**PHANTOM BLOCKS– Attacks that leave trails behind**

A Project Report Submitted

in Partial Fulfillment of the Requirements for the Degree of

**BACHELOR OF TECHNOLOGY**

in

**COMPUTER SCIENCE AND ENGINEERING**

*by*

**SAI KRISHNA POLAVARAPU**

**(Roll No. CS21B1020)**

#### 

#### *To*

#### ***DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING***

**INDIAN INSTITUTE OF INFORMATION TECHNOLOGY RAICHUR-584135, INDIA**

#### *APRIL 2025*

**DECLARATION**

I, **SAI KRISHNA POLAVARAPU** (**Roll No: CS21B1020**), hereby declare that, this report entitled **“PHANTOM BLOCKS– Attacks that leave trails behind'' submitted** to Indian Institute of Information Technology Raichur towards partial requirement of **Bachelor of Technology** in **Computer Science and Engineering Department** is an original work carried out by me under the supervision of **Prioyduti Pradhan** and has not formed the basis for the award of any degree or diploma, in this or any other institution or university. I have sincerely tried to uphold academic ethics and honesty. Whenever an external information or statement or result is used then, that has been duly acknowledged and cited.

Raichur-584135 [SAI KRISHNA POLAVARAPU](mailto:cs21b1020@iiitr.ac.in)

May 2025

**CERTIFICATE**

This is to certify that the work contained in this project report entitled **“PHANTOM BLOCKS– Attacks that leave trails behind”** submitted by **SAI KRISHNA POLAVARAPU**(**Roll No: CS21B1020**) to the Indian Institute of Information Technology Raichur towards partial requirement of **Bachelor of Technology** in **Computer Science and Engineering** has been carried out by him under my supervision and that it has not been submitted elsewhere for the award of any degree.

Raichur-584135 (Dr. Priodyuti Pradhan)

May 2025 Project Supervisor

# **ABSTRACT:**

Due to its quick development, blockchain technology—especially Ethereum—has become a prime target for many kinds of malicious activity. By examining transaction-level data, this project aims to identify and describe possible attack areas within the Ethereum network. Patterns suggestive of anomalous behaviour were found by thoroughly examining Ethereum's architecture and a survey of known attack vectors. A set of crucial features, including the frequency of contract interactions, anomalies in gas prices, irregularities in transaction timing, and patterns of value transfer, were developed using these insights. In order to differentiate between legitimate and possibly malicious transactions, thresholds were subsequently established using statistical analysis and empirical observations. The resulting system facilitates the proactive identification of threats by enabling the flagging of areas of suspicious activity. This approach provides a foundation for enhancing Ethereum's security monitoring and opens avenues for real-time anomaly detection in decentralized environments.

**Contents**

**List of Figures XXIX**

**1 Chapter-1 Introduction**  7

**2 Chapter-2 Ethereum**

2.1 Externally Owned Accounts: 8

2.2 Smart Contracts: 9

2.3 Working 10

**3 Chapter-3 Attacks on Ethereum**

**Attacks**

2.1 DOA Hack 11

2.2 ICO Mania 13

2.3 Crypto-Kitties 15

2.4 Govern-Mental contract attack 17

**3 Chapter-3 Vulnerabilities**

3.1 Spam Account Creation 19

3.2 Out Of Gas Transactions 20

3.3 Revert Transactions 22

**4 Chapter-4 Dataset**

4.1 Data Collection 24

4.2 Data Preprocessing 24

**5 Chapter-5 Results**

5.1 Spam Account Creation 26

5.2 Out Of Gas Transactions 28

5.3 Revert Transactions 29

**6 Chapter-6**

6.1 Conclusion 32

6.2 Future work 33

6.3 References 34

# **INTRODUCTION:**

Digital transactions and decentralised applications have been revolutionised by Ethereum, one of the most popular blockchain technologies. Unlike traditional centralised systems, Ethereum runs on a decentralised network that employs accounts to store and transfer assets. These accounts fall into two categories: smart contracts and externally owned accounts (EOAs). Smart Contracts are self-executing code that is implemented on the blockchain to automate processes, while EOAs, which frequently represent people or organisations, are governed by private keys. For many applications, such as transaction analysis, fraud detection, and resource optimisation in blockchain networks, determining the type of account is essential. However, because of its decentralised structure, Ethereum is vulnerable to misuse, including hacking, fraud, and other things.

The objective of this project is to examine Ethereum transaction data in order to spot trends and actions that could point to potential attack scenarios. We identified a number of characteristics from raw transaction data that can be used as markers of unusual activity by carefully examining Ethereum's architecture and attack history. These characteristics include things like the frequency of transactions, the account creation and anomalies in revert and out of gas transactions.  
  
We established threshold values for these features based on data-driven analysis and empirical observation to improve detection even more. Transactions or sets of transactions that surpass these limits are identified as possible areas for attack. The suggested approach offers a useful framework for identifying questionable activity, providing insightful information for Ethereum security monitoring and attack avoidance.

# **ETHEREUM:**

By enabling programmable smart contracts, Ethereum, a decentralized blockchain platform first presented by Vitalik Buterin in 2015, expands the potential of cryptocurrencies such as Bitcoin. At its core, Ethereum employs accounts to interact with the blockchain network and store assets. Ethereum's support for decentralized applications (dApps) makes it more flexible than Bitcoin, which is only a cryptocurrency. Ethereum accounts fall into two main categories:

## **Externally Owned Accounts:**

EOAs are user-managed accounts that are private key-controlled. They stand in for people or businesses that start transactions on the Ethereum network.

* **Key Features:**
* **Ownership:** Directly managed by private keys. Complete access to the account is possible with a working private key.
* **No Code:** Neither logic nor code are present in EOAs. They are only utilised to store and move assets or to start conversations with smart contracts.
* **Transactions**: EOAs can send and receive Ether (ETH) or ERC- 20 tokens.
* **Gas Fees:** An EOA must pay a gas fee in ETH for each transaction they start.
* **Security:** The private key determines an EOA's level of security. Access to the account is lost in the event that the private key is lost or compromised.

## **Smart Contracts:**

Accounts connected to self-governing code implemented on the Ethereum blockchain are known as smart contracts. When certain circumstances are met, these contracts carry out predetermined activities. They cannot start transactions on their own; they need to be contacted by EOAs or other smart contracts.

