents

### 6.1.10 mythril.support package

#### Submodules

### **mythril.support.loader module**

This module contains the dynamic loader logic to get on-chain storage data and dependencies.

```
class mythril.support.loader.DynLoader (eth: Optional[mythril.ethereum.interface.rpc.client.EthJsonRpc],  
                                         active=True)
```

Bases: object

The dynamic loader class.

**dynld**

**Parameters** *dependency\_address* –

**Returns**

**read\_balance**

**Parameters** *address* –

**Returns**

**read\_storage**

**Parameters**

- **contract\_address** –
- **index** –

**Returns**

### **mythril.support.lock module**

```
class mythril.support.lock.LockFile (file_name, timeout=100, delay=0.05)
```

Bases: object

Locks files.

**acquire()**

Acquires a lock when possible.

**release()**

Releases the lock

```
exception mythril.support.lock.LockFileException
```

Bases: Exception

### **mythril.support.model module**

```
mythril.support.model.get_model
```

Returns a model based on given constraints as a tuple :param constraints: Tuple of constraints :param minimize: Tuple of minimization conditions :param maximize: Tuple of maximization conditions :param enforce\_execution\_time: Bool variable which enforces –execution-timeout’s time :return:

### **mythril.support.opcodes module**

### **mythril.support.signatures module**

The Mythril function signature database.

```
class mythril.support.signatures.SQLiteDB(path)
```

Bases: object

Simple context manager for sqlite3 databases.

Commits everything at exit.

```
class mythril.support.signatures.SignatureDB(enable_online_lookup: bool = False, path: str = None)
```

Bases: object

```
add(byte_sig: str, text_sig: str) → None
```

Adds a new byte - text signature pair to the database. :param byte\_sig: 4-byte signature string :param text\_sig: resolved text signature :return:

```
get(byte_sig: str, online_timeout: int = 2) → List[str]
```

Get a function text signature for a byte signature 1) try local cache 2) try online lookup (if enabled; if not flagged as unavailable)

#### Parameters

- **byte\_sig** – function signature hash as hexstr
- **online\_timeout** – online lookup timeout

**Returns** list of matching function text signatures

```
import_solidity_file(file_path: str, solc_binary: str = 'solc', solc_settings_json: str = None)
```

Import Function Signatures from solidity source files.

#### Parameters

- **solc\_binary** –
- **solc\_settings\_json** –
- **file\_path** – solidity source code file path

#### Returns

```
static lookup_online(byte_sig: str, timeout: int, proxies=None) → List[str]
```

Lookup function signatures from 4byte.directory.

#### Parameters

- **byte\_sig** – function signature hash as hexstr
- **timeout** – optional timeout for online lookup
- **proxies** – optional proxy servers for online lookup

**Returns** a list of matching function signatures for this hash

```
class mythril.support.signatures.Singleton
```

Bases: type

A metaclass type implementing the singleton pattern.

```
mythril.support.signatures.synchronized(sync_lock)
```

A decorator synchronizing multi-process access to a resource.

## mythril.support.source\_support module

```
class mythril.support.source_support.Source(source_type=None, source_format=None, source_list=None)
```

Bases: object

Class to handle to source data

**get\_source\_from\_contracts\_list** (*contracts*)

get the source data from the contracts list :param contracts: the list of contracts :return:

**get\_source\_index** (*bytecode\_hash: str*) → int

Find the contract index in the list :param bytecode\_hash: The contract hash :return: The index of the contract in the \_source\_hash list

### mythril.support.start\_time module

**class** mythril.support.start\_time.**StartTime**

Bases: object

Maintains the start time of the current contract in execution

### mythril.support.support\_args module

**class** mythril.support.support\_args.**Args**

Bases: object

This module helps in preventing args being sent through multiple of classes to reach any analysis/laser module

### mythril.support.support\_utils module

This module contains utility functions for the Mythril support package.

**class** mythril.support.support\_utils.**Singleton**

Bases: type

A metaclass type implementing the singleton pattern.

mythril.support.support\_utils.**get\_code\_hash**

**Parameters** *code* – bytecode

**Returns** Returns hash of the given bytecode

mythril.support.support\_utils.**rzpad** (*value, total\_length*)

Right zero pad value *x* at least to length *l*.

mythril.support.support\_utils.**sha3** (*value*)

mythril.support.support\_utils.**zpad** (*x, l*)

Left zero pad value *x* at least to length *l*.

### Module contents

## 6.2 Submodules

## 6.3 mythril.exceptions module

This module contains general exceptions used by Mythril.

**exception** `mythril.exceptions.CompilerError`

Bases: `mythril.exceptions.MythrilBaseException`

A Mythril exception denoting an error during code compilation.

**exception** `mythril.exceptions.CriticalError`

Bases: `mythril.exceptions.MythrilBaseException`

A Mythril exception denoting an unknown critical error has been encountered.

**exception** `mythril.exceptions.DetectorNotFoundError`

Bases: `mythril.exceptions.MythrilBaseException`

A Mythril exception denoting attempted usage of a non-existent detection module.

**exception** `mythril.exceptions.IllegalArgumentError`

Bases: `ValueError`

The argument used does not exist

**exception** `mythril.exceptions.MythrilBaseException`

Bases: `Exception`

The Mythril exception base type.

**exception** `mythril.exceptions.NoContractFoundError`

Bases: `mythril.exceptions.MythrilBaseException`

A Mythril exception denoting that a given contract file was not found.

**exception** `mythril.exceptions.SolverTimeoutException`

Bases: `mythril.exceptions.UnsatError`

A Mythril exception denoting the unsatisfiability of a series of constraints.

**exception** `mythril.exceptions.UnsatError`

Bases: `mythril.exceptions.MythrilBaseException`

A Mythril exception denoting the unsatisfiability of a series of constraints.

## 6.4 Module contents





## CHAPTER 7

---

### Indices and Tables

---

- `genindex`
- `modindex`
- `search`



### m

mythril, 107

mythril.analysis, 34

mythril.analysis.analysis\_args, 29

mythril.analysis.call\_helpers, 29

mythril.analysis.callgraph, 29

mythril.analysis.issue\_annotation, 30

mythril.analysis.module, 29

mythril.analysis.module.base, 27

mythril.analysis.module.loader, 28

mythril.analysis.module.module\_helpers, 29

mythril.analysis.module.modules, 27

mythril.analysis.module.modules.arbitrary\_jump, 21

mythril.analysis.module.modules.arbitrary\_write, 22

mythril.analysis.module.modules.delegate\_call, 22

mythril.analysis.module.modules.dependence\_on\_origin, 22

mythril.analysis.module.modules.dependence\_on\_predictable\_vars, 23

mythril.analysis.module.modules.ether\_thief, 23

mythril.analysis.module.modules.exceptions, 24

mythril.analysis.module.modules.external\_calls, 24

mythril.analysis.module.modules.integer, 24

mythril.analysis.module.modules.multiple\_sends, 25

mythril.analysis.module.modules.state\_change\_external\_calls, 25

mythril.analysis.module.modules.suicide, 26

mythril.analysis.module.modules.unchecked\_retval, 26

mythril.analysis.module.modules.user\_assertions, 27

mythril.analysis.module.util, 29

mythril.analysis.ops, 30

mythril.analysis.potential\_issues, 31

mythril.analysis.report, 31

mythril.analysis.security, 32

mythril.analysis.solver, 33

mythril.analysis.swc\_data, 33

mythril.analysis.symbolic, 33

mythril.analysis.traceexplore, 34

mythril.concolic, 35

mythril.concolic.concolic\_execution, 34

mythril.concolic.concrete\_data, 35

mythril.concolic.find\_trace, 35

mythril.disassembler, 37

mythril.disassembler.asm, 35

mythril.disassembler.disassembly, 36

mythril.ethereum, 41

mythril.ethereum.evmcontract, 40

mythril.ethereum.interface, 40

mythril.ethereum.interface.rpc, 40

mythril.ethereum.interface.rpc.base\_client, 37

mythril.ethereum.interface.rpc.client, 38

mythril.ethereum.interface.rpc.constants, 39

mythril.ethereum.interface.rpc.exceptions, 39

mythril.ethereum.interface.rpc.utils, 39

mythril.ethereum.util, 40

mythril.exceptions, 106

mythril.interfaces, 44

mythril.interfaces.cli, 41

mythril.interfaces.epic, 43

mythril.laser, 99

mythril.laser.ethereum, 84

mythril.laser.ethereum.call, 61

mythril.laser.ethereum.cfg, 63	mythril.laser.execution_info, 99
mythril.laser.ethereum.evm_exceptions, 63	mythril.laser.plugin, 89
mythril.laser.ethereum.function_managers, 45	mythril.laser.plugin.builder, 88
mythril.laser.ethereum.function_managers.explicit_function_manager, 44	mythril.laser.plugin.interface, 88
mythril.laser.ethereum.function_managers.keccak_function_manager, 44	mythril.laser.plugin.loader, 89
mythril.laser.ethereum.instruction_data, 64	mythril.laser.plugin.plugins.benchmark, 85
mythril.laser.ethereum.instructions, 64	mythril.laser.plugin.plugins.call_depth_limiter, 85
mythril.laser.ethereum.natives, 79	mythril.laser.plugin.plugins.coverage, 85
mythril.laser.ethereum.state, 53	mythril.laser.plugin.plugins.coverage.coverage_plugin, 84
mythril.laser.ethereum.state.account, 45	mythril.laser.plugin.plugins.coverage.coverage_strategy, 84
mythril.laser.ethereum.state.annotation, 46	mythril.laser.plugin.plugins.dependency_pruner, 85
mythril.laser.ethereum.state.calldata, 46	mythril.laser.plugin.plugins.instruction_profiler, 86
mythril.laser.ethereum.state.constraints, 48	mythril.laser.plugin.plugins.mutation_pruner, 87
mythril.laser.ethereum.state.environment, 48	mythril.laser.plugin.plugins.plugin_annotations, 87
mythril.laser.ethereum.state.global_state, 49	mythril.laser.plugin.plugins.summary_backup, 85
mythril.laser.ethereum.state.machine_state, 50	mythril.laser.plugin.signals, 89
mythril.laser.ethereum.state.memory, 51	mythril.laser.smt, 98
mythril.laser.ethereum.state.return_data, 52	mythril.laser.smt.array, 92
mythril.laser.ethereum.state.world_state, 52	mythril.laser.smt.bitvec, 92
mythril.laser.ethereum.strategy, 56	mythril.laser.smt.bitvec_helper, 93
mythril.laser.ethereum.strategy.basic, 54	mythril.laser.smt.bool, 96
mythril.laser.ethereum.strategy.beam, 55	mythril.laser.smt.expression, 97
mythril.laser.ethereum.strategy.concolic, 55	mythril.laser.smt.function, 97
mythril.laser.ethereum.strategy.extensions, 54	mythril.laser.smt.model, 98
mythril.laser.ethereum.svm, 80	mythril.laser.smt.solver, 92
mythril.laser.ethereum.time_handler, 82	mythril.laser.smt.solver.independence_solver, 90
mythril.laser.ethereum.transaction, 61	mythril.laser.smt.solver.solver, 91
mythril.laser.ethereum.transaction.concolic, 56	mythril.laser.smt.solver.solver_statistics, 92
mythril.laser.ethereum.transaction.symbolic, 58	mythril.mythril, 101
mythril.laser.ethereum.transaction.transformation_model, 59	mythril.mythril.mythril_analyzer, 99
mythril.laser.ethereum.util, 82	mythril.mythril.mythril_config, 100
	mythril.mythril.mythril_disassembler, 100
	mythril.plugin, 102
	mythril.plugin.discovery, 101
	mythril.plugin.interface, 101
	mythril.plugin.loader, 102
	mythril.solidity, 103
	mythril.solidity.soliditycontract, 103
	mythril.support, 106
	mythril.support.loader, 104

