National College of Ireland

Configuration Manual

MSc Research Project
MSc in Cyber Security

Rahul --
Student ID: 20243804

School of Computing
National College of Ireland

Supervisor: Prof. Michael Prior

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Rahul --
Student ID:	20243804
Programme:	MSc in Cyber Security **Year:** 2021-22
Module:	MSc Research Project/Internship
Lecturer:	Prof. Michael Prior
Submission Due Date:	15/08/2022
Project Title:	How to Improve Security of Smart Contracts written in Solidity in Blockchain by Detecting Reentrancy Vulnerability
Word Count:	1182 **Page Count:** 5

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Rahul --

Date: 15/08/2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Rahul --
Student ID: 20243804

1 Understanding Smart Contracts

First, the package Brownie (christianb93, 2021) was installed on three different platforms Ubuntu, Kali Linux, and Windows OS using the administrative privilege of the terminal and command prompt. Brownie is a Python3 package named eth-brownie.

To get the package running successfully, a module named Ganache (version 6.1) was also installed.

The path where Brownie was installed was added to the path/environment variable.

Occasionally Brownie maintains items that shouldn't be included in the owner's GitHub repository. As a result, usually a subdirectory is created in the repository and added to the gitignore file. A project is created within this subdirectory, and then symlinks are created to the data sets and tests that have to be used.

After that, a smart contract named Counter was compiled. In the case of a project with smart contracts, Brownie compiles all contracts in that project when it runs.

Once the compilation is complete, the Brownie console can be accessed. Brownie is most commonly accessed through this tool. Brownie consoles are basically interactive Python consoles with the additional Brownie features built in.

Now the smart contract, Counter, is deployed using the deploy functionality.

Upon running the command all the methods that the newly created object has will be listed. The methods named increment and read will be used. The logs, events, and information are listed.

By using either the user accounts array or the corresponding API call, Brownie allows us to inspect a list of user accounts handled by Ganache. Alternatively, apps-handeled accounts method could be used to reach the same result.

Then the account is saved in a password-protected file called “myAccount”. Now, it can be reloaded whenever the Brownie console is restarted through the same file.

Now, for running the tests of Brownie we will import some functions and the next step is to obtain a copy of the smart contract that has been deployed. The Fixture module that was imported earlier handles this.

Finally, the scripted tests are run using the test functionality of Brownie.

Second, Web3 (christianb93, 2021) is installed on all three platforms through the terminal/command prompt. It is important that GCC and Python Development package are present on the system for Web3 to run successfully.

Now, the Web3 library needs to be imported. Then a link to the Ganache server is established. To verify whether the link has been successfully made or not, the version string is requested from the server.

Now, we choose one account randomly from the ten accounts present in the Ganache server through the w3 function of Web3 module.

After that, Ether is sent to the address that has been obtained from the account created in the Brownie console using myAccount. Then, Ether is called back from the user account Alice.

The smart contract is retrieved from Web3 by stating the smart contract ABI.

Then the smart contract address and its ABI are made known to the library so as to communicate with the smart contract.

Finally, a contract function method was used such that the counter value is read, incremented by one, and then read again to call the smart contract.

A contract function was run which was not part of the ABI so it would throw an error.

2 Debugged Tools that didn’t Function

I tried running tools such as Rechecker and Oyente but even after debugging the source codes that were used differently in the newer version of the language, I wasn’t able to run them successfully.

Rechecker took more than 10 minutes just to train the model as it is based on deep learning which is not efficient.

While Oyente couldn’t be imported from pip3 or the docker command properly, and hence it crashed.

3 Debugged Tools that Function

Slither (trailofbits, 2022):
First, I installed the dependencies of slither analyzer which are Python 3.8+, and solidity compiler (solc-select), the link to installing the solidity compiler given is wrong it is the solcjs installed through npm which will not work for the slither analyzer, for slither we need solc-select and I found that I download the wrong solidity compiler after it threw error a few times.

Also, we have to give administrative privileges to the terminal/ command prompt on which the slither is installed for it to access the directory to be stored.

After installing the correct dependencies, I download a buggy smart contract through the following link,

Then I installed the slither analyzer through pip3 command of Python3 package, and it ran successfully.

Finally, I tested the slither tool on the buggy smart contract that I downloaded, and the tool successfully did the analysis and generated a report of the reentrancy vulnerability.

Mythril (ConsenSys, 2020):
I downloaded the mithril tool using pip3 command which was successful.

Then I ran the test on a buggy contract that I downloaded from the following link,

It ran successfully but It only runs if the smart contract is written in the same version as of the solidity compiler or else it throws an error.

4 References

christianb93. (2021, 08 18). *Fun with Solidity and Brownie*. Retrieved from LeftAsExercise: https://leftasexercise.com/2021/08/18/fun-with-brownie/

christianb93. (2021, 08 22). *Using web3.py to interact with an Ethereum smart contract*. Retrieved from LeftAsExercise: https://leftasexercise.com/2021/08/22/using-web3-py-to-interact-with-a-smart-contract/

ConsenSys. (2020, 03 23). *Mythril*. Retrieved from GitHub: https://github.com/ConsenSys/mythril

Mueller, B. (2021, 08 06). *Mythril Documentation.* Retrieved 04 03, 2022, from https://readthedocs.org/projects/mythril-classic/downloads/pdf/master/

trailofbits. (2022, 04 21). *Slither*. Retrieved from GitHub: https://github.com/crytic/slither#how-to-install