* **Key features:**

* **Code-Driven:** High-level programming languages like Solidity or Vyper are used to create smart contracts. The Ethereum Virtual Machine (EVM) is where the code is installed.
* **Autonomous:** Smart contracts function independently of direct human involvement once they are put into place. Deterministic execution adheres to the guidelines specified in the code.
* **Immutable:** A smart contract's code is unchangeable after it has been deployed, guaranteeing dependability and consistency. Nonetheless, certain design principles can be used to achieve upgradability.
* **Interactions:** Smart contracts have the ability to store information, carry out calculations, and communicate with other EOAs or smart contracts.  
  They enable flexible and composable systems by calling functions in other smart contracts.
* **Dependency on Transactions:** A smart contract is unable to start a transaction on its own. An EOA or similar smart contract must initiate it.

## **Working:**

A transaction in Ethereum starts when a user creates it in a wallet or application and signs it with their private key. Important information like the sender and recipient addresses, the amount of data or Ether being transferred, the gas price, and the gas limit are all included in the transaction. After being signed, the transaction is sent out to the Ethereum peer-to-peer network, where it is checked for sufficient balance, correct nonce usage, and authenticity. Valid transactions are kept in a pool known as the mempool until they are picked up by a miner (in the previous Proof of Work model) or a validator (in Ethereum's current Proof of Stake system). Based on petrol fees, validators choose transactions, giving preference to those with greater incentives. The selected transactions are combined into a block, which is subsequently confirmed and appended to the blockchain upon consensus. The transaction is deemed confirmed and becomes an unchangeable component of the Ethereum ledger once it is incorporated into a block.

# **ATTACKS ON ETHEREUM:**

Being a popular blockchain platform, Ethereum has frequently been the target of different types of attacks. These include phishing schemes, denial-of-service attacks, transaction spamming, smart contract exploits, and gas price manipulation. These attacks seek to take advantage of system behaviour, steal money, or interfere with the network. Enhancing Ethereum's security and guaranteeing the secure implementation of decentralised applications require an understanding of these threats. The following sections explore these attacks in detail.

## **DAO Hack:**

The June 2016 DAO (Decentralised Autonomous Organisation) hack was one of the biggest and most notable attacks on the Ethereum network. The Ethereum blockchain served as the foundation for the DAO, a smart contract-based venture capital fund that facilitated decentralised project proposal and voting by investors. It became well-known very fast, and thousands of investors contributed more than $150 million in Ether (ETH), making it one of the biggest crowdfunding campaigns at the time.

But on June 17, 2016, an attacker took advantage of a serious flaw in the DAO's smart contract. The vulnerability was associated with a recursive call problem, also known as a re-entrancy vulnerability. The attacker was able to drain roughly 3.6 million ETH (roughly $60 million USD at the time) into a child DAO under their control by repeatedly requesting fund withdrawals before the contract could update the user's balance.

The Ethereum community experienced a serious crisis as a result of the hack. The Ethereum community made the decision to enact a hard fork, or permanent departure from the current blockchain protocol, in order to stop the attacker from keeping the stolen money indefinitely. As part of this process, many revert transactions were executed to undo the effects of the hack and return the stolen funds to their original owners. On July 20, 2016, this fork was carried out, essentially restoring the Ethereum blockchain to its pre-attack state and giving the original investors their money back. The network consequently divided into two distinct chains:

* **Ethereum (ETH)**: the new chain with the hard fork that nullified the DAO hack.
* **Ethereum Classic (ETC)**: the original, unforked chain that continued to uphold the principle of immutability.

The Ethereum ecosystem was significantly impacted by the DAO hack and the hard fork that followed. The hard fork generated a discussion about decentralisation and immutability even though it assisted in recovering the stolen money and preserving investor confidence. The blockchain community experienced a philosophical split when Ethereum Classic (ETC) emerged, carrying on the original, unmodified blockchain. Blockchain ethics and protocol updates are still influenced by this incident, which brought to light the difficulties in striking a balance between security, governance, and trust in decentralised systems.

## **ICO Mania:**

One of the most significant events in Ethereum's history was the Initial Coin Offering (ICO) Mania, which mostly occurred in 2017 and the first part of 2018. Ethereum served as the foundation for thousands of new blockchain-based initiatives during this time that sought to raise money by issuing their own tokens through initial coin offerings (ICOs). For quick and simple token creation and distribution, these projects made use of Ethereum's ERC-20 token standard.

In an initial coin offering (ICO), investors are usually offered a newly created token in exchange for Ether (ETH). Users would receive tokens at a predetermined rate in exchange for sending ETH to a smart contract address. Most initial coin offerings (ICOs) established a hard cap, or the maximum amount of Ethereum they hoped to raise, in order to control fundraising limits. The smart contract would frequently reverse the transactions and give the contributors their ETH back if the hard cap was not reached within the allotted time.

Many projects used airdrops and giveaways to draw in investors and generate buzz. These involved giving users free tokens in return for doing basic tasks like following accounts on social media, joining Telegram groups, or setting up Ethereum wallets. Even prior to the token launch, this tactic was frequently employed to raise awareness and acquire a sizable user base.

As a result, the number of Ethereum accounts created during this period skyrocketed. Users only needed to generate a key pair using free tools or browser extensions like MetaMask to create a new Ethereum address (wallet), which was cheap and didn't require verification. Users frequently made multiple wallets to claim more free tokens because each airdrop usually only permitted one entry per address, which further inflated the number of accounts.

Particularly during high-demand ICOs when users scrambled to participate before the hard cap was reached, the surge of users and transactions resulted in more network congestion and an increase in revert transactions. The smart contract would automatically reject or reverse a user's transaction if it arrived late or exceeded the cap. This led to transaction failures and increases in gas prices, underscoring Ethereum's scalability constraints at the time.

Both the advantages and disadvantages of decentralised fundraising were illustrated by the ICO boom. It made it possible for creative projects to raise money without the need for middlemen, but it also brought about fraud, regulatory attention, and unsustainable hype. However, the time frame had a big impact on Ethereum's ecosystem and uptake, making it one of the most popular platforms for token creation and decentralised finance.

## **Crypto-kitties:**

The blockchain-based game Crypto-Kitties, which was released in late November 2017, is among the first and most prominent instances of an application putting the Ethereum network under unanticipated strain. Crypto-Kitties, an Ethereum-based platform, enabled users to purchase, breed, and exchange distinct digital cats in the form of ERC-721 non-fungible tokens (NFTs). Every "Kitty" had unique characteristics, and some gained significant value as a result of demand and scarcity.

Crypto-Kitties gained enormous attention from the mainstream media and the cryptocurrency community after going viral very quickly. Referral bonuses in the game encouraged players to invite friends in order to receive in-game rewards. Users flooded the network with transactions pertaining to buying, breeding, and trading digital cats as its popularity skyrocketed. Each of these actions set off smart contract executions, which frequently necessitated interactions with on-chain assets and several transactions per breeding cycle.