`mythril.support.lock`, [104](#)  
`mythril.support.model`, [104](#)  
`mythril.support.opcodes`, [104](#)  
`mythril.support.signatures`, [104](#)  
`mythril.support.source_support`, [105](#)  
`mythril.support.start_time`, [106](#)  
`mythril.support.support_args`, [106](#)  
`mythril.support.support_utils`, [106](#)



## A

AccidentallyKillable (class in *mythril.analysis.module.modules.suicide*), 26  
 Account (class in *mythril.laser.ethereum.state.account*), 45  
 AccountData (class in *mythril.concolic.concrete\_data*), 35  
 accounts (*mythril.laser.ethereum.state.global\_state.GlobalState* attribute), 49  
 accounts (*mythril.laser.ethereum.state.world\_state.WorldState* attribute), 52  
 accounts\_exist\_or\_load() (*mythril.laser.ethereum.state.world\_state.WorldState* method), 52  
 accumulate\_gas() (*mythril.laser.ethereum.instructions.StateTransition* method), 78  
 acquire() (*mythril.support.lock.LockFile* method), 104  
 Actors (class in *mythril.laser.ethereum.transaction.symbolic*), 58  
 add() (*mythril.laser.smt.solver.independence\_solver.IndependenceSolver* method), 90  
 add() (*mythril.laser.smt.solver.solver.BaseSolver* method), 91  
 add() (*mythril.support.signatures.SignatureDB* method), 105  
 add\_() (*mythril.laser.ethereum.instructions.Instruction* method), 64  
 add\_analysis\_args() (in module *mythril.interfaces.cli*), 41  
 add\_annotations() (*mythril.laser.ethereum.state.global\_state.GlobalState* method), 49  
 add\_args() (*mythril.laser.plugin.loader.LaserPluginLoader* method), 89  
 add\_balance() (*mythril.laser.ethereum.state.account.Account* method), 45  
 add\_block\_data() (*mythril.analysis.report.Issue* static method), 32  
 add\_code\_info() (*mythril.analysis.report.Issue* method), 32  
 add\_condition() (*mythril.laser.smt.solver.independence\_solver.IndependenceSolver* method), 90  
 add\_graph\_commands() (in module *mythril.interfaces.cli*), 41  
 addmod\_() (*mythril.laser.ethereum.instructions.Instruction* method), 65  
 address\_() (*mythril.laser.ethereum.instructions.Instruction* method), 65  
 And() (in module *mythril.laser.smt.bool*), 96  
 and\_() (*mythril.laser.ethereum.instructions.Instruction* method), 65  
 annotate() (*mythril.laser.ethereum.state.global\_state.GlobalState* method), 49  
 annotate() (*mythril.laser.ethereum.state.world\_state.WorldState* method), 52  
 annotate() (*mythril.laser.smt.expression.Expression* method), 97  
 annotations (*mythril.laser.ethereum.state.global\_state.GlobalState* attribute), 49  
 annotations (*mythril.laser.ethereum.state.world\_state.WorldState* attribute), 52  
 annotations (*mythril.laser.smt.expression.Expression* attribute), 97  
 ansi() (*mythril.interfaces.epic.LolCat* method), 43  
 append() (*mythril.laser.ethereum.state.constraints.Constraints* method), 48  
 append() (*mythril.laser.ethereum.state.machine\_state.MachineStack* method), 50  
 append() (*mythril.laser.smt.solver.independence\_solver.IndependenceSolver* method), 90  
 append() (*mythril.laser.smt.solver.solver.BaseSolver* method), 91  
 append\_issue() (*mythril.analysis.report.Report* method), 32  
 ArbitraryDelegateCall (class in *mythril.analysis.module.modules.delegatecall*), 22

ArbitraryJump (class in BaseTransaction (class in  
     mythril.analysis.module.modules.arbitrary\_jump),  
     21 mythril.laser.ethereum.transaction.transaction\_models),  
     59  
 ArbitraryStorage (class in BasicConcreteCalldata (class in  
     mythril.analysis.module.modules.arbitrary\_write),  
     22 mythril.laser.ethereum.state.calldata), 47  
 Args (class in mythril.support.support\_args), 106  
 Array (class in mythril.laser.smt.array), 92  
 as\_dict (mythril.analysis.report.Issue attribute), 32  
 as\_dict (mythril.laser.ethereum.cfg.Edge attribute), 63  
 as\_dict (mythril.laser.ethereum.state.account.Account  
     attribute), 45  
 as\_dict (mythril.laser.ethereum.state.environment.Environment  
     attribute), 48  
 as\_dict (mythril.laser.ethereum.state.machine\_state.MachineState  
     attribute), 50  
 as\_dict () (mythril.ethereum.evmcontract.EVMContract  
     method), 40  
 as\_dict () (mythril.laser.execution\_info.ExecutionInfo  
     method), 99  
 as\_json () (mythril.analysis.report.Report method), 32  
 as\_list (mythril.laser.ethereum.state.constraints.Constraints  
     attribute), 48  
 as\_markdown () (mythril.analysis.report.Report  
     method), 32  
 as\_swc\_standard\_format ()  
     (mythril.analysis.report.Report method),  
     32  
 as\_text () (mythril.analysis.report.Report method), 32  
 assert\_fail\_ () (mythril.laser.ethereum.instructions.Instruction  
     method), 65  
 assign\_bytecode ()  
     (mythril.disassembler.disassembly.Disassembly  
     method), 36  
 attacker (mythril.laser.ethereum.transaction.symbolic.actors  
     attribute), 58  
 author (mythril.plugin.interface.MythrilPlugin at-  
     tribute), 102

**B**

BadJsonError, 39  
 BadResponseError, 39  
 BadStatusCodeError, 39  
 balance\_ () (mythril.laser.ethereum.instructions.Instruction  
     method), 65  
 BaseArray (class in mythril.laser.smt.array), 92  
 BaseCalldata (class in  
     mythril.laser.ethereum.state.calldata), 46  
 BaseClient (class in  
     mythril.ethereum.interface.rpc.base\_client), 37  
 basefee\_ () (mythril.laser.ethereum.instructions.Instruction  
     method), 65  
 BaseSolver (class in mythril.laser.smt.solver.solver),  
     91

BasicSearchStrategy (class in  
     mythril.laser.ethereum.strategy), 56  
 BasicSymbolicCalldata (class in  
     mythril.laser.ethereum.state.calldata), 47  
 beam\_priority () (mythril.laser.ethereum.strategy.beam.BeamSearch  
     static method), 55  
 BeamSearch (class in  
     mythril.laser.ethereum.strategy.beam), 55  
 beginsub\_ () (mythril.laser.ethereum.instructions.Instruction  
     method), 66  
 BenchmarkPlugin (class in  
     mythril.laser.plugin.plugins.benchmark),  
     85  
 BenchmarkPluginBuilder (class in  
     mythril.laser.plugin.plugins.benchmark),  
     85  
 BitVec (class in mythril.laser.smt.bitvec), 92  
 BitVecSym () (mythril.laser.smt.SymbolFactory static  
     method), 98  
 BitVecVal () (mythril.laser.smt.SymbolFactory static  
     method), 98  
 blake2b\_fcompress () (in module  
     mythril.laser.ethereum.natives), 79  
 blockhash\_ () (mythril.laser.ethereum.instructions.Instruction  
     method), 66  
 Bool (class in mythril.laser.smt.bool), 96  
 Bool () (mythril.laser.smt.SymbolFactory static  
     method), 98  
 BoolSym () (mythril.laser.smt.SymbolFactory static  
     method), 98  
 BoundedLoopsStrategy (class in  
     mythril.laser.ethereum.strategy.extensions.bounded\_loops),  
     54  
 BreadthFirstSearchStrategy (class in  
     mythril.laser.ethereum.strategy.basic), 54  
 build\_plugin () (mythril.plugin.discovery.PluginDiscovery  
     method), 101  
 BVAddNoOverflow () (in module  
     mythril.laser.smt.bitvec\_helper), 93  
 BVMulNoOverflow () (in module  
     mythril.laser.smt.bitvec\_helper), 93  
 BVSubNoUnderflow () (in module  
     mythril.laser.smt.bitvec\_helper), 93  
 byte\_ () (mythril.laser.ethereum.instructions.Instruction  
     method), 66  
 bytearray\_to\_int () (in module  
     mythril.laser.ethereum.util), 82  
 bytecode\_hash (mythril.ethereum.evmcontract.EVMContract  
     attribute), 40



## C

`calculate_extension_size()` (`mythril.laser.ethereum.state.machine_state.MachineState` method), 50  
`calculate_hash()` (`mythril.laser.ethereum.strategy.extensions.bottlenecked_loops.BoundedLoopsStrategy` static method), 54  
`calculate_memory_gas()` (`mythril.laser.ethereum.state.machine_state.MachineState` method), 50  
`calculate_native_gas()` (in module `mythril.laser.ethereum.instruction_data`), 64  
`calculate_sha3_gas()` (in module `mythril.laser.ethereum.instruction_data`), 64  
`Call` (class in `mythril.analysis.ops`), 30  
`CALL` (`mythril.laser.ethereum.cfg.JumpType` attribute), 63  
`call_()` (`mythril.laser.ethereum.instructions.Instruction` method), 66  
`call_on_state_copy()` (`mythril.laser.ethereum.instructions.StateTransition` static method), 78  
`call_post()` (`mythril.laser.ethereum.instructions.Instruction` method), 66  
`CALLBACK` (`mythril.analysis.module.base.EntryPoint` attribute), 28  
`callcode_()` (`mythril.laser.ethereum.instructions.Instruction` method), 66  
`callcode_post()` (`mythril.laser.ethereum.instructions.Instruction` method), 66  
`calldatacopy_()` (`mythril.laser.ethereum.instructions.Instruction` method), 67  
`calldataload_()` (`mythril.laser.ethereum.instructions.Instruction` method), 67  
`calldatasize` (`mythril.laser.ethereum.state.calldata.BaseCalldata` attribute), 46  
`calldatasize_()` (`mythril.laser.ethereum.instructions.Instruction` method), 67  
`CallDepthLimit` (class in `mythril.laser.plugin.plugins.call_depth_limiter`), 85  
`CallDepthLimitBuilder` (class in `mythril.laser.plugin.plugins.call_depth_limiter`), 85  
`caller_()` (`mythril.laser.ethereum.instructions.Instruction` method), 67  
`callvalue_()` (`mythril.laser.ethereum.instructions.Instruction` method), 67  
`cat()` (`mythril.interfaces.epic.LolCat` method), 43  
`ceil32()` (in module `mythril.laser.ethereum.instruction_data`), 64  
`ceil32()` (in module `mythril.laser.ethereum.state.machine_state`), 51  
`chainid_()` (`mythril.laser.ethereum.instructions.Instruction` method), 67  
`check()` (`mythril.laser.smt.solver.independence_solver.IndependenceSolver` method), 90  
`check()` (`mythril.laser.smt.solver.solver.BaseSolver` method), 91  
`check_completion_criterion()` (`mythril.laser.ethereum.strategy.concolic.ConcolicStrategy` method), 55  
`check_gas()` (`mythril.laser.ethereum.state.machine_state.MachineState` method), 51  
`check_gas_usage_limit()` (`mythril.laser.ethereum.instructions.StateTransition` static method), 79  
`check_merge_annotation()` (`mythril.laser.ethereum.state.annotation.MergeableStateAnnotation` method), 46  
`check_merge_annotation()` (`mythril.laser.plugin.plugins.plugin_annotations.DependencyAnnotation` method), 87  
`check_merge_annotation()` (`mythril.laser.plugin.plugins.plugin_annotations.WSDependencyAnnotation` method), 88  
`check_potential_issues()` (in module `mythril.analysis.potential_issues`), 31  
`clean_hex()` (in module `mythril.ethereum.interface.rpc.utils`), 39  
`close()` (`mythril.ethereum.interface.rpc.client.EthJsonRpc` method), 38  
`codecopy_()` (`mythril.laser.ethereum.instructions.Instruction` method), 68  
`codesize_()` (`mythril.laser.ethereum.instructions.Instruction` method), 68  
`coinbase_()` (`mythril.laser.ethereum.instructions.Instruction` method), 68  
`CompilerError`, 106  
`concat_()` (in module `mythril.laser.smt.bitvec_helper`), 94  
`concolic_execution()` (in module `mythril.concolic.concolic_execution`), 34  
`ConcolicStrategy` (class in `mythril.laser.ethereum.strategy.concolic`), 55  
`CONCRETE` (`mythril.analysis.ops.VarType` attribute), 30  
`concrete()` (`mythril.laser.ethereum.state.calldata.BaseCalldata` method), 46  
`concrete()` (`mythril.laser.ethereum.state.calldata.BasicConcreteCalldata` method), 47  
`concrete()` (`mythril.laser.ethereum.state.calldata.BasicSymbolicCalldata` method), 47  
`concrete()` (`mythril.laser.ethereum.state.calldata.ConcreteCalldata` method), 47  
`concrete()` (`mythril.laser.ethereum.state.calldata.SymbolicCalldata` method), 47

