INCEPTION DOCUMENTATION

Github link: https://github.com/saikiran-ksk/Advanced-SE

Echidna Introduction: Echidna is a Haskell program designed for fuzzing/property-based testing of Ethereum smarts contracts. It uses sophisticated grammar-based fuzzing campaigns based on a contract ABI to falsify user-defined predicates or Solidity assertions.
It is designed with modularity in mind, so it can be easily extended to include new mutations or test specific contracts in specific cases[2] .

Features of Echidna: It's the first-ever fuzzer to target smart contracts, and has powerful features like abstract state-machine modeling and automatic minimal test case generation.[3]
Echidna has echidna-test executable which makes it capable of finding bugs within no time.[3]
Generates inputs tailored to your actual code.[2]
Optional corpus collection, mutation and coverage guidance to find deeper bugs.[2]
Powered by Slither to extract useful information before the fuzzing campaign.[2]
Source code integration to identify which lines are covered after the fuzzing campaign.[2]
Automatic testcase minimization for quick solution.[2]

Using Echidna: The core Echidna functionality is an executable called echidna-test. echidna-test takes a contract and a list of invariants (properties that should always remain true) as input. For each invariant, it generates random sequences of calls to the contract and checks if the invariant holds. If it can find some way to falsify the invariant, it prints the call sequence that does so. If it can't, you have some assurance the contract is safe.[2]

Writring Invariants: Invariants are expressed as Solidity functions with names that begin with echidna_, have no arguments, and return a boolean.[3]
For example, if you have some balance variable that should never go below 25, you can write an extra function in your contract like this one:

```
function echidna_check_balance() public returns (bool) {
        return(balance >= 25);[2].
```

Limitations: The debug information can be insufficient, when Echidna finds a property failure, It will be nice to have additional information on what Echidna is doing.
The other limitation is Echidna works on vyper contracts running the .vy files directly, but properties should be hardcoded in the contract.
Solidity generally allows to define libraries. If libraries contain only internal functions, Echidna will work fine. If there are any external functions, this tool only support linking libraries when solc is directly using solcLibs argument. In any other cases the result would be unexpected.
If at all the contract isn't properly linked, Echidna will crash.
Echidna will not detect the assertion failure in the internal transaction. This limitation is important as it can give confidence in the codebase, while assertions fail.

Project Enhancement: Make Echidna work with all contract type
Make Echidna work approrpiately when contracts are not properly linked
]Give additional information on debug logs in Echidna

References:

- https://blog.trailofbits.com/2018/03/09/echidna-a-smart-fuzzer-for-ethereum/ [1]
- https://hakin9.org/echidna-ethereum-smart-contract-fuzzer/ [2]
- https://github.com/crytic/echidna [3]

Team 6
Oluwadamilola Ajayi, 1001987642
Chandra Sujith Reddy Aramalla,
1001916647 Saikiran Kolli,
1001960017 Kavya Rayala,1001967276