Due to this abrupt and unheard-of demand, the Ethereum network experienced an unusual load, which led to significant network congestion. Confirmation times increased dramatically, and transaction queues grew quickly. In order to prioritise their transactions, users had to pay more for gas, which increased the network's overall transaction fees. Block space congestion caused delays for even unrelated Ethereum-based apps and initial coin offerings.

According to reports, the game was the most popular decentralised application on the Ethereum network at the time of its peak, accounting for over 10% of all Ethereum traffic. Ethereum's scalability issues were recognised by both developers and users, underscoring the pressing need for more effective consensus techniques.  
  
The Ethereum ecosystem learnt a valuable lesson from the crypto-Kitties incident. It illustrated the need for improved transaction throughput, fee structures, and network optimisation techniques, as well as how even an enjoyable, collectible game could put strain on a global decentralised network.

## **Govern-Mental contract attack:**

One of the earliest Ethereum-based Ponzi schemes that ultimately resulted in an aberrant and exploitative state on the network was the Govern-Mental contract. This contract, which had a structure resembling a pyramid or Ponzi scheme, came to light in the middle to end of 2016. Its basic working principle was straightforward but risky: new users would send Ether to the contract in exchange for a bigger return when more users joined after them. Early participants' payouts were totally reliant on later participants' deposits.

The contract's design created a loop of transactions that promoted continuous network activity by funding the previous participant in line with each new transaction. In the hopes that the cycle would be maintained by the flood of new deposits, users fought to be the next in queue. The system became increasingly complex as more users joined, with certain smart contract features causing more internal calls and resulting in a recursive transaction pattern.

The computational cost increased as activity increased. The contract eventually started to experience "out of gas" errors as a result of overly intricate loops and an oversized participant list. The recursive nature of the contract started to surpass Ethereum's gas limits, which are in place to prevent infinite loops and resource exhaustion. The system was essentially frozen as a result of numerous transactions failing and new users being unable to initiate payouts to prior users.

Because of gas limitations, users were unable to withdraw money or initiate additional transactions, resulting in a situation where funds were locked inside the contract. The trapped Ether brought to light the crucial problem with badly thought out or purposefully malicious smart contracts: once they are implemented on Ethereum, they cannot be altered without a hard fork or outside assistance.  
  
An early warning about the danger of contract immutability, gas inefficiency, and unsustainable incentive structures was provided by the Govern-Mental case. Additionally, it highlighted Ethereum's need for improved contract auditing, user education, and network protections against financial exploitation that could endanger user funds and the ecosystem's stability.

# **VULNERABILITIES:**

## **Spam account creation:**

The excessive creation of spam accounts has been one of the Ethereum ecosystem's subtle but significant vulnerabilities, especially during times of high traffic like ICO mania, airdrop campaigns, and the Crypto-Kitties craze. Ethereum, in contrast to conventional financial systems, permits anyone to create a wallet address without requiring identity verification. This process made it possible for users to easily create hundreds or even thousands of accounts because it is quick and computationally cheap.

This vulnerability was extensively exploited during initial coin offerings (ICOs) and airdrops. Users started creating multiple wallets to get around contribution caps and optimise token rewards because many projects restricted token distributions per address. This behaviour was particularly encouraged by airdrops, which gave away tokens to Ethereum addresses whether or not they were being used. The number of Ethereum accounts increased dramatically as a result of this practice, many of which were either dormant or used only for manipulation.

Similar to this, users occasionally made multiple wallets during the Crypto-Kitties boom in order to manage various digital assets or receive referral bonuses, which again led to needless account bloat. The volume of transactions and the size of the global Ethereum state grew with the rise in these spam accounts. Overall performance and user experience suffered as a result of the network becoming clogged, transaction processing slowing down, and petrol prices rising.

The Ethereum community and core developers came up with a number of solutions to address this issue. EIP-158, which sought to deactivate unused or empty accounts and lessen their influence on the network, was one of the initial responses. Ideas like state rent, which would require inactive accounts to pay fees to stay on the chain, and increased petrol costs for storage operations were introduced over time. These initiatives were essential to deterring spam and maximising Ethereum's efficiency and scalability.

## **Out of Gas transactions:**

"Out-of-gas" errors, which happen when a transaction or smart contract goes over its permitted computational limit, have been one of the Ethereum network's persistent technical issues. During times of high network activity, like the ICO Mania, the launch of Crypto-Kitties, airdrop-driven spam, and pyramid-style scams like the Govern-Mental contract, this issue was particularly apparent. Out-of-gas transactions surged on the Ethereum network during these events as users hurried to engage with crowded smart contracts or subpar Dapps. These spikes revealed flaws in Ethereum's gas management model in addition to contract design inefficiencies.

A type of Denial-of-Service (DoS) attack was a notable consequence of these out-of-gas spikes. By creating transactions that would use gas but never finish, malicious actors and flawed contracts started to take advantage of the gas mechanism. By filling blocks with unsuccessful operations and clogging the mem-pool, these transactions would essentially reduce the amount of space available for valid transactions. Miners were overloaded with unsuccessful or low-value transactions during attacks or unusual loads, wasting block space and lowering network throughput. In severe situations, this resulted in lengthy delays, unsuccessful interactions, and increased gas fees for Dapps and users in an effort to compete for block inclusion.

Misconfigured smart contracts, where programmers neglected to optimise loops or neglected gas limits in function calls, were a frequent source of these mistakes. Recursive loops and internal payouts that increased with each new user were features of the smart contract in situations such as the Govern-Mental pyramid attack. Newer transactions began to run out of gas as the payout chain eventually grew too big to fit inside the gas limit. The contract stalled as a result, permanently locking the funds inside because no subsequent transactions could be completed. Ether was lost and functionality was completely frozen as a result of users repeatedly failing to withdraw or interact with the contract.

Out-of-gas errors frequently result in indirect but substantial economic losses. These consist of stranded or unrecoverable Ether, lost transaction fees, and a deteriorated network-wide user experience. These incidents also raised concerns about Ethereum's scalability and dependability, which led developers to create gas-efficient systems and conduct more thorough smart contract audits. The Ethereum community's long-term response included enhancing gas estimation algorithms, suggesting EIPs to manage contract efficiency, and investigating Layer 2 solutions to reduce congestion.