`method`), 47  
`concrete_execution()` (in module `mythril.concolic.find_trace`), 35  
`concrete_int_from_bytes()` (in module `mythril.laser.ethereum.util`), 82  
`concrete_int_to_bytes()` (in module `mythril.laser.ethereum.util`), 82  
`ConcreteCalldata` (class in `mythril.laser.ethereum.state.calldata`), 47  
`ConcreteData` (class in `mythril.concolic.concrete_data`), 35  
`CONDITIONAL` (`mythril.laser.ethereum.cfg.JumpType` attribute), 63  
`ConnectionError`, 39  
`Constraints` (class in `mythril.laser.ethereum.state.constraints`), 48  
`contract_hash_to_address()` (in module `mythril.interfaces.cli`), 41  
`ContractCreationTransaction` (class in `mythril.laser.ethereum.transaction.transaction_model`), 59  
`convert_bv()` (in module `mythril.laser.ethereum.state.memory`), 52  
`copy()` (`mythril.laser.ethereum.state.constraints.Constraints` method), 48  
`count_key()` (`mythril.laser.ethereum.strategy.extensions.bounded_loops.BoundedLoopsStrategy` static method), 54  
  
`CoveragePluginBuilder` (class in `mythril.laser.plugin.plugins.coverage.coverage_plugin`), 84  
`CoverageStrategy` (class in `mythril.laser.plugin.plugins.coverage.coverage_strategy`), 84  
`create2_()` (`mythril.laser.ethereum.instructions.Instruction` method), 68  
`create2_post_()` (`mythril.laser.ethereum.instructions.Instruction` method), 68  
`create_()` (`mythril.laser.ethereum.instructions.Instruction` method), 68  
`create_account()` (`mythril.laser.ethereum.state.world_state.WorldState` method), 52  
`create_analyzer_parser()` (in module `mythril.interfaces.cli`), 41  
`create_concolic_parser()` (in module `mythril.interfaces.cli`), 41  
`create_condition()` (`mythril.laser.ethereum.function_managers.exponent_function_manager.ExponentFunctionManager` method), 44  
`create_conditions()` (`mythril.laser.ethereum.function_managers.keccak_function_manager.KeccakFunctionManager` method), 44  
`create_disassemble_parser()` (in module `mythril.interfaces.cli`), 41  
`create_func_to_hash_parser()` (in module `mythril.interfaces.cli`), 41  
`create_hash_to_addr_parser()` (in module `mythril.interfaces.cli`), 41  
`create_initialized_contract_account()` (`mythril.laser.ethereum.state.world_state.WorldState` method), 53  
`create_keccak_()` (`mythril.laser.ethereum.function_managers.keccak_function_manager.KeccakFunctionManager` method), 44  
`create_post_()` (`mythril.laser.ethereum.instructions.Instruction` method), 68  
`create_read_storage_parser()` (in module `mythril.interfaces.cli`), 42  
`create_safe_functions_parser()` (in module `mythril.interfaces.cli`), 42  
`creation_bytecode_hash` (`mythril.ethereum.evmcontract.EVMContract` attribute), 40  
`creator` (`mythril.laser.ethereum.transaction.symbolic.actors.ActorSearchStrategy` (class in `mythril.laser.ethereum.strategy`), 56  
`CriticalError`, 107  
`current_transaction` (`mythril.laser.ethereum.state.global_state.GlobalState` attribute), 49  
  
**D**  
`decls_()` (`mythril.laser.smt.model.Model` method), 98  
`delegatecall_()` (`mythril.laser.ethereum.instructions.Instruction` method), 69  
`delegatecall_post_()` (`mythril.laser.ethereum.instructions.Instruction` method), 69  
`DependenceBucket` (class in `mythril.laser.smt.solver.independence_solver`), 90  
`DependenceMap` (class in `mythril.laser.smt.solver.independence_solver`), 90  
`DependenceAnnotation` (class in `mythril.laser.plugin.plugins.plugin_annotations`), 87  
`DependencyPruner` (class in `mythril.laser.plugin.plugins.dependency_pruner`), 85  
`DependencyPrunerBuilder` (class in `mythril.laser.plugin.plugins.dependency_pruner`), 86  
`DepthFirstSearchStrategy` (class in `mythril.laser.ethereum.strategy.basic`), 54  
`description` (`mythril.analysis.module.base.DetectionModule` attribute), 27

description (mythril.analysis.module.modules.arbitrary\_jump.ArbitraryJump (in module mythril.laser.ethereum.natives), attribute), 21  
description (mythril.analysis.module.modules.arbitrary\_write.ArbitraryWrite (in module mythril.laser.ethereum.natives), attribute), 22  
description (mythril.analysis.module.modules.delegatecall.ArbitraryDelegateCall (in module mythril.laser.ethereum.natives), attribute), 22  
description (mythril.analysis.module.modules.dependence\_on\_origin.ToOrigin () (in module mythril.laser.ethereum.natives), attribute), 22  
description (mythril.analysis.module.modules.dependence\_on\_predictable\_variables.EnablePredictableVariables (in module mythril.laser.plugin.loader.LaserPluginLoader), attribute), 23  
description (mythril.analysis.module.modules.ether\_thief.EtherThief (in module mythril.laser.ethereum.natives), attribute), 23  
description (mythril.analysis.module.modules.exceptions.Exceptions (in module mythril.laser.ethereum.natives), attribute), 24  
description (mythril.analysis.module.modules.external\_calls.ExternalCall (in module mythril.laser.ethereum.transaction.transaction\_models.ContractCall), attribute), 24  
description (mythril.analysis.module.modules.integer.IntegerArithmetic (in module mythril.laser.ethereum.transaction.transaction\_models.MessageCall), attribute), 24  
description (mythril.analysis.module.modules.multiple\_sends.MultipleSends (in module mythril.analysis.module.base.DetectionModule), attribute), 25  
description (mythril.analysis.module.modules.state\_change\_external\_calls.StateChangeAfterCall (in module mythril.analysis.module.modules.arbitrary\_jump.ArbitraryJump), attribute), 25  
description (mythril.analysis.module.modules.suicide.AccidentallyKillBaby (in module mythril.analysis.module.modules.arbitrary\_write.ArbitraryWrite), attribute), 26  
description (mythril.analysis.module.modules.unchecked\_retval.UncheckedRetVal (in module mythril.analysis.module.modules.delegatecall.ArbitraryDelegateCall), attribute), 26  
description (mythril.analysis.module.modules.user\_assertions.UserAssertions (in module mythril.analysis.module.modules.dependence\_on\_origin.ToOrigin), attribute), 27  
detect\_mode () (in module mythril.interfaces.epic), 43  
DetectionModule (class in mythril.analysis.module.base), 27  
DetectorNotFoundError, 107  
difficulty\_ () (mythril.laser.ethereum.instructions.Instruction (in module mythril.laser.ethereum.instructions.Instruction), method), 69  
disassemble () (in module mythril.disassembler.asm), 35  
Disassembly (class in mythril.disassembler.disassembly), 36  
div\_ () (mythril.laser.ethereum.instructions.Instruction (in module mythril.laser.ethereum.instructions.Instruction), method), 69  
dump\_statespace () (mythril.mythril.mythril\_analyzer.MythrilAnalyzer (in module mythril.mythril.mythril\_analyzer.MythrilAnalyzer), method), 99  
dup\_ () (mythril.laser.ethereum.instructions.Instruction (in module mythril.laser.ethereum.instructions.Instruction), method), 69  
dynld (mythril.support.loader.DynLoader (in module mythril.support.loader), attribute), 104  
DynLoader (class in mythril.support.loader), 104  
  
**E**  
ec\_add () (in module mythril.laser.ethereum.natives), 79  
environment (mythril.analysis.report.Report (in module mythril.analysis.report.Report), attribute), 32

`eq_()` (`mythril.laser.ethereum.instructions.Instruction` `method`), 69  
`eth_blockNumber()` (`mythril.ethereum.interface.rpc.base_client.BaseClient` `method`), 37  
`eth_coinbase()` (`mythril.ethereum.interface.rpc.base_client.BaseClient` `method`), 38  
`eth_getBalance()` (`mythril.ethereum.interface.rpc.base_client.BaseClient` `method`), 38  
`eth_getBlockByNumber()` (`mythril.ethereum.interface.rpc.base_client.BaseClient` `method`), 38  
`eth_getCode()` (`mythril.ethereum.interface.rpc.base_client.BaseClient` `method`), 38  
`eth_getStorageAt()` (`mythril.ethereum.interface.rpc.base_client.BaseClient` `method`), 38  
`eth_getTransactionReceipt()` (`mythril.ethereum.interface.rpc.base_client.BaseClient` `method`), 38  
`ether_to_wei()` (`mythril.ethereum.interface.rpc.utils`), 39  
`EtherThief` (`mythril.analysis.module.modules.ether_thief`), 23  
`EthJsonRpc` (`mythril.ethereum.interface.rpc.client`), 38  
`EthJsonRpcError`, 39  
`eval()` (`mythril.laser.smt.model.Model` `method`), 98  
`evaluate()` (`mythril.laser.ethereum.instructions.Instruction` `method`), 70  
`EVMContract` (`mythril.ethereum.evmcontract`), 40  
`EvmInstruction` (`mythril.disassembler.asm`), 35  
`Exceptions` (`mythril.analysis.module.modules.exceptions`), 24  
`exec()` (`mythril.laser.ethereum.svm.LaserEVM` `method`), 81  
`execute()` (`mythril.analysis.module.base.DetectionModule` `method`), 27  
`execute_command()` (`mythril.interfaces.cli`), 42  
`execute_contract_creation()` (`mythril.laser.ethereum.transaction.concolic`), 56  
`execute_contract_creation()` (`mythril.laser.ethereum.transaction.symbolic`), 58  
`execute_message_call()` (`mythril.laser.ethereum.transaction.concolic`), 57  
`execute_message_call()` (`mythril.laser.ethereum.transaction.symbolic`), 58  
`execute_state()` (`mythril.laser.ethereum.svm.LaserEVM` `method`), 81  
`execute_transaction()` (`mythril.laser.ethereum.transaction.concolic`), 57  
`execute_transaction()` (`mythril.laser.ethereum.transaction.symbolic`), 58  
`ExecutionInfo` (`mythril.analysis.symbolic.SymExecWrapper` `attribute`), 34  
`ExecutionInfo` (`mythril.laser.execution_info`), 99  
`exit_with_error()` (`mythril.interfaces.cli`), 42  
`exp_()` (`mythril.laser.ethereum.instructions.Instruction` `method`), 70  
`ExponentFunctionManager` (`mythril.laser.ethereum.function_managers.exponent_function_manager`), 44  
`Expression` (`mythril.laser.smt.expression`), 97  
`extcodecopy_()` (`mythril.laser.ethereum.instructions.Instruction` `method`), 70  
`extcodehash_()` (`mythril.laser.ethereum.instructions.Instruction` `method`), 70  
`extcodesize_()` (`mythril.laser.ethereum.instructions.Instruction` `method`), 70  
`extend()` (`mythril.laser.ethereum.state.memory.Memory` `method`), 51  
`extend_storage_write_cache()` (`mythril.laser.plugin.plugins.plugin_annotations.DependencyAnnotation` `method`), 87  
`extend_strategy()` (`mythril.laser.ethereum.svm.LaserEVM` `method`), 81  
`ExternalCalls` (`mythril.analysis.module.modules.external_calls`), 24  
`Extract()` (`mythril.laser.smt.bitvec_helper`), 94  
`extract32()` (`mythril.laser.ethereum.util`), 83  
`extract_binary()` (`mythril.ethereum.util`), 40  
`extract_copy()` (`mythril.laser.ethereum.util`), 83  
`extract_edges()` (`mythril.analysis.callgraph`), 29  
`extract_nodes()` (`mythril.analysis.callgraph`), 29  
`extract_version()` (`mythril.ethereum.util`), 40



