# SMART CONTRACT AUTOMATED TESTING GUIDELINES

Author: enderlocphan@gmail.com

Git Repo: https://github.com/enderphan94/solidity-pentest/

## Foreword

The documents aim to recap my experience in smart contract automated testing besides the manual testing. I also put the issues that I faced during the execution, indeed, solutions are given.

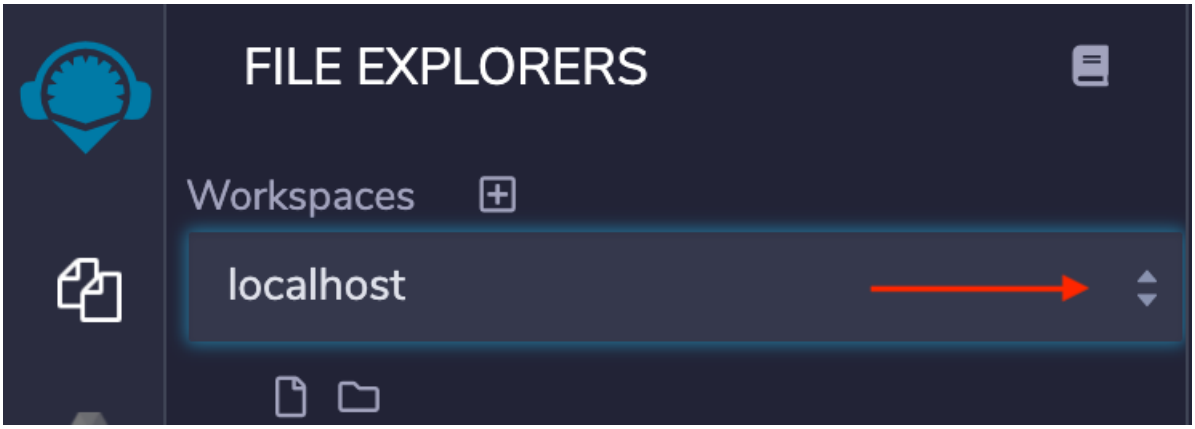
## Connecting with Remix from localhost

For a complex project, you can't just copy paste the single sol file and let it run. To make our life easier, Remix has localhost connection which allows you to interact with your project in your local machine remotely.

This is something I'm used to doing when the project has a large number of inheritant contracts. Obviously, this make our life easier than ever by just downloading the git project and do some commands.

### Steps:

1. Compile your truffle contract if needed with `npm install` (!remember remvove the package-lock.json, if it does have it). Otherwise, the remix wouldn't be able to load all libraries for the contracts that are being called.
2. Go to WorkSpaces on the left panel and choose "Connect to Localhost"




3. The message box pops up and you just need to read carefully and copy the command shown in the box to connect your localhost

```
remixd -s path-to-the-shared-folder --remix-ide remix-ide-instance-URL
```

Important:

> path-to-the-shared-folder: needs to be an absolute path
>
> remix-ide-instance-URL: needs to plain with http or https

eg:

```
remixd -s /home/enderphan/LOLToken/ --remix-ide http(s)://your-remix-address.com/
```

### Issues

#### Issue 1

Somtimes I still got this error from Remix

> Cannot connect to the remixd daemon. Please make sure you have the remixd running in the background.

What I usually do is just switch to a new terminal tab and re-type the remixd command. If needed, you can just uinstall and reinstall the remixd (Close VS-Code to do this, if you have it opened)

https://remix-ide.readthedocs.io/en/latest/remixd.html

#### Issue 2

The same error but another issue.

https://ethereum.stackexchange.com/questions/78637/cant-connect-remix-ide-using-remixd

# Solc version problems

Source: https://github.com/crytic/solc-select

### Issues

You need to just switch the version of solc quickly by a command. The version of solc is kindda painful, depending on the tools and project, you need to use a specific and exact version to compile.. otherwise broke.

During my audit, I've suffered with solc-select installations. I used to install via the shell command, but now they've upraded to pip3. The thing is that some docker containers do not support pip3, so you would need to install solc-selct into that docker but pip3. Therefore, I'v a copied version of the solc-select installed via shell.