## **Revert transactions:**

The high number of revert transactions—failed transactions in which the state changes are rolled back because of mistakes or unmet conditions—has also been a significant sign of anomalous activity and inefficiency in the Ethereum network. Around 2017, at the height of the ICO boom, when many smart contracts were hurriedly deployed to raise money, there was a discernible increase in revert transactions. Many of these contracts had to be reverted because they were badly written, lacked sound logic, or did not handle exceptions. Furthermore, more gas was required than expected due to intricate contract structures with nested loops or pyramid-like referral systems. These transactions would automatically revert when their petrol limits were reached in the middle of execution, wasting transaction fees and processing power.

Reverted transactions frequently filled blocks during this time, severely congested the network, and delayed the confirmation of legitimate transactions. These reverts weren't merely innocuous errors; if a transaction is improperly structured or doesn't revert correctly, it can cause the system to become inconsistent and balances or contract states may not update as intended. Particularly in contracts that do not gracefully handle partial execution, this creates the possibility of vulnerabilities or even exploit scenarios. Although the Ethereum protocol makes sure that unsuccessful transactions are undone in order to preserve atomicity, the sheer number of these operations led to performance snags and increased strain on nodes and miners. In order to prioritise successful transactions, users were frequently compelled to pay higher gas fees. The pattern of these revert spikes not only illustrates the immaturity of development at that time, but it also highlights how crucial gas-efficient architecture and secure contract development are to preserving Ethereum's stability and credibility.

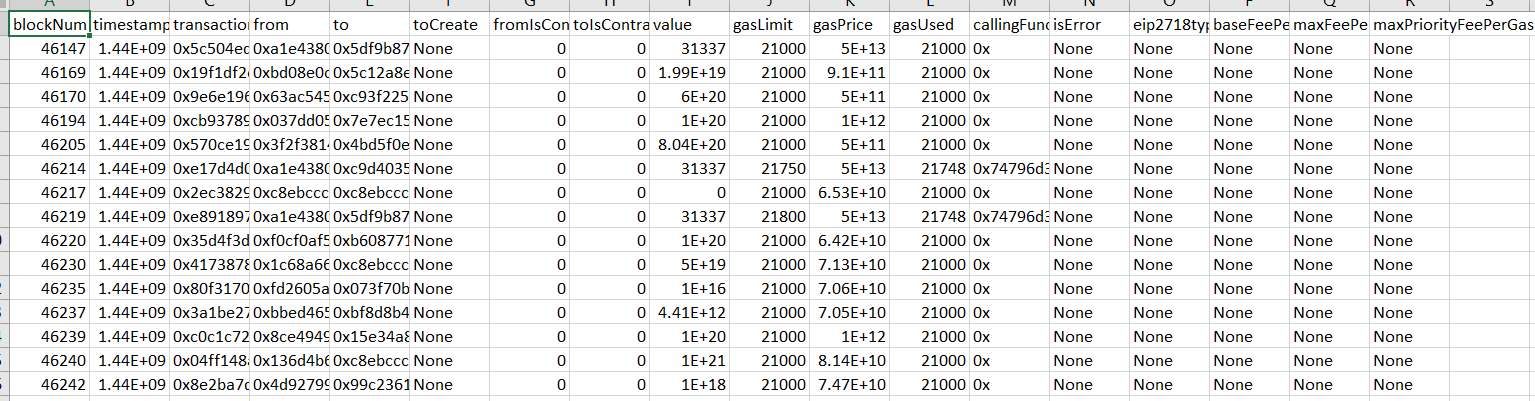
# **DATASET:**

## **Data Collection:**

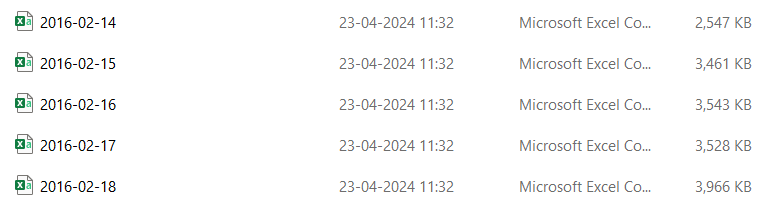
I have collected the data from <https://xblock.pro/xblock-eth.html> . And downloaded the block transaction data from it.

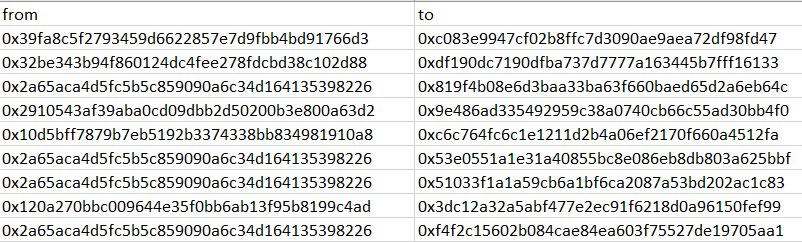
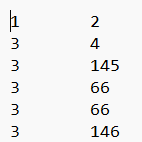
## **Data preprocessing:**

This is my initial dataset format.



The dataset was initially divided based on the exact day that each record was created in order to analyse data on a daily basis. By transforming the raw timestamps into a more readable date format, this division was achieved. This conversion made it easier to split down the data by day, enabling more precise analysis and easy day-to-day comparisons. I have considered the data from 2015-08-07 to 2019-01-02.

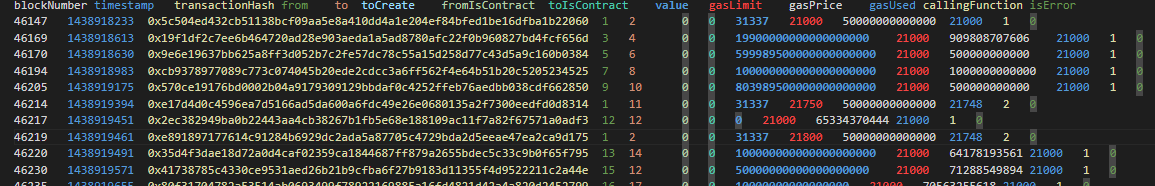
  
  
 We then created mappings to guarantee consistency throughout the collection and standardise important data elements. To monitor the flow of communications, transactions, or other pertinent activities between various entities within the dataset, "From" and "To" mappings were made. Building meaningful relationships and conducting accurate data analysis required this phase.

 🡪 

Similar to address mapping, I have mapped calling function and “isError” column also.

Reduce the columns and remember to convert the csv’s to text file format, for easy computation and consider only necessary columns.

This is how my preprocessed data looked like.