## F

`find_concrete_keccak()` (`mythril.laser.ethereum.function_managers.keccak_function_manager.KeccakFunctionManager` static method), 44

`find_op_code_sequence()` (in module `mythril.disassembler.asm`), 36

`fire_lasers()` (in module `mythril.analysis.security`), 32

`fire_lasers()` (`mythril.mythril.mythril_analyzer.MythrilAnalyzer` method), 99

`flip_branches()` (in module `mythril.concolic.concolic_execution`), 34

`Function` (class in `mythril.laser.smt.function`), 97

`get_concrete_hash_data()` (`mythril.laser.ethereum.function_managers.keccak_function_manager.KeccakFunctionManager` method), 44

`get_concrete_int()` (in module `mythril.laser.ethereum.util`), 83

`get_contracts_from_file()` (in module `mythril.solidity.soliditycontract`), 103

`get_creation_easm()` (`mythril.ethereum.evmcontract.EVMContract` method), 40

`get_creation_input_parser()` (in module `mythril.interfaces.cli`), 42

`get_current_instruction()` (`mythril.laser.ethereum.state.global_state.GlobalState` method), 49

## G

`gas_()` (`mythril.laser.ethereum.instructions.Instruction` method), 70

`gaslimit_()` (`mythril.laser.ethereum.instructions.Instruction` method), 71

`gasprice_()` (`mythril.laser.ethereum.instructions.Instruction` method), 71

`generate_function_constraints()` (in module `mythril.laser.ethereum.transaction.symbolic`), 58

`generate_graph()` (in module `mythril.analysis.callgraph`), 30

`get()` (`mythril.support.signatures.SignatureDB` method), 105

`get_all_constraints()` (`mythril.laser.ethereum.state.constraints.Constraints` method), 48

`get_annotations()` (`mythril.laser.ethereum.state.global_state.GlobalState` method), 49

`get_annotations()` (`mythril.laser.ethereum.state.world_state.WorldState` method), 53

`get_annotations()` (`mythril.laser.smt.expression.Expression` method), 97

`get_call_data()` (in module `mythril.laser.ethereum.call`), 61

`get_call_from_state()` (in module `mythril.analysis.call_helpers`), 29

`get_call_parameters()` (in module `mythril.laser.ethereum.call`), 61

`get_callee_account()` (in module `mythril.laser.ethereum.call`), 62

`get_callee_address()` (in module `mythril.laser.ethereum.call`), 62

`get_cfg_dict()` (`mythril.laser.ethereum.cfg.Node` method), 63

`get_code_hash` (in module `mythril.support.support_utils`), 106

`get_dependency_annotation()` (in module `mythril.laser.plugin.plugins.dependency_pruner`), 86

`get_detection_module_hooks()` (in module `mythril.analysis.module.util`), 29

`get_detection_modules()` (`mythril.analysis.module.loader.ModuleLoader` method), 28

`get_easm()` (`mythril.disassembler.disassembly.Disassembly` method), 36

`get_easm()` (`mythril.ethereum.evmcontract.EVMContract` method), 40

`get_empty_keccak_hash()` (`mythril.laser.ethereum.function_managers.keccak_function_manager.KeccakFunctionManager` static method), 45

`get_full_contract_src_maps()` (`mythril.solidity.soliditycontract.SolidityContract` static method), 103

`get_function()` (`mythril.laser.ethereum.function_managers.keccak_function_manager.KeccakFunctionManager` method), 45

`get_function_info()` (in module `mythril.disassembler.disassembly`), 37

`get_indexed_address()` (in module `mythril.ethereum.util`), 40

`get_instruction_index()` (in module `mythril.laser.ethereum.util`), 83

`get_issue()` (`mythril.analysis.module.modules.state_change_external_state_change_external` method), 26

`get_loop_count()` (`mythril.laser.ethereum.strategy.extensions.bounded_execution_bounded_execution` static method), 54

`get_model` (in module `mythril.support.model`), 104

`get_next_tx_id()` (`mythril.laser.ethereum.transaction.transaction_manager.transaction_manager` method), 61

`get_opcode_from_name()` (in module `mythril.disassembler.asm`), 36

`get_opcode_gas()` (in module `mythril.laser.ethereum.instruction_data`), 64

`get_output_parser()` (in module `mythril.laser.ethereum.instruction_data`), 64

[mythril.interfaces.cli](#)), 42  
[get\\_plugins\(\)](#) ([mythril.plugin.discovery.PluginDiscovery](#) [method](#)), 101  
[get\\_potential\\_issues\\_annotation\(\)](#) (in [module](#) [mythril.analysis.potential\\_issues](#)), 31  
[get\\_random\\_address\(\)](#) (in [module](#) [mythril.ethereum.util](#)), 40  
[get\\_required\\_stack\\_elements\(\)](#) (in [module](#) [mythril.laser.ethereum.instruction\\_data](#)), 64  
[get\\_rpc\\_parser\(\)](#) (in [module](#) [mythril.interfaces.cli](#)), 42  
[get\\_runtime\\_input\\_parser\(\)](#) (in [module](#) [mythril.interfaces.cli](#)), 42  
[get\\_safe\\_functions\\_parser\(\)](#) (in [module](#) [mythril.interfaces.cli](#)), 42  
[get\\_serializable\\_statespace\(\)](#) (in [module](#) [mythril.analysis.traceexplore](#)), 34  
[get\\_solc\\_indices\(\)](#) ([mythril.solidity.soliditycontract.SolidityContract](#) [static method](#)), 103  
[get\\_solc\\_json\(\)](#) (in [module](#) [mythril.ethereum.util](#)), 41  
[get\\_source\\_from\\_contracts\\_list\(\)](#) ([mythril.support.source\\_support.Source](#) [method](#)), 106  
[get\\_source\\_index\(\)](#) ([mythril.support.source\\_support.Source](#) [method](#)), 106  
[get\\_source\\_info\(\)](#) ([mythril.solidity.soliditycontract.SolidityContract](#) [method](#)), 103  
[get\\_sources\(\)](#) ([mythril.solidity.soliditycontract.SolidityContract](#) [static method](#)), 103  
[get\\_state\\_variable\\_from\\_storage\(\)](#) ([mythril.mythril.mythril\\_disassembler.MythrilDisassembler](#) [method](#)), 100  
[get\\_storage\\_write\\_cache\(\)](#) ([mythril.laser.plugin.plugins.plugin\\_annotations.DependencyAnnotation](#) [method](#)), 87  
[get\\_strategic\\_global\\_state\(\)](#) ([mythril.laser.ethereum.strategy.basic.BreadthFirstSearchStrategy](#) [method](#)), 54  
[get\\_strategic\\_global\\_state\(\)](#) ([mythril.laser.ethereum.strategy.basic.DepthFirstSearchStrategy](#) [method](#)), 54  
[get\\_strategic\\_global\\_state\(\)](#) ([mythril.laser.ethereum.strategy.basic.ReturnRandomNaiveStrategy](#) [method](#)), 55  
[get\\_strategic\\_global\\_state\(\)](#) ([mythril.laser.ethereum.strategy.basic.ReturnWeightedRandomStrategy](#) [method](#)), 55  
[get\\_strategic\\_global\\_state\(\)](#) ([mythril.laser.ethereum.strategy.BasicSearchStrategy](#) [method](#)), 56  
[get\\_strategic\\_global\\_state\(\)](#) ([mythril.laser.ethereum.strategy.beam.BeamSearch](#) [method](#)), 55  
[get\\_strategic\\_global\\_state\(\)](#) ([mythril.laser.ethereum.strategy.concolic.ConcolicStrategy](#) [method](#)), 55  
[get\\_strategic\\_global\\_state\(\)](#) ([mythril.laser.ethereum.strategy.CriterionSearchStrategy](#) [method](#)), 56  
[get\\_strategic\\_global\\_state\(\)](#) ([mythril.laser.ethereum.strategy.extensions.bounded\\_loops.BoundedLoopsStrategy](#) [method](#)), 54  
[get\\_strategic\\_global\\_state\(\)](#) ([mythril.laser.plugin.plugins.coverage.coverage\\_strategy.CoverageStrategy](#) [method](#)), 84  
[get\\_trace\\_line\(\)](#) (in [module](#) [mythril.laser.ethereum.util](#)), 83  
[get\\_transaction\\_sequence\(\)](#) (in [module](#) [mythril.analysis.solver](#)), 33  
[get\\_utilities\\_parser\(\)](#) (in [module](#) [mythril.interfaces.cli](#)), 42  
[get\\_variable\(\)](#) (in [module](#) [mythril.analysis.ops](#)), 31  
[get\\_word\\_at\(\)](#) ([mythril.laser.ethereum.state.calldata.BaseCalldata](#) [method](#)), 47  
[get\\_word\\_at\(\)](#) ([mythril.laser.ethereum.state.memory.Memory](#) [method](#)), 51  
[get\\_ws\\_dependency\\_annotation\(\)](#) (in [module](#) [mythril.laser.plugin.plugins.dependency\\_pruner](#)), 86  
[GlobalState](#) (class in [mythril.laser.ethereum.state.global\\_state](#)), 49  
[graph\\_html\(\)](#) ([mythril.mythril.mythril\\_analyzer.MythrilAnalyzer](#) [method](#)), 99  
[has\\_instruction\(\)](#) ([mythril.laser.ethereum.instructions.Instruction](#) [method](#)), 71  
[handle\\_vm\\_exception\(\)](#) ([mythril.laser.ethereum.svm.LaserEVM](#) [method](#)), 81  
[hash\\_for\\_function\\_signature\(\)](#) ([mythril.mythril.mythril\\_disassembler.MythrilDisassembler](#) [static method](#)), 100  
[hash\\_matcher](#) ([mythril.laser.ethereum.function\\_managers.keccak\\_function\\_manager.KeccakFunctionManager](#) [attribute](#)), 45  
[hex\\_encode\(\)](#) (in [module](#) [mythril.ethereum.interface.rpc.utils](#)), 39  
[identity\(\)](#) (in [module](#) [mythril.laser.ethereum.natives](#)), 80  
[if\(\)](#) (in [module](#) [mythril.laser.smt.bitvec\\_helper](#)), 94  
[IllegalArgumentError](#), 107

