Iteration Report 1

Improvements to Existing tool: Echidna for Analyzing Smart Contracts

The University of Texas at Arlington

Advanced Topics in Software Engineering

Fall 2022, CSE 6324 - 001

GitHub link: https://github.com/AtulUpadhye17/CSE-6324-Team4

Team 8:

Atul Upadhye (1002030159)

Kundana Vaka(1001827398)

Yamini Dhulipalla(1001913007)

Srikar Sai Yarlagadda(1001860709)

Vigneshwar Selvaraj(1001863627)

Project Plan:

Features:

The following are the features that we will be adding to our tool:

1. **More Inputs Algorithm:** The algorithm that compares with previously tested contracts finds similar ones and tests with the input which worked to find vulnerabilities on those similar ones for faster testing. The algorithm will be a part of the tool we are trying to improve as a feature improvement.
2. **Code Coverage:** Instead of directly starting with the inputs, we can first go through the code line by line looking for functions that have known vulnerabilities and testing them on those functions. As we go line by line code coverage is more.
3. **Improve Automation:** Using the same algorithm which we are planning to develop to find similar previous ran contracts and their respective configuration file and suggest these configurations for current users for better settings and faster and simpler execution of the tool.

Iteration Plan:

Iteration1(Current)	Iteration2	Iteration3
• We have decided upon the features which we are going to implement in future iterations. • Getting a hands-on experience with the echidna tool by doing analysis on the sample smart contracts for a better understanding of the tool. • Explore existing algorithms currently being used for automation.	• Develop the pseudo-code for the algorithm and decide the language for the implementation. • We are going to develop pseudo code supporting the improvements which we are going to make in our tool. • Testing the tool to identify if any new vulnerabilities are detected after making changes to the existing tool.	• Convert the pseudo code to actual code and test the tool on smart contracts to verify the improvements are working.

Risks we faced for Iteration 1:

1. We had issues installing the Echidna tool on our local machine.
2. Due to lack of background knowledge about the Linux system we faced problems for setting the path variables for the tool.

Risks we may face for the upcoming Iterations:

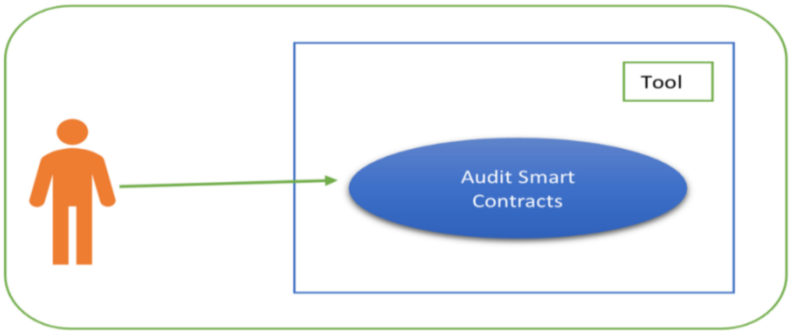
1. When making or implementing new features, we may introduce new vulnerabilities that we need to monitor.
2. After adding the feature, the tool may take longer time than expected to test the contract. (Time Complexity).
3. Echidna implemented in Solidity and Haskell, we might face issues to figure out to call Python Code (Algorithm).

Specification and design

- **Inputs and Outputs:**
    - **Input:** Smart Contracts.
    - **Output:** Testcases and Results of the Analysis.

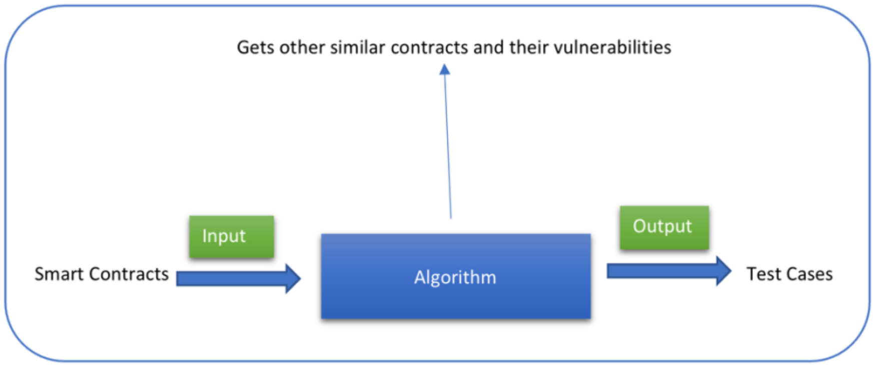
- **Use-Case (User):**

  The user will be able to audit smart contracts with the improvements made in the Echidna tool