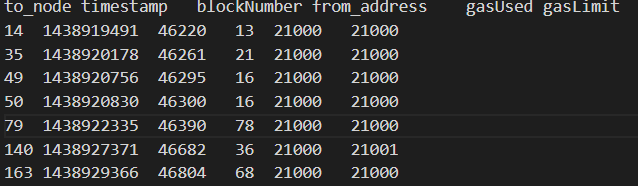
****

# **RESULTS**:

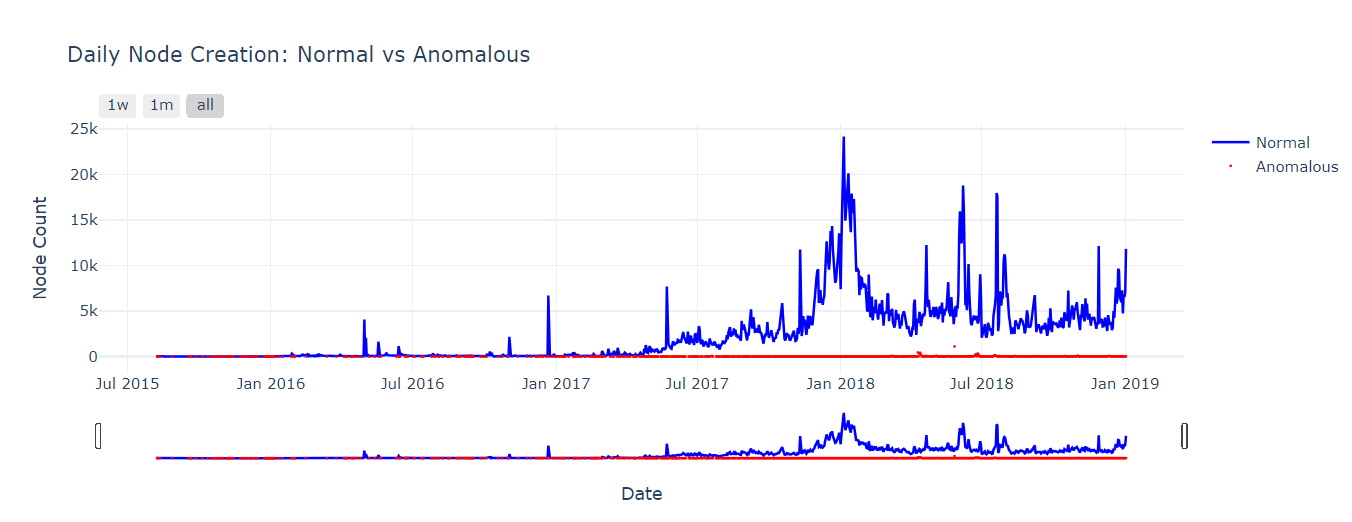
We have seen, many attacks and their patterns. There are several vulnerable patterns which we have noticed from these attacks, now we will try to detect these attacks using these vulnerabilities.

## **Spam account creation:**

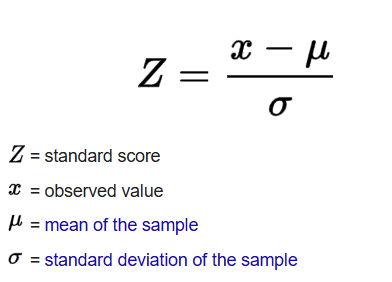
I have extracted the nodes which have been present in the to node position only once and never in from node. Here, these nodes are considered as newly created nodes in the network but considering this node never appeared again in to column, we can justify that it is created clearly for spamming purpose or to receive free rewards in any Dapps. Found 29,45,819 such accounts.



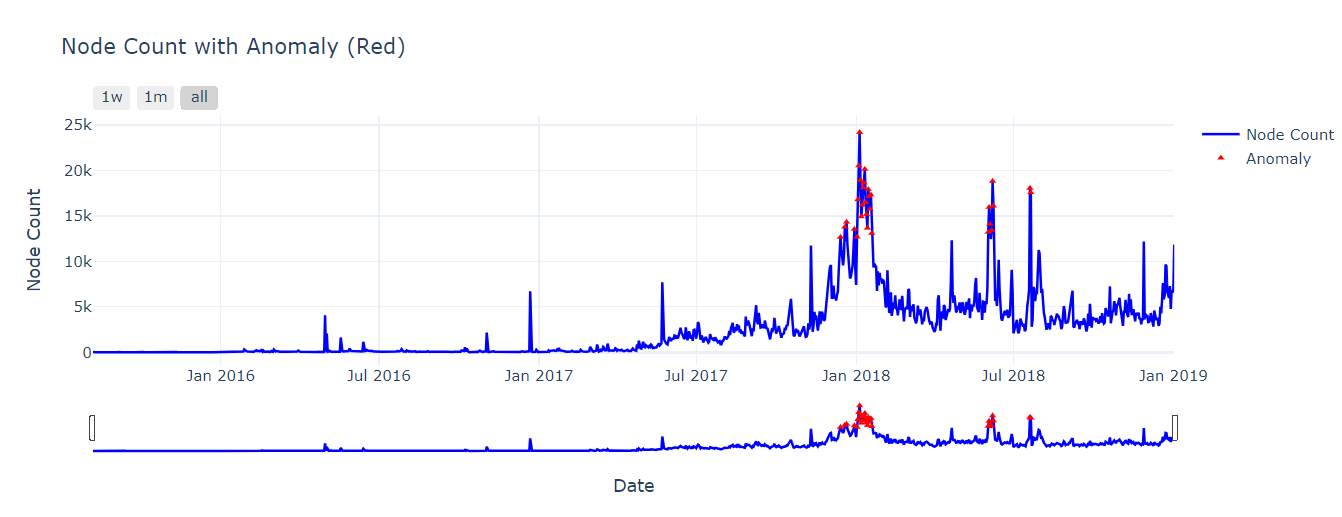
Later, I have extracted few columns of those nodes. And I tried to distinguish the anomalous nodes by passing timestamp, block number, gas limits, gas price etc as features for clustering them. Here, I have used K-means clustering and plotted anomalous node count vs day to identify of any pattern is present between them. But failed in finding any trend using this method (as shown in below figure).



Later, I tried to identify the pattern using Z-score. Here, the feature is node count for that particular day.



Here, I have considered various values like Z-score>2 and Z-score>3 as anomalous but when I tried to mark anomaly with Z-score>2, there is no particular trend or attack identification but when I tried to plot for Z-score>3, then it found anomalies on attack days. Approx 30 days were found to be anomalous.