---

```

import_solidity_file()
    (mythril.support.signatures.SignatureDB
     method), 105
increment_states_pc()
    (mythril.laser.ethereum.instructions.StateTransition
     method), 79
IndependenceSolver (class in
    mythril.laser.smt.solver.independence_solver),
    90
init_installed_plugins()
    (mythril.plugin.discovery.PluginDiscovery
     method), 101
init_mythril_dir()
    (mythril.mythril.mythril_config.MythrilConfig
     static method), 100
initial_global_state()
    (mythril.laser.ethereum.transaction.transaction_models.BaseTransaction
     method), 59
initial_global_state()
    (mythril.laser.ethereum.transaction.transaction_models.ContractOrMythrilTransaction
     method), 60
initial_global_state()
    (mythril.laser.ethereum.transaction.transaction_models.MessageOrEVMTransaction, 63
     method), 60
initial_global_state_from_environment()
    (mythril.laser.ethereum.transaction.transaction_models.BaseTransaction
     method), 59
initialize() (mythril.laser.plugin.interface.LaserPlugin
    method), 88
initialize() (mythril.laser.plugin.plugins.benchmark.BenchmarkPlugin in module mythril.laser.smt.bool), 97
    method), 85
initialize() (mythril.laser.plugin.plugins.call_depth_limiter.CallDepthLimiter in module mythril.laser.smt.bool), 97
    method), 85
initialize() (mythril.laser.plugin.plugins.coverage.coverage_plugin.LaserPluginCoverage.coverage_plugin.InstructionCoveragePlugin
    method), 84
initialize() (mythril.laser.plugin.plugins.dependency_pruner.DependencyPruner in module mythril.laser.ethereum.state.constraints.Constraints
    method), 86
initialize() (mythril.laser.plugin.plugins.instruction_pruner.InstructionPruner in module mythril.analysis.module.module_helpers),
    method), 86
initialize() (mythril.laser.plugin.plugins.mutation_pruner.MutationPruner in module mythril.analysis.module.module_helpers),
    method), 87
InitialState (class in
    mythril.concolic.concrete_data), 35
insert_ret_val() (in module
    mythril.laser.ethereum.call), 62
installed_plugins
    (mythril.plugin.discovery.PluginDiscovery
     attribute), 101
instr_hook() (mythril.laser.ethereum.svm.LaserEVM
    method), 81
Instruction (class in
    mythril.laser.ethereum.instructions), 64
instruction (mythril.laser.ethereum.state.global_state.GlobalState
    attribute), 49
instruction_list_to_easm() (in module
    mythril.disassembler.asm), 36
InstructionCoveragePlugin (class in
    mythril.laser.plugin.plugins.coverage.coverage_plugin),
    84
InstructionProfiler (class in
    mythril.laser.plugin.plugins.instruction_profiler),
    86
InstructionProfilerBuilder (class in
    mythril.laser.plugin.plugins.instruction_profiler),
    87
instrument_virtual_machine()
    (mythril.laser.plugin.loader.LaserPluginLoader
     method), 89
int_to_32bytearray() (in module
    mythril.laser.ethereum.natives), 80
IntegerBaseArithmetic (class in
    mythril.analysis.module.modules.integer),
    24
InvalidInstruction, 63
InvalidMessageOrEVMTransaction, 63
is_assertion_failure() (in module
    mythril.analysis.module.modules.exceptions),
    24
is_enabled() (mythril.laser.plugin.loader.LaserPluginLoader
    method), 89
is_false (mythril.laser.smt.bool.Bool attribute), 96
is_benchmark_plugin (in module mythril.laser.smt.bool), 97
is_installed() (mythril.plugin.discovery.PluginDiscovery
    method), 101
is_instruction_covered()
    (mythril.laser.plugin.plugins.coverage.coverage_plugin.InstructionCoveragePlugin
     method), 84
is_message_or_evm_transaction (in module mythril.laser.ethereum.state.constraints.Constraints
    method), 48
is_profile_instruction (in module mythril.analysis.module.module_helpers),
    86
is_sequence_match() (in module
    mythril.disassembler.asm), 36
is_true (mythril.laser.smt.bool.Bool attribute), 96
is_true() (in module mythril.laser.smt.bool), 97
is_unique_jumpdest() (in module
    mythril.analysis.module.modules.arbitrary_jump),
    22
Issue (class in mythril.analysis.report), 31
IssueAnnotation (class in
    mythril.analysis.issue_annotation), 30
iszero_() (mythril.laser.ethereum.instructions.Instruction
    method), 71

```

## J

`jump_()` (*mythril.laser.ethereum.instructions.Instruction* method), 71

`jumpdest_()` (*mythril.laser.ethereum.instructions.Instruction* method), 71

`JumpdestCountAnnotation` (class in *mythril.laser.ethereum.state.machine\_state*), 50

`JumpdestLoopAnnotation` (class in *mythril.laser.ethereum.state.machine\_state*), 50

`jumpi_()` (*mythril.laser.ethereum.instructions.Instruction* method), 72

`jumpsub_()` (*mythril.laser.ethereum.instructions.Instruction* method), 72

`JumpType` (class in *mythril.laser.ethereum.cfg*), 63

## K

`K` (class in *mythril.laser.smt.array*), 92

`KeccakFunctionManager` (class in *mythril.laser.ethereum.function\_managers.keccak\_function\_manager*), 44

## L

`laser_hook_()` (*mythril.laser.ethereum.svm.LaserEVM* method), 81

`LaserEVM` (class in *mythril.laser.ethereum.svm*), 80

`LaserPlugin` (class in *mythril.laser.plugin.interface*), 88

`LaserPluginLoader` (class in *mythril.laser.plugin.loader*), 89

`LastJumpAnnotation` (class in *mythril.analysis.module.modules.exceptions*), 24

`load_()` (*mythril.laser.plugin.loader.LaserPluginLoader* method), 89

`load_()` (*mythril.plugin.loader.MythrilPluginLoader* method), 102

`load_code_()` (in module *mythril.interfaces.cli*), 42

`load_from_address_()` (*mythril.mythril.mythril\_disassembler.MythrilDisassembler* method), 100

`load_from_bytecode_()` (*mythril.mythril.mythril\_disassembler.MythrilDisassembler* method), 101

`load_from_solidity_()` (*mythril.mythril.mythril\_disassembler.MythrilDisassembler* method), 101

`LockFile` (class in *mythril.support.lock*), 104

`LockFileException`, 104

`log_()` (*mythril.laser.ethereum.instructions.Instruction* method), 72

`LolCat` (class in *mythril.interfaces.epic*), 43

`lookup_online_()` (*mythril.support.signatures.SignatureDB* static method), 105

`LShR_()` (in module *mythril.laser.smt.bitvec\_helper*), 94

`lt_()` (*mythril.laser.ethereum.instructions.Instruction* method), 72

## M

`MachineStack` (class in *mythril.laser.ethereum.state.machine\_state*), 50

`MachineState` (class in *mythril.laser.ethereum.state.machine\_state*), 50

`main_()` (in module *mythril.interfaces.cli*), 42

`manage_cfg_()` (*mythril.laser.ethereum.svm.LaserEVM* method), 81

`matches_expression_()` (*mythril.ethereum.evmcontract.EVMContract* method), 40

`maximize_()` (*mythril.laser.smt.solver.solver.Optimize* method), 91

`mem_extend_()` (*mythril.laser.ethereum.state.machine\_state.MachineState* method), 51

`Memory` (class in *mythril.laser.ethereum.state.memory*), 51

`memory_size_()` (*mythril.laser.ethereum.state.machine\_state.MachineState* attribute), 51

`memory_write_()` (*mythril.laser.ethereum.state.machine\_state.MachineState* method), 51

`merge_annotation_()` (*mythril.laser.ethereum.state.annotation.MergeableStateAnnotation* method), 46

`merge_annotation_()` (*mythril.laser.plugin.plugins.plugin\_annotations.DependencyAnnotation* method), 87

`merge_annotation_()` (*mythril.laser.plugin.plugins.plugin\_annotations.WSDependencyAnnotation* method), 88

`MergeableStateAnnotation` (class in *mythril.laser.ethereum.state.annotation*), 46

`MessageCallTransaction` (class in *mythril.laser.ethereum.transaction.transaction\_models*), 60

`minimize_()` (*mythril.laser.smt.solver.solver.Optimize* method), 91

`mload_()` (*mythril.laser.ethereum.instructions.Instruction* method), 72

`mod_()` (*mythril.laser.ethereum.instructions.Instruction* method), 72

`mod_exp_()` (in module *mythril.laser.ethereum.natives*), 80

`Model` (class in *mythril.laser.smt.model*), 98

`model_()` (*mythril.laser.smt.solver.independence\_solver.IndependenceSolver* method), 90

