Benchmarks and Vulnerabilities of DeFi

Kuldeep Meena (2019CS10490)

May 27, 2021

0.1 Introduction

DeFi or Decentralised Finance is an open, permissionless, and highly interoperable protocol stack built on public smart contract platforms, such as the Ethereum blockchain. Instead of relying on intermediaries of traditional financial instruments like brokerages, exchanges, or banks it depends on open protocols and DApps.The agreement between two parties are enforced by code that executes all the transactions in a way that is verified by all the nodes in a blockchain and this is secure. The design is such that there is no central authority.
Smart Contracts are central to a DeFi infrastructure.Smart contract is a code in a high level language that is executed on blockchain to complete a transction automatically and is verified by large number of validators. Since the smart contracts are to be verified by large number of validators the throughput is often slow as compared to the system of servient client architecture where we trust the server or a central authority but because of validations the transaction is much more secure.

0.2 Benchmarks of DeFi

- Transparency: Since DeFi is a blockchain bases infrastructure so transactions are publicly observable and smart contract code can be analysed on chain.
- Accessibility:

0.3 Security vulnerabilities of DeFi

- Smart Contract Vulnerability: Smart Contract Vulnerability risk is the risk caused due to an incorrect code of smart contract or an attacker using well know attack vectors to alter functionalities of a smart code. The most famous of such smart contract attacks is 'The DAO' attack which used the re-entrancy vulnerability and the cost to about 60 million dollars of ETH.
  Some Smart Contract Vulnerability Risks are as follows:
  - **Re-Entrancy Vulnerability**: The vulnerability occurs because of incoorect code. If ETH are send before state chsnge it might be possible that the target adddress is itself a contract that might call for requesting ETH again causing a cycle without state change and

possiblity of extracting huge amounts of ETH. The 'DAO' attack utilised this vulnerability and became the worst attack so far.

- **Unhandled Exceptions Vulnerability**: A smart contract does not work independently and often calls another contracts' function to complete the functionality, calls are made by sending istructions with a refrence to the contract.If an exception is raised it results in termination of contract and reverting to original state. But some low level operations in Solidity such as send doesn't throw an exception but instead return a boolean to mark success or failure. If the return value is unchecked in the caller function it might continue execution even when the payment failed causing inconsistencies.
- **Integer Underflow/Overflow Vulnerability**: The risk is common to not onyl Solidity but to most programming languages and is the most common vulnerability in most Smart Contracts.The Solidity compiler does not trigger any error flag to resolve the code with integer overflow/underflow problem.If an integer variable is assigned to a value larger than the number for the assigned range for example in the cause uint8 max value is 255, it resets to 0; if the variable assigned to a value less than the range, it would be reset to the top value of the range.If a loop counter were to overflow, creating an infinite loop, the funds of a contract could become completely frozen. This can be exploited by an attacker if he has a way of incrementing the number of iterations of the loop, for example, by registering enough users to trigger an overflow. 'Proof of Weak Hands' is the famous hack that utilised the risk. The risk is more in case of smart contracts where values goes to 0 and changing it to negative might make it the largest number.

- Accessibility:

0.4 References

1. https://www.geeksforgeeks.org/hungarian-algorithm-assignment-problem-set-1-introduction/
2. https://en.wikipedia.org/wiki/Hungarian_algorithm