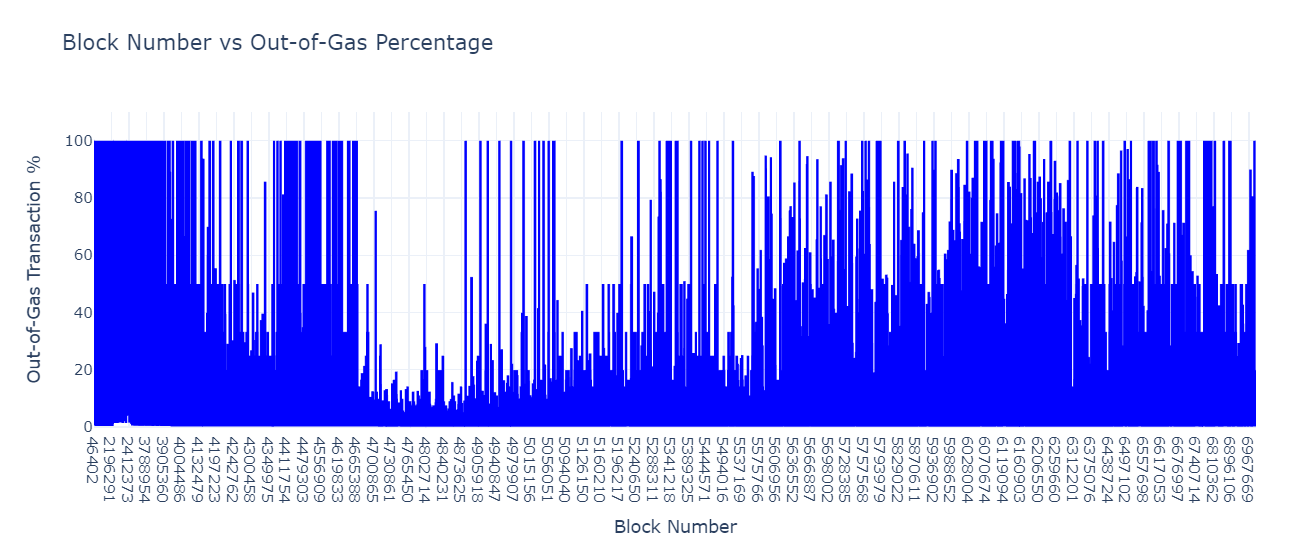


These markings are around days of ICO Mania, Crypto-kitties drawback peak times. So, this says that our threshold of Z-score>3 is validated and true.

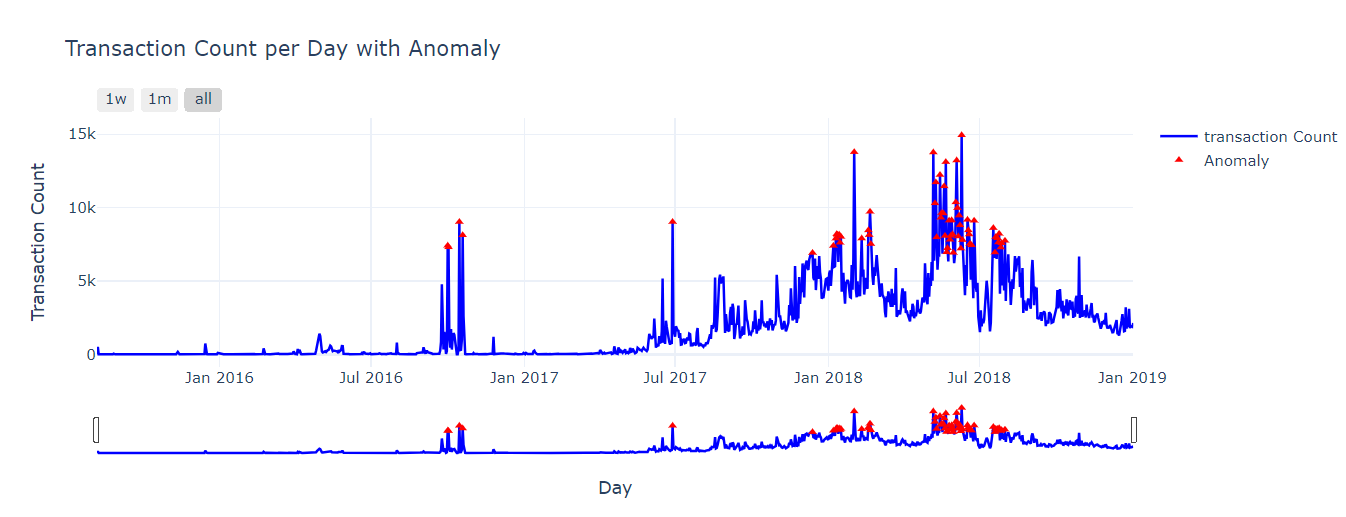
## **Out-of-Gas transactions:**

Here, I have extracted every transaction done until now, which has resulted as out of gas. This can be done in two ways, either by comparing gas used and gas limit of that transaction or by checking the is error column. There are 22,92,143 such transactions. For extracting the data, we stored block number and timestamp of it.

We tried to check, if there is any relation for block and percent of transactions in it being anomalous, but have found most of the blocks in which at least one such transaction is present, are almost full by these kind of transactions in most cases.



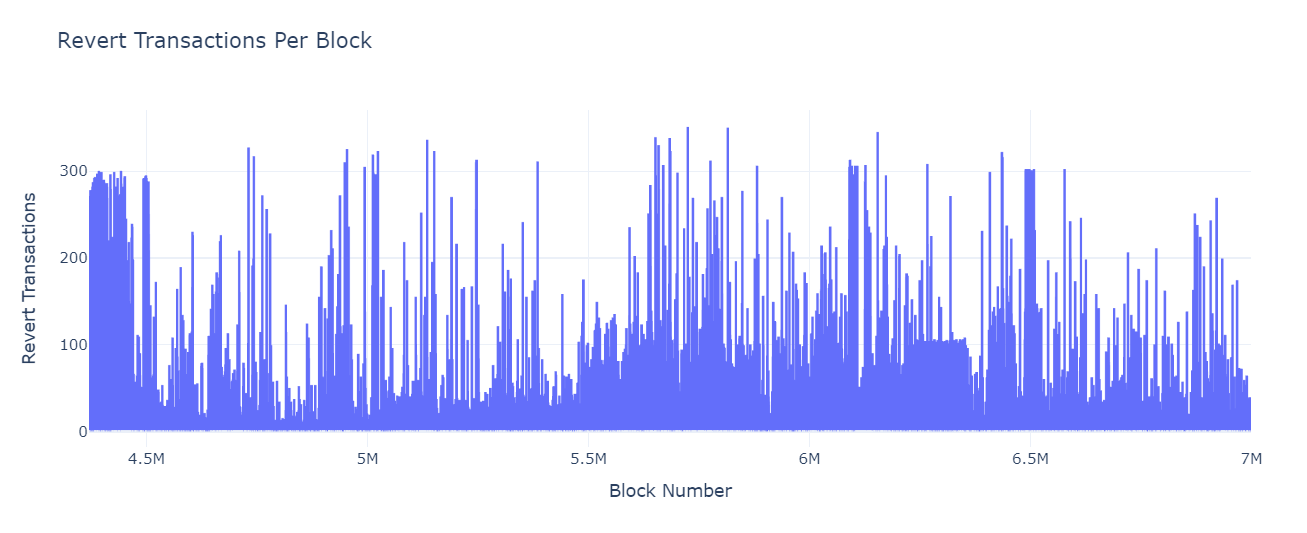
Later, I have plotted the day vs out of gas transaction count, and took Z-score>2, with out-of-gas transaction count per day as feature, so that the threshold taken satisfied many of the attacks days in which it is a key feature.