`model_()` (*mythril.laser.smt.solver.solver.BaseSolver* method), 91

`ModuleLoader` (class in *mythril.analysis.module.loader*), 28



[msize\\_\(\) \(mythril.laser.ethereum.instructions.Instruction method\), 73](#)  
[mstore8\\_\(\) \(mythril.laser.ethereum.instructions.Instruction method\), 73](#)  
[mstore\\_\(\) \(mythril.laser.ethereum.instructions.Instruction method\), 73](#)  
[mul\\_\(\) \(mythril.laser.ethereum.instructions.Instruction method\), 73](#)  
[mulmod\\_\(\) \(mythril.laser.ethereum.instructions.Instruction method\), 73](#)  
[MultipleSends \(class in mythril.analysis.module.modules.multiple\\_sends\), 25](#)  
[MultipleSendsAnnotation \(class in mythril.analysis.module.modules.multiple\\_sends\), 25](#)  
[MutationAnnotation \(class in mythril.laser.plugin.plugins.plugin\\_annotations\), 87](#)  
[MutationPruner \(class in mythril.laser.plugin.plugins.mutation\\_pruner\), 87](#)  
[MutationPrunerBuilder \(class in mythril.laser.plugin.plugins.mutation\\_pruner\), 87](#)  
[mythril \(module\), 107](#)  
[mythril.analysis \(module\), 34](#)  
[mythril.analysis.analysis\\_args \(module\), 29](#)  
[mythril.analysis.call\\_helpers \(module\), 29](#)  
[mythril.analysis.callgraph \(module\), 29](#)  
[mythril.analysis.issue\\_annotation \(module\), 30](#)  
[mythril.analysis.module \(module\), 29](#)  
[mythril.analysis.module.base \(module\), 27](#)  
[mythril.analysis.module.loader \(module\), 28](#)  
[mythril.analysis.module.module\\_helpers \(module\), 29](#)  
[mythril.analysis.module.modules \(module\), 27](#)  
[mythril.analysis.module.modules.arbitrary\\_jump \(module\), 21](#)  
[mythril.analysis.module.modules.arbitrary\\_write \(module\), 22](#)  
[mythril.analysis.module.modules.delegatecall \(module\), 22](#)  
[mythril.analysis.module.modules.dependence\\_on\\_arg \(module\), 22](#)  
[mythril.analysis.module.modules.dependence\\_on\\_arg1 \(module\), 23](#)  
[mythril.analysis.module.modules.ether\\_throw \(module\), 23](#)  
[mythril.analysis.module.modules.exceptions \(module\), 23](#)  
[mythril.analysis.module.modules.external\\_calls \(module\), 24](#)  
[mythril.analysis.module.modules.integer \(module\), 24](#)  
[mythril.analysis.module.modules.multiple\\_sends \(module\), 25](#)  
[mythril.analysis.module.modules.state\\_change\\_external \(module\), 25](#)  
[mythril.analysis.module.modules.suicide \(module\), 25](#)  
[mythril.analysis.module.modules.unchecked\\_retval \(module\), 25](#)  
[mythril.analysis.module.modules.user\\_assertions \(module\), 27](#)  
[mythril.analysis.module.util \(module\), 29](#)  
[mythril.analysis.ops \(module\), 30](#)  
[mythril.analysis.potential\\_issues \(module\), 31](#)  
[mythril.analysis.report \(module\), 31](#)  
[mythril.analysis.security \(module\), 32](#)  
[mythril.analysis.solver \(module\), 33](#)  
[mythril.analysis.swc\\_data \(module\), 33](#)  
[mythril.analysis.symbolic \(module\), 33](#)  
[mythril.analysis.traceexplore \(module\), 34](#)  
[mythril.concolic \(module\), 35](#)  
[mythril.concolic.concolic\\_execution \(module\), 34](#)  
[mythril.concolic.concrete\\_data \(module\), 35](#)  
[mythril.concolic.find\\_trace \(module\), 35](#)  
[mythril.disassembler \(module\), 37](#)  
[mythril.disassembler.asm \(module\), 35](#)  
[mythril.disassembler.disassembly \(module\), 36](#)  
[mythril.ethereum \(module\), 41](#)  
[mythril.ethereum.evmcontract \(module\), 40](#)  
[mythril.ethereum.interface \(module\), 40](#)  
[mythril.ethereum.interface.rpc \(module\), 40](#)  
[mythril.ethereum.interface.rpc.base\\_client \(module\), 37](#)  
[mythril.ethereum.interface.rpc.client \(module\), 38](#)  
[mythril.ethereum.interface.rpc.constants \(module\), 39](#)  
[mythril.ethereum.interface.rpc.exceptions \(module\), 39](#)  
[mythril.ethereum.interface.rpc.utils \(module\), 39](#)  
[mythril.ethereum.util \(module\), 40](#)  
[mythril.exceptions \(module\), 106](#)  
[mythril.interfaces \(module\), 44](#)  
[mythril.interfaces.cli \(module\), 41](#)

mythril.interfaces.epic (module), 43	mythril.laser.smt.solver.independence_solver (module), 90
mythril.laser (module), 99	mythril.laser.smt.solver.solver (module), 91
mythril.laser.ethereum (module), 84	mythril.laser.smt.solver.solver_statistics (module), 92
mythril.laser.ethereum.call (module), 61	
mythril.laser.ethereum.cfg (module), 63	
mythril.laser.ethereum.evm_exceptions (module), 63	
mythril.laser.ethereum.function_managers (module), 45	
mythril.laser.ethereum.function_managers.exponential_function_manager (module), 44	
mythril.laser.ethereum.function_managers.mythic (module), 44	
mythril.laser.ethereum.instruction_data (module), 64	
mythril.laser.ethereum.instructions (module), 64	
mythril.laser.ethereum.natives (module), 79	
mythril.laser.ethereum.state (module), 53	
mythril.laser.ethereum.state.account (module), 45	
mythril.laser.ethereum.state.annotation (module), 46	
mythril.laser.ethereum.state.calldata (module), 46	
mythril.laser.ethereum.state.constraints (module), 48	
mythril.laser.ethereum.state.environment (module), 48	
mythril.laser.ethereum.state.global_state (module), 49	
mythril.laser.ethereum.state.machine_state (module), 50	
mythril.laser.ethereum.state.memory (module), 51	
mythril.laser.ethereum.state.return_data (module), 52	
mythril.laser.ethereum.state.world_state (module), 52	
mythril.laser.ethereum.strategy (module), 56	
mythril.laser.ethereum.strategy.basic (module), 54	
mythril.laser.ethereum.strategy.beam (module), 55	
mythril.laser.ethereum.strategy.concolic (module), 55	
mythril.laser.ethereum.strategy.extension (module), 54	
mythril.laser.ethereum.strategy.extension_mythril (module), 54	
mythril.laser.ethereum.svm (module), 80	
mythril.laser.ethereum.time_handler	
	mythril.laser.ethereum.transaction (module), 61
	mythril.laser.ethereum.transaction.concolic (module), 56
	mythril.laser.ethereum.transaction.symbolic (module), 58
	mythril.laser.ethereum.transaction.transaction_model (module), 59
	mythril.laser.execution_info (module), 99
	mythril.laser.plugin.builder (module), 88
	mythril.laser.plugin.interface (module), 88
	mythril.laser.plugin.loader (module), 89
	mythril.laser.plugin.plugins (module), 88
	mythril.laser.plugin.plugins.benchmark (module), 85
	mythril.laser.plugin.plugins.call_depth_limiter (module), 85
	mythril.laser.plugin.plugins.coverage (module), 85
	mythril.laser.plugin.plugins.coverage.coverage_plugin (module), 84
	mythril.laser.plugin.plugins.coverage.coverage_strategy (module), 84
	mythril.laser.plugin.plugins.dependency_pruner (module), 85
	mythril.laser.plugin.plugins.instruction_profiler (module), 86
	mythril.laser.plugin.plugins.mutation_pruner (module), 87
	mythril.laser.plugin.plugins.plugin_annotations (module), 87
	mythril.laser.plugin.plugins.summary_backup (module), 85
	mythril.laser.plugin.signals (module), 89
	mythril.laser.smt (module), 98
	mythril.laser.smt.array (module), 92
	mythril.laser.smt.bitvec (module), 92
	mythril.laser.smt.bitvec_helper (module), 93
	mythril.laser.smt.bool (module), 96
	mythril.laser.smt.expression (module), 97
	mythril.laser.smt.function (module), 97
	mythril.laser.smt.model (module), 98
	mythril.laser.smt.solver (module), 92
	mythril.laser.smt.solver.independence_solver (module), 90
	mythril.laser.smt.solver.solver (module), 91
	mythril.laser.smt.solver.solver_statistics (module), 92

mythril.mythril (module), 101  
 mythril.mythril.mythril\_analyzer (module), 99  
 mythril.mythril.mythril\_config (module), 100  
 mythril.mythril.mythril\_disassembler (module), 100  
 mythril.plugin (module), 102  
 mythril.plugin.discovery (module), 101  
 mythril.plugin.interface (module), 101  
 mythril.plugin.loader (module), 102  
 mythril.solidity (module), 103  
 mythril.solidity.soliditycontract (module), 103  
 mythril.support (module), 106  
 mythril.support.loader (module), 104  
 mythril.support.lock (module), 104  
 mythril.support.model (module), 104  
 mythril.support.opcodes (module), 104  
 mythril.support.signatures (module), 104  
 mythril.support.source\_support (module), 105  
 mythril.support.start\_time (module), 106  
 mythril.support.support\_args (module), 106  
 mythril.support.support\_utils (module), 106  
 MythrilAnalyzer (class in mythril.mythril.mythril\_analyzer), 99  
 MythrilBaseException, 107  
 MythrilCLIPlugin (class in mythril.plugin.interface), 101  
 MythrilConfig (class in mythril.mythril.mythril\_config), 100  
 MythrilDisassembler (class in mythril.mythril.mythril\_disassembler), 100  
 MythrilLaserPlugin (class in mythril.plugin.interface), 101  
 MythrilPlugin (class in mythril.plugin.interface), 102  
 MythrilPluginLoader (class in mythril.plugin.loader), 102

## N

name (mythril.analysis.module.base.DetectionModule attribute), 28  
 name (mythril.analysis.module.modules.arbitrary\_jump.ArbitraryJump attribute), 21  
 name (mythril.analysis.module.modules.arbitrary\_write.ArbitraryWrite attribute), 22  
 name (mythril.analysis.module.modules.delegatecall.ArbitraryDelegateCall attribute), 22  
 name (mythril.analysis.module.modules.dependence\_on\_origin.TxOrigin attribute), 23  
 name (mythril.analysis.module.modules.dependence\_on\_predictable\_vars.attribute), 23  
 name (mythril.analysis.module.modules.ether\_thief.EtherThief attribute), 23  
 name (mythril.analysis.module.modules.exceptions.Exceptions attribute), 24  
 name (mythril.analysis.module.modules.external\_calls.ExternalCalls attribute), 24  
 name (mythril.analysis.module.modules.integer.IntegerArithmetics attribute), 25  
 name (mythril.analysis.module.modules.multiple\_sends.MultipleSends attribute), 25  
 name (mythril.analysis.module.modules.state\_change\_external\_calls.StateChange attribute), 25  
 name (mythril.analysis.module.modules.suicide.AccidentallyKillable attribute), 26  
 name (mythril.analysis.module.modules.unchecked\_retval.UncheckedRetVal attribute), 26  
 name (mythril.analysis.module.modules.user\_assertions.UserAssertions attribute), 27  
 name (mythril.laser.plugin.builder.PluginBuilder attribute), 88  
 name (mythril.laser.plugin.plugins.benchmark.BenchmarkPluginBuilder attribute), 85  
 name (mythril.laser.plugin.plugins.call\_depth\_limiter.CallDepthLimitBuilder attribute), 85  
 name (mythril.laser.plugin.plugins.coverage.coverage\_plugin.CoveragePlugin attribute), 84  
 name (mythril.laser.plugin.plugins.dependency\_pruner.DependencyPruner attribute), 86  
 name (mythril.laser.plugin.plugins.instruction\_profiler.InstructionProfiler attribute), 87  
 name (mythril.laser.plugin.plugins.mutation\_pruner.MutationPrunerBuilder attribute), 87  
 name (mythril.plugin.interface.MythrilPlugin attribute), 102  
 native\_call() (in module mythril.laser.ethereum.call), 62  
 native\_contracts() (in module mythril.laser.ethereum.natives), 80  
 NativeContractException, 79  
 new\_bitvec() (mythril.laser.ethereum.state.global\_state.GlobalState method), 49  
 NoContractFoundError, 107  
 NoCopyAnnotation (class in mythril.laser.ethereum.state.annotation), 46  
 NoDelegateCall (class in mythril.laser.ethereum.cfg), 63  
 NodeFlags (class in mythril.laser.ethereum.cfg), 63  
 NoDelete (in module mythril.laser.smt.bool), 96  
 not\_() (mythril.laser.ethereum.instructions.Instruction method), 73  
 number\_() (mythril.laser.ethereum.instructions.Instruction method), 73

## O

OldBlockNumberUsedAnnotation (class in [mythril.analysis.module.modules.dependence\\_on\\_predictable\\_vars](#)), 23  
 Op (class in [mythril.analysis.ops](#)), 30  
 Optimize (class in [mythril.laser.smt.solver.solver](#)), 91  
 Or () (in module [mythril.laser.smt.bool](#)), 96  
 or\_ () ([mythril.laser.ethereum.instructions.Instruction](#) method), 74  
 origin\_ () ([mythril.laser.ethereum.instructions.Instruction](#) method), 74  
 OutOfGasException, 63  
 OverUnderflowAnnotation (class in [mythril.analysis.module.modules.integer](#)), 25  
 OverUnderflowStateAnnotation (class in [mythril.analysis.module.modules.integer](#)), 25

