**INTRODUCTION**

With blockchain, you can send money to someone else without a bank or a third party getting in the way. It's just you and the person who is receiving your transactions within the decentralized networks of blockchain.

Ethereum is an open-source, public, blockchain-based distributed computing platform featuring smart contract functionality. It provides a decentralized virtual machine that can execute peer-to-peer contracts using a cryptocurrency called Ether. Ethereum has its own programming language called Solidity which is used to create smart contracts. These programs allow you to set up rules and penalties around an agreement so that they are automatically enforced by the network.

**RELATED WORK**

In the paper [1], authors present a number of fresh security issues where a malicious party might influence the execution of a smart contract in order to profit. These flaws point to minute weaknesses in the platform's underlying distributed semantics knowledge. They suggest approaches to improve Ethereum's operational semantics in order to make contracts more secure. They created a symbolic execution tool called Oyente to help contract writers for the current Ethereum system detect possible security flaws. Oyente classifies 8, 833 of the 19, 366 currently in use Ethereum contracts as insecure, including the TheDAO flaw that cost the company $60 million in June 2016. Several case studies with published source code that confirm the assaults in the primary Ethereum network are also discussed, along with the severity of additional attacks for each.

Studies have shown security flaws, faults, and vulnerabilities in the Ethereum smart contract. The only method to end a contract on the blockchain system and transfer all of the Ethers in the contract balance is to use the Selfdestruct function. As a result, when issues are found, many developers utilize this function to terminate the contract and deploy a new one. A deep learning-based algorithm was used in one study [2] to retrieve the updated version of a destroyed contract in order to identify security flaws in Ethereum smart contracts. Then, using open card sorting, we look for security flaws in the upgraded versions. In another study [3] Authors personally found their defined contract flaws in 587 actual smart contracts by examining Feedback; they then made their dataset available to the public. Finally, they listed five effects brought on by contract flaws. These aid developers in comprehending the problems' symptoms and order of importance for elimination. Another study’s say [4] In addition to providing a thorough taxonomy of all known security concerns and reviewing the security code analysis techniques used to spot known vulnerabilities, the authors explore the topic of security of smart contract programming. We investigate security code analysis tools on Ethereum by evaluating their efficacy and precision on known problems on a representative sample of weak contracts. We tested the effectiveness of four security tools—Oyente, Securify, Remix, and SmartCheck—as well as 21 secure and 24 susceptible contracts to determine how well-rounded modern security analysis tools are for Ethereum.

The author gives [5] Their research sheds light on the smart-contract security field. We offer an up-to-date taxonomy of vulnerabilities, their architectural categorization, in conjunction with their severity level, based on extensive study and extensive experimentation on security flaws and code analysis tools. Additionally, they test a number of security technologies to determine their precision, potency, and consistency. They have categorized the tools according to the technique they are able to provide a "state of the art" set of security tools on Ethereum thanks to their user interface and their ability to do analysis. In order to find gaps and missing vulnerability checks, they build a matrix of security tools and the vulnerabilities they cover. In another work [6], they provide a security assurance approach for smart contracts in order to prevent potential vulnerabilities while creating smart contracts. Their solution has two essential components. Developers can benefit from clearer understanding of their code structure with the topology diagram creation of invocation relationships for cross-file smart contracts. In addition, they broaden the range of logical dangers and may identify and pinpoint them using syntax analysis and symbolic execution. Together with syntax analysis, they can identify logic vulnerabilities to certain functions. They created the SASC tool based on these characteristics. It can offer Ethereum smart contracts very strong quality assurance.

**REFERENCE**

1. Loi Luu, Duc-Hiep Chu, Hrishi Olickel, Prateek Saxena, and Aquinas Hobor. 2016. Making Smart Contracts Smarter. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16). Association for Computing Machinery, New York, NY, USA, 254–269.
2. J. Chen, "Finding Ethereum Smart Contracts Security Issues by Comparing History Versions," 2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE), 2020, pp. 1382-1384.
3. J. Chen, X. Xia, D. Lo, J. Grundy, X. Luo and T. Chen, "Defining Smart Contract Defects on Ethereum," in IEEE Transactions on Software Engineering, vol. 48, no. 1, pp. 327-345, 1 Jan. 2022, doi: 10.1109/TSE.2020.2989002.
4. R. Sujeetha and C. A. S. Deiva Preetha, "A Literature Survey on Smart Contract Testing and Analysis for Smart Contract Based Blockchain Application Development," 2021 2nd International Conference on Smart Electronics and Communication (ICOSEC), 2021, pp. 378-385, doi: 10.1109/ICOSEC51865.2021.9591750.
5. A. Dika and M. Nowostawski, "Security Vulnerabilities in Ethereum Smart Contracts," 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), 2018, pp. 955-962, doi: 10.1109/Cybermatics\_2018.2018.00182.
6. E. Zhou et al., "Security Assurance for Smart Contract," 2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS), 2018, pp. 1-5, doi: 10.1109/NTMS.2018.8328743.