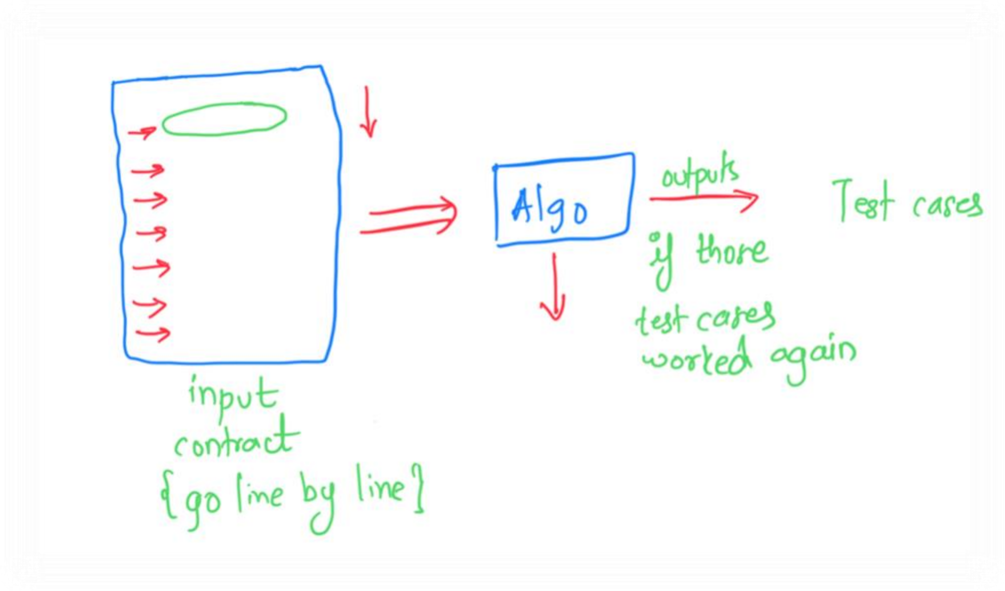
- **Mock Designs:**
    - **The main idea behind the feature:**

  The input will be a smart contract, the algorithm will then analyze the smart contract and finds similar previous one's analyzed before and tries the test cases for which those showed vulnerabilities and outputs those test cases if they show vulnerabilities on the input contract.



- **Line by Line (Code Coverage):**

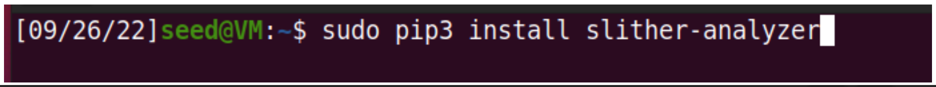
  We will go through the code line by line instead of directly starting with the inputs



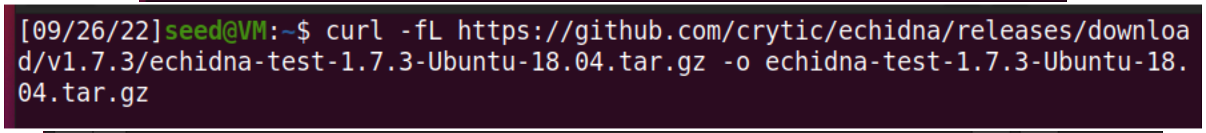
Tool:

For getting more exposure to the Echidna Tool, we tried to install the tool on our laptops and ran the analysis tool on the sample contract using the tutorial links on the GitHub page.