**P**  
 parse\_args\_and\_execute () (in module [mythril.interfaces.cli](#)), 42  
 pc\_ () ([mythril.laser.ethereum.instructions.Instruction](#) method), 74  
 persist\_over\_calls ([mythril.analysis.issue\\_annotation.IssueAnnotation](#) attribute), 30  
 persist\_over\_calls ([mythril.laser.ethereum.state.annotation.StateAnnotation](#) attribute), 46  
 persist\_over\_calls ([mythril.laser.ethereum.strategy.concolic.TraceAnnotation](#) attribute), 56  
 persist\_over\_calls ([mythril.laser.plugin.plugins.plugin\\_annotations.MutationAnnotation](#) attribute), 87  
 persist\_to\_world\_state ([mythril.laser.ethereum.state.annotation.StateAnnotation](#) attribute), 46  
 persist\_to\_world\_state () ([mythril.analysis.issue\\_annotation.IssueAnnotation](#) method), 30  
 plugin\_description ([mythril.plugin.interface.MythrilPlugin](#) attribute), 102  
 plugin\_license ([mythril.plugin.interface.MythrilPlugin](#) attribute), 102  
 plugin\_type ([mythril.plugin.interface.MythrilPlugin](#) attribute), 102  
 plugin\_version ([mythril.plugin.interface.MythrilPlugin](#) attribute), 102  
 PluginBuilder (class in [mythril.laser.plugin.builder](#)), 88  
 PluginDiscovery (class in [mythril.plugin.discovery](#)), 101  
 PluginSignal, 89  
 PluginSkipState, 89  
 PluginSkipWorldState, 89  
 pop () ([mythril.laser.ethereum.state.machine\\_state.MachineStack](#) method), 50  
 pop () ([mythril.laser.ethereum.state.machine\\_state.MachineState](#) method), 51  
 pop () ([mythril.laser.smt.solver.independence\\_solver.IndependenceSolver](#) method), 90  
 pop () ([mythril.laser.smt.solver.solver.Solver](#) method), 91  
 pop\_ () ([mythril.laser.ethereum.instructions.Instruction](#) method), 74  
 pop\_bitvec () (in module [mythril.laser.ethereum.util](#)), 83  
 POST ([mythril.analysis.module.base.EntryPoint](#) attribute), 28  
 post\_handler () ([mythril.laser.ethereum.instructions.Instruction](#) method), 74  
 post\_hook () ([mythril.laser.ethereum.svm.LaserEVM](#) method), 81  
 post\_hooks ([mythril.analysis.module.base.DetectionModule](#) attribute), 28  
 post\_hooks ([mythril.analysis.module.modules.dependence\\_on\\_origin.TxContext](#) attribute), 23  
 post\_hooks ([mythril.analysis.module.modules.dependence\\_on\\_predictable\\_vars.TxContext](#) attribute), 23  
 post\_hooks ([mythril.analysis.module.modules.ether\\_thief.EtherThief](#) attribute), 23  
 post\_hooks ([mythril.analysis.module.modules.unchecked\\_retval.UncheckedRetval](#) attribute), 26  
 PotentialIssue (class in [mythril.analysis.potential\\_issues](#)), 31  
 PotentialIssuesAnnotation (class in [mythril.analysis.potential\\_issues](#)), 31  
 pre\_hook () ([mythril.laser.ethereum.svm.LaserEVM](#) method), 81  
 pre\_hooks ([mythril.analysis.module.base.DetectionModule](#) attribute), 28  
 pre\_hooks ([mythril.analysis.module.modules.arbitrary\\_jump.ArbitraryJump](#) attribute), 21  
 pre\_hooks ([mythril.analysis.module.modules.arbitrary\\_write.ArbitraryWrite](#) attribute), 22  
 pre\_hooks ([mythril.analysis.module.modules.delegatecall.ArbitraryDelegatecall](#) attribute), 22  
 pre\_hooks ([mythril.analysis.module.modules.dependence\\_on\\_origin.TxContext](#) attribute), 23  
 pre\_hooks ([mythril.analysis.module.modules.dependence\\_on\\_predictable\\_vars.TxContext](#) attribute), 23  
 pre\_hooks ([mythril.analysis.module.modules.exceptions.Exceptions](#) attribute), 24  
 pre\_hooks ([mythril.analysis.module.modules.external\\_calls.ExternalCalls](#) attribute), 24  
 pre\_hooks ([mythril.analysis.module.modules.integer.IntegerArithmetics](#)

attribute), 25  
 pre\_hooks (mythril.analysis.module.modules.multiple\_sends.MultipleSends attribute), 25  
 pre\_hooks (mythril.analysis.module.modules.state\_change\_external\_modules.StateChangeAfterCall attribute), 25  
 pre\_hooks (mythril.analysis.module.modules.suicide.AccidentallyKilled attribute), 26  
 pre\_hooks (mythril.analysis.module.modules.unchecked\_retval.UncheckedRetval attribute), 26  
 pre\_hooks (mythril.analysis.module.modules.user\_assertions.UserAssertions attribute), 27  
 PredictableValueAnnotation (class in mythril.analysis.module.modules.dependence\_on\_predictable\_value), 23  
 PredictableVariables (class in mythril.analysis.module.modules.dependence\_on\_predictable\_value), 23  
 pretty\_print\_model () (in module mythril.analysis.solver), 33  
 print\_function\_report () (in module mythril.interfaces.cli), 42  
 println () (mythril.interfaces.epic.LolCat method), 43  
 println\_ani () (mythril.interfaces.epic.LolCat method), 43  
 println\_plain () (mythril.interfaces.epic.LolCat method), 43  
 push\_ () (mythril.laser.ethereum.instructions.Instruction method), 74  
 put\_account () (mythril.laser.ethereum.state.world\_state.WorldState attribute), 53  
**R**  
 rainbow () (mythril.interfaces.epic.LolCat method), 43  
 read\_balance (mythril.support.loader.DynLoader attribute), 104  
 read\_storage (mythril.support.loader.DynLoader attribute), 104  
 register\_hooks () (mythril.laser.ethereum.svm.LaserEVM method), 81  
 register\_instr\_hooks () (mythril.laser.ethereum.svm.LaserEVM method), 82  
 register\_laser\_hooks () (mythril.laser.ethereum.svm.LaserEVM method), 82  
 register\_module () (mythril.analysis.module.loader.ModuleLoader method), 28  
 release () (mythril.support.lock.LockFile method), 104  
 Report (class in mythril.analysis.report), 32  
 reset () (in module mythril.interfaces.epic), 43  
 reset () (mythril.laser.ethereum.function\_managers.keccak\_function\_manager.KeccakFunctionManager method), 45  
 reset () (mythril.laser.smt.solver.independence\_solver.IndependenceSolver method), 90  
 reset () (mythril.laser.smt.solver.solver.Solver method), 90  
 reset\_callback\_modules () (in module mythril.analysis.module.util), 29  
 reset\_module () (mythril.analysis.module.base.DetectionModule method), 28  
 reset\_module () (mythril.analysis.module.modules.arbitrary\_jump.ArbitraryJump method), 21  
 reset\_module () (mythril.analysis.module.modules.arbitrary\_write.ArbitraryWrite method), 21  
 reset\_module () (mythril.analysis.module.modules.ether\_thief.EtherThief method), 23  
 reset\_module () (mythril.analysis.module.modules.integer.IntegerArithmetic method), 25  
 reset\_module () (mythril.analysis.module.modules.suicide.AccidentallyKilled method), 26  
 resolve\_function\_names () (mythril.analysis.report.Issue method), 32  
 resolve\_input () (mythril.analysis.report.Issue static method), 32  
 restart\_counter () (mythril.laser.ethereum.transaction.transaction\_models.TxIdManager method), 61  
 retrieve\_callback\_issues () (in module mythril.analysis.security), 33  
 RETURN (mythril.laser.ethereum.cfg.JumpType attribute), 63  
 return\_ () (mythril.laser.ethereum.instructions.Instruction method), 74  
 ReturnData (class in mythril.laser.ethereum.state.return\_data), 52  
 returndatacopy\_ () (mythril.laser.ethereum.instructions.Instruction method), 75  
 returndatasize\_ () (mythril.laser.ethereum.instructions.Instruction method), 75  
 ReturnRandomNaivelyStrategy (class in mythril.laser.ethereum.strategy.basic), 55  
 returnsub\_ () (mythril.laser.ethereum.instructions.Instruction method), 75  
 ReturnWeightedRandomStrategy (class in mythril.laser.ethereum.strategy.basic), 55  
 RetVal (class in mythril.analysis.module.modules.unchecked\_retval), 26  
 revert\_ () (mythril.laser.ethereum.instructions.Instruction method), 75  
 ripemd160 () (in module mythril.laser.ethereum.natives), 80  
 rlp () (in module mythril.interfaces.epic), 43  
 rzpad () (in module mythril.support.support\_utils), 106



## S

- `safe_decode()` (in module `mythril.ethereum.util`), 41
- `safe_decode()` (in module `mythril.laser.ethereum.util`), 83
- `safe_ord()` (in module `mythril.laser.ethereum.natives`), 80
- `sar_()` (`mythril.laser.ethereum.instructions.Instruction` method), 75
- `sdiv_()` (`mythril.laser.ethereum.instructions.Instruction` method), 75
- `search_importance` (`mythril.analysis.potential_issues.PotentialIssuesAnnotation` attribute), 31
- `search_importance` (`mythril.laser.ethereum.state.annotation.StateAnnotation` attribute), 46
- `selfbalance_()` (`mythril.laser.ethereum.instructions.Instruction` method), 75
- `selfdestruct_()` (`mythril.laser.ethereum.instructions.Instruction` method), 76
- `serialised_code()` (`mythril.laser.ethereum.state.account.Account` method), 45
- `set_api_from_config_path()` (`mythril.mythril.mythril_config.MythrilConfig` method), 100
- `set_api_infura_id()` (`mythril.mythril.mythril_config.MythrilConfig` method), 100
- `set_api_rpc()` (`mythril.mythril.mythril_config.MythrilConfig` method), 100
- `set_api_rpc_infura()` (`mythril.mythril.mythril_config.MythrilConfig` method), 100
- `set_api_rpc_localhost()` (`mythril.mythril.mythril_config.MythrilConfig` method), 100
- `set_args()` (`mythril.plugin.loader.MythrilPluginLoader` method), 102
- `set_balance()` (`mythril.laser.ethereum.state.account.Account` method), 45
- `set_config()` (in module `mythril.interfaces.cli`), 42
- `set_criterion_satisfied()` (`mythril.laser.ethereum.strategy.CriterionSearchStrategy` method), 56
- `set_storage()` (`mythril.laser.ethereum.state.account.Account` method), 45
- `set_timeout()` (`mythril.laser.smt.solver.independence_solver.IndependenceSolver` method), 90
- `set_timeout()` (`mythril.laser.smt.solver.solver.BaseSolver` method), 91
- `setup_concrete_initial_state()` (in module `mythril.concolic.find_trace`), 35
- `sexpr()` (`mythril.laser.smt.solver.solver.BaseSolver` method), 91
- `sgt_()` (`mythril.laser.ethereum.instructions.Instruction` method), 76
- `sha256()` (in module `mythril.laser.ethereum.natives`), 80
- `sha3()` (in module `mythril.support.support_utils`), 106
- `sha3_()` (`mythril.laser.ethereum.instructions.Instruction` method), 76
- `shl_()` (`mythril.laser.ethereum.instructions.Instruction` method), 76
- `shr_()` (`mythril.laser.ethereum.instructions.Instruction` method), 76
- `SignatureDB` (class in `mythril.support.signatures`), 105
- `signextend_()` (`mythril.laser.ethereum.instructions.Instruction` method), 76
- `simplify()` (in module `mythril.laser.smt.expression`), 97
- `simplify()` (`mythril.laser.smt.expression.Expression` method), 97
- `Singleton` (class in `mythril.support.signatures`), 105
- `Singleton` (class in `mythril.support.support_utils`), 106
- `size` (`mythril.laser.ethereum.state.calldata.BaseCalldata` attribute), 47
- `size` (`mythril.laser.ethereum.state.calldata.BasicConcreteCalldata` attribute), 47
- `size` (`mythril.laser.ethereum.state.calldata.BasicSymbolicCalldata` attribute), 47
- `size` (`mythril.laser.ethereum.state.calldata.ConcreteCalldata` attribute), 47
- `size` (`mythril.laser.ethereum.state.calldata.SymbolicCalldata` attribute), 48
- `size` (`mythril.laser.ethereum.state.return_data.ReturnData` attribute), 52
- `size()` (`mythril.laser.smt.bitvec.BitVec` method), 92
- `size()` (`mythril.laser.smt.expression.Expression` method), 97
- `sload_()` (`mythril.laser.ethereum.instructions.Instruction` method), 77
- `slt_()` (`mythril.laser.ethereum.instructions.Instruction` method), 77
- `smod_()` (`mythril.laser.ethereum.instructions.Instruction` method), 77
- `solc_exists()` (in module `mythril.ethereum.util`), 41
- `SolidityContract` (class in `mythril.solidity.soliditycontract`), 103
- `SolidityFile` (class in `mythril.solidity.soliditycontract`), 103
- `Solver` (class in `mythril.laser.smt.solver.solver`), 91
- `SolverStatistics` (class in `mythril.laser.smt.solver.solver_statistics`), 92
- `SolverTimeOutException`, 107