### Installation

Via shell: https://github.com/enderphan94/solc-select-sh-version

Via pip3: https://github.com/crytic/solc-select

### Usage:

Install the version you want

```
solc-select install 0.8.0
```

And use it

```
solc-select use 0.8.0
```

Check your solc version again

```
solc --version
```

# Tools

### 1. Slither

Source: https://github.com/crytic/slither

### Features

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

### How to install

Slither requires Python 3.6+ and solc, the Solidity compiler.

#### Using Pip

```
pip3 install slither-analyzer
```

#### Using Git

```
git clone https://github.com/crytic/slither.git && cd slither
python3 setup.py install
```

We recommend using an Python virtual environment, as detailed in the Developer Installation Instructions, if you prefer to install Slither via git.

#### Using Docker

Use the `eth-security-toolbox` docker image. It includes all of our security tools and every major version of Solidity in a single image. `/home/share` will be mounted to `/share` in the container.

```
docker pull trailofbits/eth-security-toolbox
```

To share a directory in the container:

```
docker run -it -v /home/share:/share trailofbits/eth-security-toolbox
```

### Usage

`slither <file-name>.sol`

### Isssue

> Error: Source "@openzeppelin/contracts/utils/Context.sol" not found: File outside of allowed directories.

Fixed: the `--allow-path` does not work, just download the library and copy them into the dir.. casual way :/

## 2. Mythril

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.

Source: https://github.com/ConsenSys/mythril

### How to install

```
$ docker pull mythril/myth
```

Install from Pypi:

```
$ pip3 install mythril
```

Note: In my exprience, I prefer using mythril version installed via pip3 rather than Docker. I've faced so many issues with the docker version, and I decided to switch to pip3 one.

### Usage

Via pip3: https://github.com/ConsenSys/mythril/blob/develop/README.md#usage

Via Docker: `docker run -v $(pwd):/tmp mythril/myth a /tmp/<file-name>.sol --solv 0.5.0`

### Issues

#### Issue 1

In case the tool gives you this error:

> mythril.mythril.mythril_disassembler [ERROR]: The file Token.sol does not contain a compilable contract. mythril.interfaces.cli [ERROR]: input files do not contain any valid contracts

We can use contract address in testnet or ganache https://mythril-classic.readthedocs.io/en/master/security-analysis.html

Ganache: `myth a --rpc ganache -a <address>`

#### Issue 2

Evn: MacOS

Just in case the command `Pip3 install mythril` does not work. I don't remember what happened exactly but something does not work with pip3 in MacOS :)

Use the following command

```
sudo xcode-select --switch /Library/Developer/CommandLineTools
```

#### Issue 3

Error

> in self.solidity_files[file_index].full_contract_src_maps IndexError: list index out of range

Just uninstall mythril and reinstall it

```
pip3 uninstall mythril
```

```
pip3 install mythril
````

### 3. Manticore

This tool takes quite a long time to complete.

#### Features

Program Exploration: Manticore can execute a program with symbolic inputs and explore all the possible states it can reach

Input Generation: Manticore can automatically produce concrete inputs that result in a given program state

Error Discovery: Manticore can detect crashes and other failure cases in binaries and smart contracts

Instrumentation: Manticore provides fine-grained control of state exploration via event callbacks and instruction hooks

Programmatic Interface: Manticore exposes programmatic access to its analysis engine via a Python API

#### Installation

> Note: We recommend installing Manticore in a [virtual environment](https://packaging.python.org/guides/installing-using-pip-and-vi
 to prevent conflicts with other projects or packages

Option 1: Installing from PyPI:

```bash
pip install manticore
```

Option 2: Installing from PyPI, with extra dependencies needed to execute native binaries:

```
pip install "manticore[native]"
```

Option 3: Installing a nightly development build:

```
pip install --pre "manticore[native]"
```

Option 4: Installing from the `master` branch:

```
git clone https://github.com/trailofbits/manticore.git
cd manticore
pip install -e ".[native]"
```

Option 5: Install via Docker:

```
docker pull trailofbits/manticore
```

Once installed, the `manticore` CLI tool and Python API will be available.

For a development installation, see our wiki.