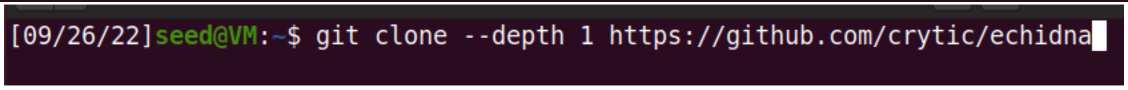
- Instructions for installing the tool:
  - Commands for installing the Echidna tool on the Ubuntu Virtual Machine:



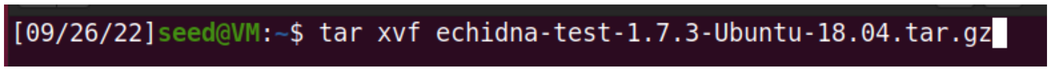
```
[09/26/22]seed@VM:~$ sudo pip3 install slither-analyzer
```



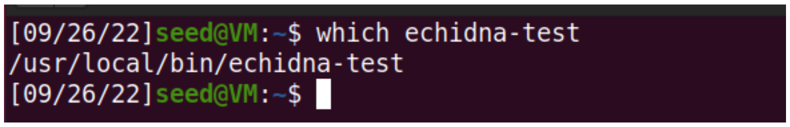
```
[09/26/22]seed@VM:~$ curl -fL https://github.com/crytic/echidna/releases/downloa
d/v1.7.3/echidna-test-1.7.3-Ubuntu-18.04.tar.gz -o echidna-test-1.7.3-Ubuntu-18.
04.tar.gz
```



```
[09/26/22]seed@VM:~$ git clone --depth 1 https://github.com/crytic/echidna
```

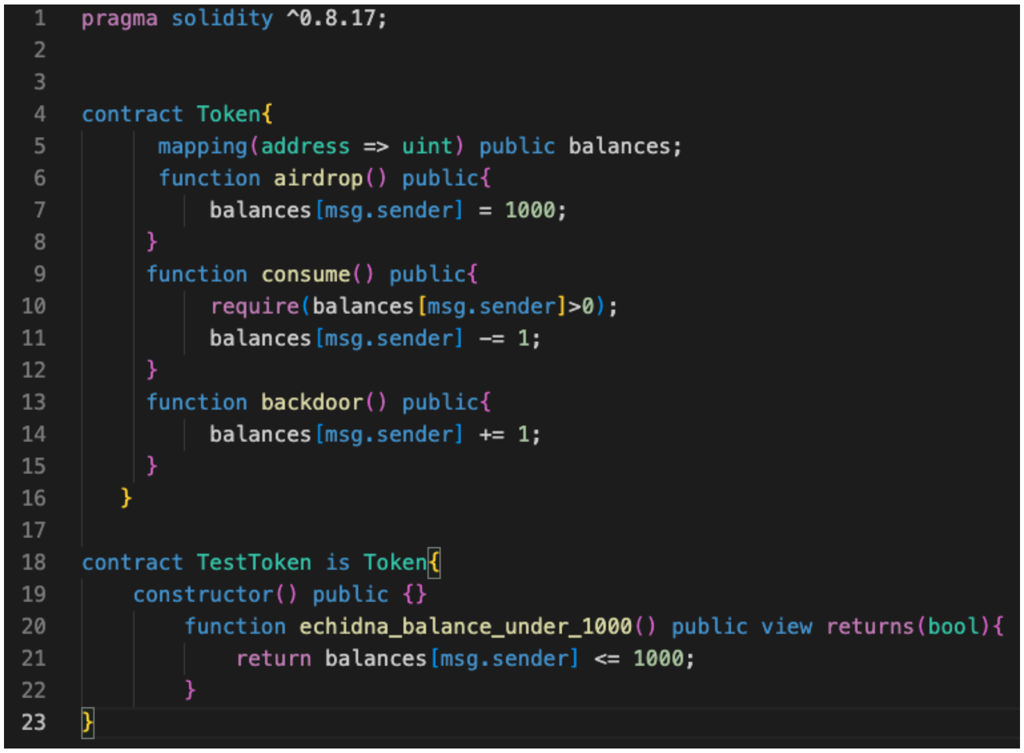


```
[09/26/22]seed@VM:~$ tar xvf echidna-test-1.7.3-Ubuntu-18.04.tar.gz
```



```
[09/26/22]seed@VM:~$ which echidna-test
/usr/local/bin/echidna-test
[09/26/22]seed@VM:~$
```

- Running the tool:
  - Sample Contract:

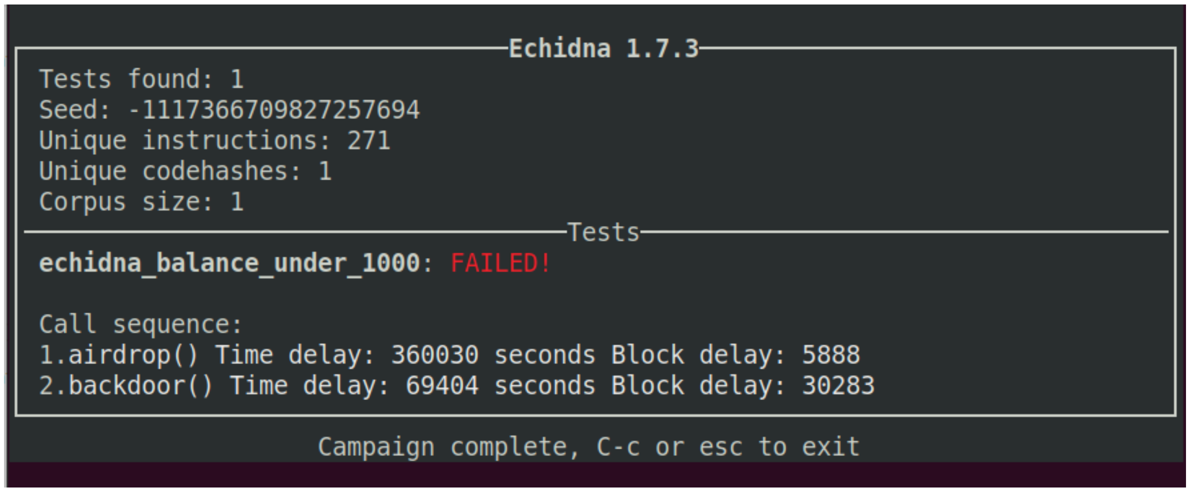


```
pragma solidity ^0.8.17;


contract Token{
      mapping(address => uint) public balances;
      function airdrop() public{
          balances[msg.sender] = 1000;
     }
     function consume() public{
          require(balances[msg.sender]>0);
          balances[msg.sender] -= 1;
     }
     function backdoor() public{
          balances[msg.sender] += 1;
     }
   }

contract TestToken is Token{
    constructor() public {}
        function echidna_balance_under_1000() public view returns(bool){
            return balances[msg.sender] <= 1000;
        }
}
```

- Output by the tool(GUI):

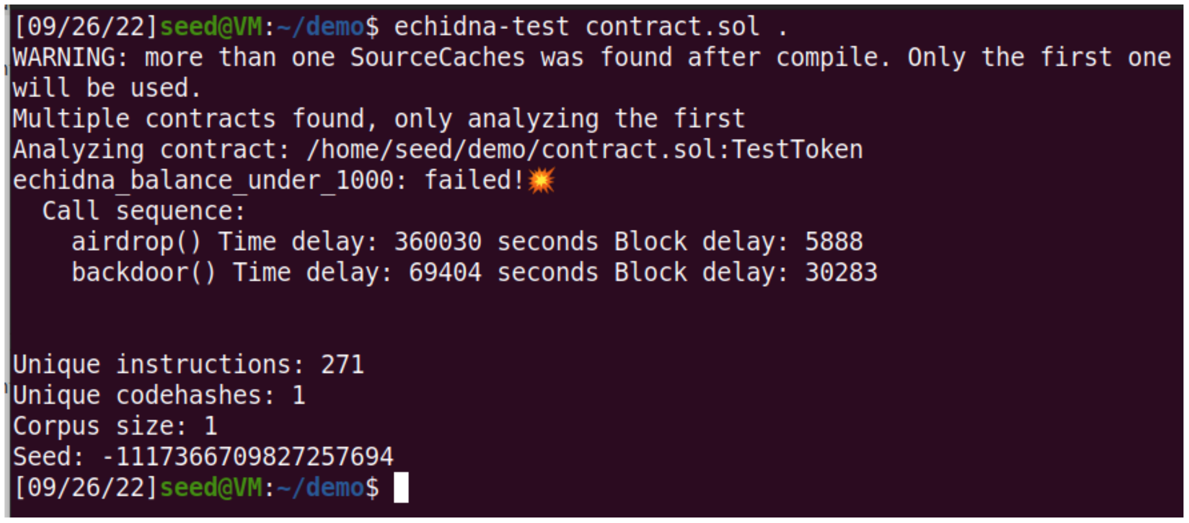


```
Echidna 1.7.3
Tests found: 1
Seed: -1117366709827257694
Unique instructions: 271
Unique codehashes: 1
Corpus size: 1
Tests
echidna_balance_under_1000: FAILED!

Call sequence:
1.airdrop() Time delay: 360030 seconds Block delay: 5888
2.backdoor() Time delay: 69404 seconds Block delay: 30283

Campaign complete, C-c or esc to exit
```