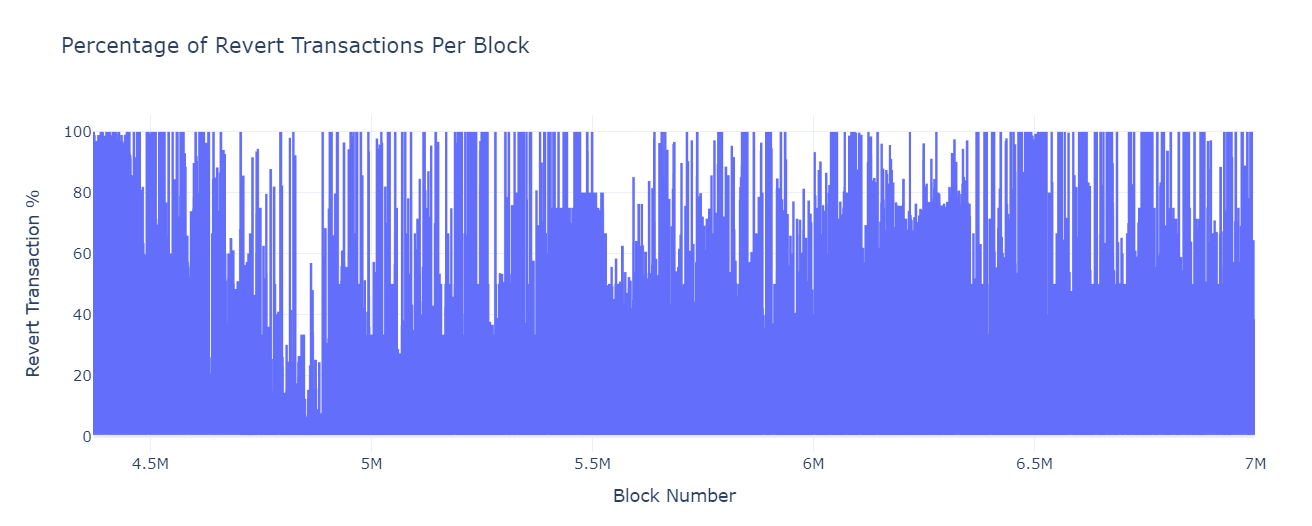


By this, I assure that the threshold taken is valid.

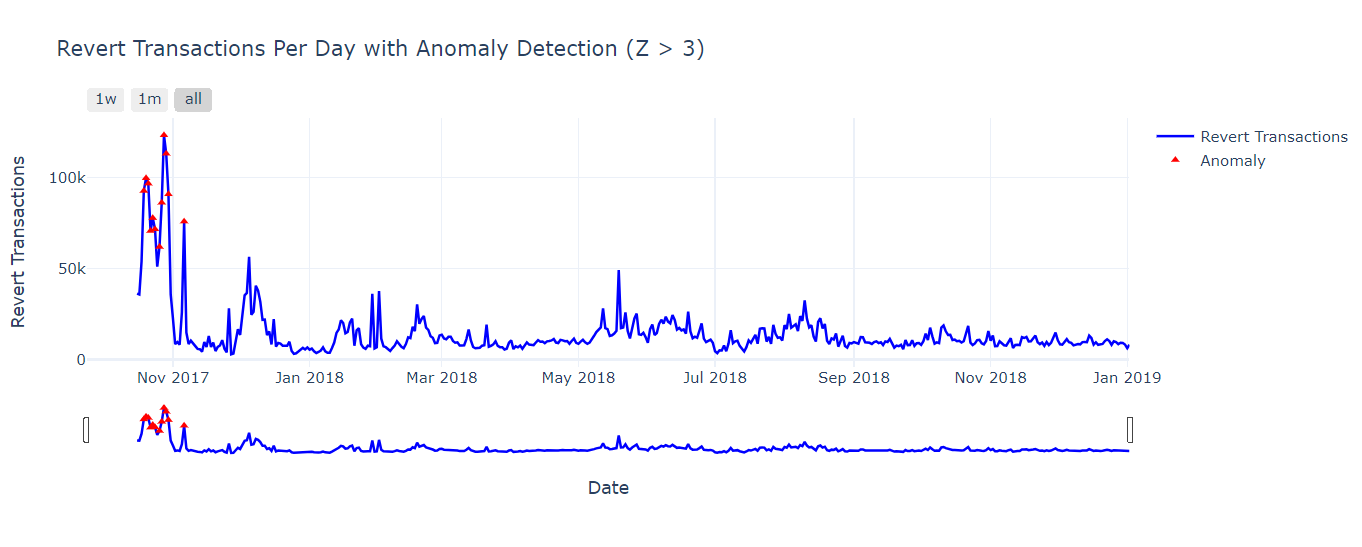
## **Revert Transactions:**

I have extracted each and every revert transaction occurred in the dataset by checking the is error column. I have extracted only timestamp and block number and found how many transactions occurred in each block where at least one revert transaction is present and found many of them are almost full which didn’t capture useful insights.





So, later I have considered Z-score>3, as threshold which helped in identifying anomalous transactions during the specific attacks. Here, revert transaction count per day is the feature for Z-score calculation.



After finding the anomalous days, I have validated it with attack days and it matched with ICO boom etc, which proved that the threshold taken is valid.