[sort\\_and\\_eliminate\\_states\(\)](#) ([mythril.laser.ethereum.strategy.beam.BeamSearch](#) [method](#)), 55  
[sorted\\_issues\(\)](#) ([mythril.analysis.report.Report](#) [method](#)), 32  
[Source](#) (class in [mythril.support.source\\_support](#)), 105  
[SourceCodeInfo](#) (class in [mythril.solidity.soliditycontract](#)), 103  
[SourceMapping](#) (class in [mythril.solidity.soliditycontract](#)), 103  
[SQLiteDB](#) (class in [mythril.support.signatures](#)), 104  
[SRem\(\)](#) (in module [mythril.laser.smt.bitvec\\_helper](#)), 94  
[sstore\\_\(\)](#) ([mythril.laser.ethereum.instructions.Instruction](#) [method](#)), 77  
[STACK\\_LIMIT](#) ([mythril.laser.ethereum.state.machine\\_state.MachineState](#) [attribute](#)), 50  
[StackOverflowException](#), 63  
[StackUnderflowException](#), 64  
[start\\_execution\(\)](#) ([mythril.laser.ethereum.time\\_handler.TimeHandler](#) [method](#)), 82  
[StartTime](#) (class in [mythril.support.start\\_time](#)), 106  
[stat\\_smt\\_query\(\)](#) (in module [mythril.laser.smt.solver.solver\\_statistics](#)), 92  
[StateAnnotation](#) (class in [mythril.laser.ethereum.state.annotation](#)), 46  
[StateChangeAfterCall](#) (class in [mythril.analysis.module.modules.state\\_change\\_external\\_calls](#)), 25  
[StateChangeCallsAnnotation](#) (class in [mythril.analysis.module.modules.state\\_change\\_external\\_calls](#)), 26  
[StateTransition](#) (class in [mythril.laser.ethereum.instructions](#)), 78  
[staticcall\\_\(\)](#) ([mythril.laser.ethereum.instructions.Instruction](#) [method](#)), 77  
[staticcall\\_post\(\)](#) ([mythril.laser.ethereum.instructions.Instruction](#) [method](#)), 77  
[stop\\_\(\)](#) ([mythril.laser.ethereum.instructions.Instruction](#) [method](#)), 77  
[Storage](#) (class in [mythril.laser.ethereum.state.account](#)), 45  
[sub\\_\(\)](#) ([mythril.laser.ethereum.instructions.Instruction](#) [method](#)), 78  
[substitute\(\)](#) ([mythril.laser.smt.array.BaseArray](#) [method](#)), 92  
[substitute\(\)](#) ([mythril.laser.smt.bool.Bool](#) [method](#)), 96  
[Sum\(\)](#) (in module [mythril.laser.smt.bitvec\\_helper](#)), 94  
[SVMError](#), 82  
[swap\\_\(\)](#) ([mythril.laser.ethereum.instructions.Instruction](#) [method](#)), 78  
[swc\\_id](#) ([mythril.analysis.module.base.DetectionModule](#) [attribute](#)), 28  
[swc\\_id](#) ([mythril.analysis.module.modules.arbitrary\\_jump.ArbitraryJump](#) [attribute](#)), 22  
[swc\\_id](#) ([mythril.analysis.module.modules.arbitrary\\_write.ArbitraryStorage](#) [attribute](#)), 22  
[swc\\_id](#) ([mythril.analysis.module.modules.delegatecall.ArbitraryDelegatecall](#) [attribute](#)), 22  
[swc\\_id](#) ([mythril.analysis.module.modules.dependence\\_on\\_origin.TxOrigin](#) [attribute](#)), 23  
[swc\\_id](#) ([mythril.analysis.module.modules.dependence\\_on\\_predictable\\_value](#) [attribute](#)), 23  
[swc\\_id](#) ([mythril.analysis.module.modules.ether\\_thief.EtherThief](#) [attribute](#)), 24  
[swc\\_id](#) ([mythril.analysis.module.modules.exceptions.Exceptions](#) [attribute](#)), 24  
[swc\\_id](#) ([mythril.analysis.module.modules.external\\_calls.ExternalCalls](#) [attribute](#)), 24  
[swc\\_id](#) ([mythril.analysis.module.modules.integer.IntegerArithmetics](#) [attribute](#)), 25  
[swc\\_id](#) ([mythril.analysis.module.modules.multiple\\_sends.MultipleSends](#) [attribute](#)), 25  
[swc\\_id](#) ([mythril.analysis.module.modules.state\\_change\\_external\\_calls.StateChangeCalls](#) [attribute](#)), 26  
[swc\\_id](#) ([mythril.analysis.module.modules.suicide.AccidentallyKillable](#) [attribute](#)), 26  
[swc\\_id](#) ([mythril.analysis.module.modules.unchecked\\_retval.UncheckedRetVal](#) [attribute](#)), 26  
[swc\\_id](#) ([mythril.analysis.module.modules.user\\_assertions.UserAssertions](#) [attribute](#)), 27  
[sym\\_exec\(\)](#) ([mythril.laser.ethereum.svm.LaserEVM](#) [method](#)), 82  
[SymbolFactory](#) (class in [mythril.laser.smt](#)), 98  
[SYMBOLIC](#) ([mythril.analysis.ops.VarType](#) [attribute](#)), 31  
[symbolic](#) ([mythril.laser.smt.bitvec.BitVec](#) [attribute](#)), 93  
[SymbolicCalldata](#) (class in [mythril.laser.ethereum.state.calldata](#)), 47  
[SymExecWrapper](#) (class in [mythril.analysis.symbolic](#)), 33  
[synchronized\(\)](#) (in module [mythril.support.signatures](#)), 105

## T

[time\\_remaining\(\)](#) ([mythril.laser.ethereum.time\\_handler.TimeHandler](#) [method](#)), 82  
[TimeHandler](#) (class in [mythril.laser.ethereum.time\\_handler](#)), 82  
[timestamp\\_\(\)](#) ([mythril.laser.ethereum.instructions.Instruction](#) [method](#)), 78  
[to\\_dict\(\)](#) ([mythril.disassembler.asm.EvmInstruction](#) [method](#)), 35  
[to\\_signed\(\)](#) (in module [mythril.laser.ethereum.util](#)), 83

TraceAnnotation (class in [mythril.laser.ethereum.strategy.concolic](#)), 55

Transaction ([mythril.laser.ethereum.cfg.JumpType](#) attribute), 63

transaction\_sequence\_jsonv2 ([mythril.analysis.report.Issue](#) attribute), 32

transaction\_sequence\_users ([mythril.analysis.report.Issue](#) attribute), 32

TransactionData (class in [mythril.concolic.concrete\\_data](#)), 35

TransactionEndSignal, 60

TransactionStartSignal, 61

transfer\_ether() (in module [mythril.laser.ethereum.instructions](#)), 79

TxIdManager (class in [mythril.laser.ethereum.transaction.transaction\\_models](#)), 61

TxOrigin (class in [mythril.analysis.module.modules.dependence\\_on\\_origin](#)), 22

TxOriginAnnotation (class in [mythril.analysis.module.modules.dependence\\_on\\_origin](#)), 23

**U**

UDiv() (in module [mythril.laser.smt.bitvec\\_helper](#)), 94

UGE() (in module [mythril.laser.smt.bitvec\\_helper](#)), 95

UGT() (in module [mythril.laser.smt.bitvec\\_helper](#)), 95

ULE() (in module [mythril.laser.smt.bitvec\\_helper](#)), 95

ULT() (in module [mythril.laser.smt.bitvec\\_helper](#)), 95

UncheckedRetVal (class in [mythril.analysis.module.modules.unchecked\\_retval](#)), 26

UncheckedRetValAnnotation (class in [mythril.analysis.module.modules.unchecked\\_retval](#)), 26

UNCONDITIONAL ([mythril.laser.ethereum.cfg.JumpType](#) attribute), 63

UnsatError, 107

UnsupportedPluginType, 102

update\_cache() ([mythril.analysis.module.base.DetectionModule](#) method), 28

update\_calls() ([mythril.laser.plugin.plugins.dependency\\_pruner.DependencyPruner](#) method), 86

update\_sloads() ([mythril.laser.plugin.plugins.dependency\\_pruner.DependencyPruner](#) method), 86

update\_sstores() ([mythril.laser.plugin.plugins.dependency\\_pruner.DependencyPruner](#) method), 86

URem() (in module [mythril.laser.smt.bitvec\\_helper](#)), 95

UserAssertions (class in [mythril.analysis.module.modules.user\\_assertions](#)), 27

**V**

validate\_args() (in module [mythril.interfaces.cli](#)), 42

validate\_block() (in module [mythril.ethereum.interface.rpc.utils](#)), 39

value ([mythril.laser.smt.bitvec.BitVec](#) attribute), 93

value ([mythril.laser.smt.bool.Bool](#) attribute), 96

Variable (class in [mythril.analysis.ops](#)), 31

VarType (class in [mythril.analysis.ops](#)), 30

VmException, 64

**W**

wanna\_execute() ([mythril.laser.plugin.plugins.dependency\\_pruner.DependencyPruner](#) method), 86

wei\_to\_ether() (in module [mythril.ethereum.interface.rpc.utils](#)), 39

WorldState (class in [mythril.laser.ethereum.state.world\\_state](#)), 52

wrap() ([mythril.interfaces.epic.LolCat](#) method), 43

write\_word\_at() ([mythril.laser.ethereum.state.memory.Memory](#) method), 51

WriteProtection, 64

WbDependencyAnnotation (class in [mythril.laser.plugin.plugins.plugin\\_annotations](#)), 87

**X**

Xor() (in module [mythril.laser.smt.bool](#)), 96

xor\_() ([mythril.laser.ethereum.instructions.Instruction](#) method), 78

**Z**

zpad() (in module [mythril.support.support\\_utils](#)), 106