- Command Line Output:



```
[09/26/22]seed@VM:~/demo$ echidna-test contract.sol .
WARNING: more than one SourceCaches was found after compile. Only the first one
will be used.
Multiple contracts found, only analyzing the first
Analyzing contract: /home/seed/demo/contract.sol:TestToken
echidna_balance_under_1000: failed!💥
  Call sequence:
    airdrop() Time delay: 360030 seconds Block delay: 5888
    backdoor() Time delay: 69404 seconds Block delay: 30283


Unique instructions: 271
Unique codehashes: 1
Corpus size: 1
Seed: -1117366709827257694
[09/26/22]seed@VM:~/demo$ 
```

Comparison with Related Apps:

Feature	Echidna	Manticore
Execution	Improved automation will make it faster	Fast execution
Interface	Retro UI	Python API
Input	Tailored based on your code	Automatically produce concrete inputs
Setup	Easy and simple	Complex and requires effort
Coverage	Improvement will broaden the code coverage	Comprehensive
Automatic test case reduction	Available	Not available

Customers and users:

- Smart contract auditors from the following domains can perform audits whenever requested.
    - Finance, Gaming, Healthcare, Real estate etc;

References:

- C, C. (n.d.). *GitHub - crytic/echidna: Ethereum smart contract fuzzer*. GitHub. Retrieved September 12, 2022, from https://github.com/crytic/echidna
- Korobeinikov, A. (2021, August 3). 9 Most Common Smart Contract Vulnerabilities Found By. Blaize. Retrieved September 25, 2022, from https://blaize.tech/article-type/9-most-common-smart-contract-vulnerabilities-found-by-blaize/
- Hartlage, C. (2022, September 15). Why You Should Combine Symbolic Execution and Fuzzing. Code Intelligence. Retrieved September 26, 2022, from https://www.code-intelligence.com/blog/using-symbolic-execution-fuzzing
- Manticore: Symbolic execution for humans. (2017, August 10). Trail of Bits Blog. Retrieved September 26, 2022, from https://blog.trailofbits.com/2017/04/27/manticore-symbolic-execution-for-humans/
- Use our suite of Ethereum security tools. (2018, October 19). Trail of Bits Blog. Retrieved September 26, 2022, from https://blog.trailofbits.com/2018/03/23/use-our-suite-of-ethereum-security-tools/
- Builder, A. T. (2022c, May 8). Installing Echinda-test on MacOS. DEV Community 👩‍💻👨‍💻. Retrieved September 26, 2022, from https://dev.to/abhinavmir/installing-echinda-test-on-macos-c0h
- "[Errno 2] No such file or directory: “solc” " Using py-solc and solidity. (2017, September 11). Ethereum Stack Exchange. Retrieved September 26, 2022, from https://ethereum.stackexchange.com/questions/26192/errno-2-no-such-file-or-directory-solc-using-py-solc-and-solidity
- Build software better, together. (n.d.-b). GitHub. Retrieved September 26, 2022, from https://github.com/crytic/building-secure-contracts/blob/master/program-analysis/echidna/how-to-test-a-property.md+https://github.com/crytic/slither
- GitHub - crytic/slither: Static Analyzer for Solidity. (n.d.). GitHub. Retrieved September 26, 2022, from https://github.com/crytic/slither
- Releases · crytic/echidna. (n.d.). GitHub. Retrieved September 26, 2022, from https://github.com/crytic/echidna/releases