## Usage

Sigle contract in a file

```
manticore <file-name>.sol
```

Mutiple contracts in a file

```
manticore <file-name>.sol --contract <main-contract-name>
```

Note:

Manticore takes quite a long time to complete the scan by default, so usually I also use `--quick-mode` option for quick exploration. Disable gas, generate testcase only for alive states, do not explore constant functions. Disable all detectors.

```
manticore <file-name>.sol --contract <main-contract-name> --quick-mode
```

## 4. Theo

Source: https://github.com/cleanunicorn/theo

### Features

- Automatic smart contract scanning which generates a list of possible exploits.
- Sending transactions to exploit a smart contract.
- Transaction pool monitor.
- Web3 console
- Frontrunning and backrunning transactions.
- Waiting for a list of transactions and sending out others.
- Estimating gas for transactions means only successful transactions are sent.
- Disabling gas estimation will send transactions with a fixed gas quantity.

### Installation

```
pip install theo
```

### Usage

Usually I deploy the smart contract in Ganache local network, from that, I can freely have the private keys of many accounts. If you have metamask installed, you can deploy in the testnet and get the private key of the accounts.

1. Deploy the contract
2. Run

```
theo --rpc-http <your-network>
```

3. Enter the private key of the attack's account
4. Enter the smart contract address

eg:

```
theo --rpc-http http://127.0.0.1:8545
```

## 5. SmartCheck

Souce: https://www.npmjs.com/package/@smartdec/smartcheck

SmartCheck is an extensible static analysis tool for discovering vulnerabilities and other code issues in Ethereum smart contracts written in the Solidity programming language

### Installation

```
npm install @smartdec/smartcheck -g
```

### Usage

1. Copy the contract to a folder
2. Run

```
smartcheck -p <path to directory or file>
```

## 6. Securitfy2

Source: https://github.com/eth-sri/securify2

### Furture

- Supports 38 vulnerabilities (see table below)
- Implements novel context-sensitive static analysis written in Datalog
- Analyzes contracts written in Solidity >= 0.5.8

### Installation

To build the container:

```
sudo docker build -t securify .
```

To run the container:

```
sudo docker run -it -v <contract-dir-full-path>:/share securify /share/<contract>.sol
```

contract-dir-full-path: should be the absolute path

eg:

```
sudo docker run -it -v </Users/foob/contract/>:/share securify /share/test.sol
```

## 7. Sohint

Source: https://github.com/duaraghav8/Ethlint

Ethlint (Formerly Solium) analyzes your Solidity code for style & security issues and fixes them.

### Installation

```
npm install -g ethlint
```

### Usage

In the root directory of your DApp:

```
solium --init
```

This creates .soliumrc.json file, which contains configuration that tells Solium how to lint your project. You should modify this file to configure rules, plugins and sharable configs.

I just usually use this simple setting.

```
{
  "extends": "solium:recommended"
}
```

Then you can run

```
solium -f foobar.sol
```

or

```
solium -d contracts/
```

## 8. Spell check

Source: https://github.com/streetsidesoftware/cspell

The cspell mono-repo, a spell checker for code.

### Installation

```
npm install -g git+https://github.com/streetsidesoftware/cspell-cli
```

### Usage

```
cspell-cli <contract-name>.sol
```

### 9. Sūrya (flow graph)

Source: https://github.com/ConsenSys/surya

Surya is an utility tool for smart contract systems. It provides a number of visual outputs and information about the contracts' structure. Also supports querying the function call graph in multiple ways to aid in the manual inspection of contracts.

#### Installation

Install graphviz

```
brew install graphviz
```

Install surya

```
npm install -g surya
```

#### Usage

```
surya graph <contract>.sol | dot -Tpng > MyContract.png
```

Note: I recommend using Surya in VS Code

## Audit with Visual Studio Code

Here is my list:

1. Name: vscode-slither
   > VS Marketplace Link: https://marketplace.visualstudio.com/items?itemName=samparsky.vscode-slither
2. Name: Solidity Visual Developer
   > VS Marketplace Link: https://marketplace.visualstudio.com/items?itemName=tintinweb.solidity-visual-auditor
3. Name: Slither
   > VS Marketplace Link: https://marketplace.visualstudio.com/items?itemName=trailofbits.slither-vscode
4. Name: Code Spell Checker
   > VS Marketplace Link: https://marketplace.visualstudio.com/items?itemName=streetsidesoftware.code-spell-checker
5. Name: mythril
   > VS Marketplace Link: https://marketplace.visualstudio.com/items?itemName=xgwang.mythril
6. Name: solidity
   > VS Marketplace Link: https://marketplace.visualstudio.com/items?itemName=JuanBlanco.solidity