# **CONCLUSION:**

Throughout its development, the Ethereum ecosystem has encountered numerous significant obstacles and attack points, despite being revolutionary in enabling decentralised applications and smart contracts. Every incident, from the DAO hack's early days to the ICO Mania, Crypto-Kitties congestion, pyramid schemes, and gas-related DoS attacks, has revealed a new layer of vulnerability, ranging from transaction spamming and network-level inefficiencies to faulty contract logic and spam account creation. These events demonstrated the necessity of improved governance procedures, gas optimisation, and more robust contract auditing. The increase in out-of-gas failures, revert transactions, and anomalous behaviour patterns further demonstrated how the Ethereum network can be strained by ill-designed smart contracts and malicious user activity. However, these challenges have pushed the Ethereum community to continuously improve through protocol upgrades, EIPs, and the development of Ethereum 2.0 and Layer 2 solutions. While vulnerabilities may never be entirely eliminated in any decentralized system, Ethereum’s journey reflects a maturing ecosystem striving to balance scalability, security, decentralization, and trust.

# **FUTURE WORK:**

Even though this study has looked at common attack patterns, odd behaviours, and weaknesses in the Ethereum network, there is still a lot of room for improvement and additional research. Future research can concentrate on developing automated anomaly detection systems that instantly spot questionable transaction patterns by using data analytics and clustering. The network's resilience could be greatly increased by creating tools that can proactively flag malicious contract behaviour, spam account activity, or unusual gas usage.

Analysing Layer 2 solutions and Ethereum 2.0 in terms of how they address vulnerabilities that have already been identified is another crucial avenue. Researchers can determine whether these improvements lessen the frequency of revert transactions, gas-based attacks, and DoS vectors by examining transaction behaviour and smart contract execution under more recent consensus models, like Proof of Stake.  
  
Developers and users can also become more situational aware during attack periods by putting in place a public dataset or dashboard that shows network health indicators, such as spikes in revert transactions, failed contracts, or unexpected account creation. Ongoing security audits, protocol upgrades, and community education will be essential as Ethereum develops to foresee potential threats and safeguard the decentralised ecosystem.

# **REFERENCES:**

* Lawal, Y. L. (2020). *Anomaly Detection in Ethereum Transactions Using Network Science Analytics* (Master's thesis, University of Cincinnati).
* Chen, Huashan, Marcus Pendleton, Laurent Njilla, and Shouhuai Xu. "A survey on ethereum systems security: Vulnerabilities, attacks, and defenses." *ACM Computing Surveys (CSUR)* 53, no. 3 (2020): 1-43.
* Chen, T., Li, X., Wang, Y., Chen, J., Li, Z., Luo, X., Au, M.H. and Zhang, X., 2017. An adaptive gas cost mechanism for ethereum to defend against under-priced dos attacks. In *Information Security Practice and Experience: 13th International Conference, ISPEC 2017, Melbourne, VIC, Australia, December 13–15, 2017, Proceedings 13* (pp. 3-24). Springer International Publishing.
* Gu, Zhexian, and Omar Dib. "Enhancing fraud detection in the Ethereum blockchain using ensemble learning." *PeerJ Computer Science* 11 (2025): e2716.
* Zhao, Lin, Sourav Sen Gupta, Arijit Khan, and Robby Luo. "Temporal analysis of the entire ethereum blockchain network." In *Proceedings of the Web Conference 2021*, pp. 2258-2269. 2021.
* Poursafaei, Farimah, Ghaith Bany Hamad, and Zeljko Zilic. "Detecting malicious Ethereum entities via application of machine learning classification." In *2020 2nd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS)*, pp. 120-127. IEEE, 2020.
* Li, J., Gu, C., Wei, F., & Chen, X. (2020). A survey on blockchain anomaly detection using data mining techniques. In *Blockchain and Trustworthy Systems: First International Conference, BlockSys 2019, Guangzhou, China, December 7–8, 2019, Proceedings 1* (pp. 491-504). Springer Singapore.
* Wu, Jiajing, Qi Yuan, Dan Lin, Wei You, Weili Chen, Chuan Chen, and Zibin Zheng. "Who are the phishers? phishing scam detection on ethereum via network embedding." *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 52, no. 2 (2020): 1156-1166.
* Podgorelec, Blaž, Muhamed Turkanović, and Sašo Karakatič. "A machine learning-based method for automated blockchain transaction signing including personalized anomaly detection." *Sensors* 20, no. 1 (2019): 147.
* Su, Liya, Xinyue Shen, Xiangyu Du, Xiaojing Liao, XiaoFeng Wang, Luyi Xing, and Baoxu Liu. "Evil under the sun: Understanding and discovering attacks on ethereum decentralized applications." In *30th USENIX Security Symposium (USENIX Security 21)*, pp. 1307-1324. 2021.
* Gu, Zhuoming, Dan Lin, and Jiajing Wu. "On-chain analysis-based detection of abnormal transaction amount on cryptocurrency exchanges." *Physica A: Statistical Mechanics and its Applications* 604 (2022): 127799.
* Scicchitano, Francesco, Angelica Liguori, Massimo Guarascio, Ettore Ritacco, and Giuseppe Manco. "A deep learning approach for detecting security attacks on blockchain." In *CEUR Workshop Proceedings*, vol. 2597, pp. 212-222. CEUR-WS, 2020.
* Aziz, Rabia Musheer, Rajul Mahto, Kartik Goel, Aryan Das, Pavan Kumar, and Akash Saxena. "Modified genetic algorithm with deep learning for fraud transactions of ethereum smart contract." *Applied Sciences* 13, no. 2 (2023): 697.
* Elbaghdadi, Abdelaziz, Soufiane Mezroui, and Ahmed El Oualkadi. "K-nearest neighbors algorithm (knn): an approach to detect illicit transaction in the bitcoin network." *Integration Challenges for Analytics, Business Intelligence, and Data Mining*. IGI Global Scientific Publishing, 2021. 161-178.
* Agarwal, Rachit, Shikhar Barve, and Sandeep Kumar Shukla. "Detecting malicious accounts in permissionless blockchains using temporal graph properties." *Applied Network Science* 6 (2021): 1-30.
* Ostapowicz, M. and Żbikowski, K., 2019. Detecting fraudulent accounts on blockchain: A supervised approach. In *Web Information Systems Engineering–WISE 2019: 20th International Conference, Hong Kong, China, January 19–22, 2020, Proceedings 20* (pp. 18-31). Springer International Publishing.
* Bhowmik, Madhuparna, Tulasi Sai Siri Chandana, and Bhawana Rudra. "Comparative study of machine learning algorithms for fraud detection in blockchain." In *2021 5th international conference on computing methodologies and communication (ICCMC)*, pp. 539-541. IEEE, 2021.