

ANALYSIS OF SUSPICIOUS ACTIVITY IN ETHEREUM

A Project Report Submitted
in Partial Fulfillment of the Requirements
for the Degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE AND ENGINEERING

by

PUNATI LAKSHMI BHAVYA
(Roll No. CS21B1036)



ಭಾರತೀಯ ಮಾಹಿತಿ ತಂತ್ರಜ್ಞಾನ ಸಂಸ್ಥೆ ರಾಯಚೂರು
भारतीय सूचना प्रौद्योगिकी संस्थान रायचूर
Indian Institute of Information Technology Raichur

To

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF INFORMATION TECHNOLOGY RAICHUR-584135, INDIA

APRIL 2025

DECLARATION

I, **PUNATI LAKSHMI BHAVYA** (Roll No: **CS21B1036**), hereby declare that, this report entitled **"Analysis of suspicious activity in ethereum"** 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

[PUNATI LAKSHMI BHAVYA](#)

May 2025

CERTIFICATE

This is to certify that the work contained in this project report entitled **“Analysis of suspicious activity in ethereum”** submitted by **PUNATI LAKSHMI BHAVYA (Roll No: CS21B1036)** 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

May 2025

(Dr. Priodyuti Pradhan)

Project Supervisor

ABSTRACT

In this report, we present a graph-based anomaly detection framework for analyzing Ethereum transaction data. The approach begins by computing standard and weighted degree metrics — specifically, in-degree, out-degree, weighted in-degree, and weighted out-degree — for each node in the Ethereum transaction graph. Based on these metrics, we isolate and classify nodes exhibiting irregular interaction patterns, such as having non-zero degrees with corresponding zero weighted degrees. Four distinct cases are identified, and nodes are scored based on the severity and nature of their discrepancies, with special consideration given to smart contract accounts. Nodes with the highest anomaly scores are flagged as highly anomalous. Subsequently, the transactions involving these anomalous nodes are extracted and visualized to reveal potential patterns or threats. This method effectively highlights behavioral inconsistencies that may be indicative of malicious activity or network misuse in Ethereum, providing a valuable tool for blockchain security and forensic analysis.

Contents

List of Figures

XXIX

1 Chapter-1 Ethereum

Introduction	7
1.1 A deep dive into the Ethereum	9
1.2 Overview of Ethereum Accounts	9
1.3 Ethereum Account Type Comparison	11
1.4 Transaction Inclusion in Ethereum	12

2 Chapter-2 Anomaly

Anomaly	
2.1 Importance of Detecting Anomalies in Ethereum	13
2.2 Types of Anomalies in Ethereum	15
2.2.1 Smart Contract Honeypots	15
2.2.2 Abnormal Contract Creation Patterns	16
2.2.3 Flash Loan Pattern Exploits	18
2.3 Anomaly Detection	20

3 Chapter-3 Methodology

Methodology

3.1 Preprocessing	23
3.2 Anomaly Detection Methodology one	25
3.2.1 Results	36
3.3 Anomaly Detection Methodology Two	42
3.3.1 Results	44

4 Chapter-4

4.1 Conclusion	48
4.2 Future work	48
4.3 References	49

Chapter-1

Introduction

Blockchain technologies, particularly Ethereum, have gained significant adoption due to their decentralized and trustless nature. However, this openness also makes them susceptible to various malicious activities, including scams, phishing attacks, and exploitation of smart contracts. Detecting anomalous behavior in such a vast and complex ecosystem is crucial for ensuring security and maintaining trust. In this report, we propose a graph-based anomaly detection framework to uncover unusual patterns within Ethereum transaction data.

Our approach models the Ethereum transaction network as a directed graph where each node represents an address, and edges represent transactions between them. To capture both the structural and behavioral aspects of node activity, we compute the in-degree and out-degree (number of incoming and outgoing transactions) as well as the weighted in-degree and out-degree (aggregated transaction values). By analyzing discrepancies between these metrics, we aim to detect nodes that exhibit atypical transaction behaviors.

We construct multiple intermediate files to filter and analyze specific cases: nodes with out-degree only, in-degree only, weighted in-degree only, and weighted out-degree only. These are then sorted based on degree values to identify highly active participants. Contract metadata is integrated to differentiate between smart contracts and externally owned accounts (EOAs), as smart contracts

often exhibit distinct patterns.

A critical step in the analysis involves identifying mismatches between degree and weighted degree values. Specifically, nodes are flagged where in-degree or out-degree is greater than zero, but the corresponding weighted degree is zero — suggesting suspicious behavior such as null-value transactions or potential obfuscation. Anomaly scores are assigned based on these inconsistencies, with additional weight given to smart contract involvement. Nodes with the highest scores are considered highly anomalous and are further investigated by extracting their related transactions.

The final step visualizes these high-risk nodes and their interactions in graph form, allowing intuitive identification of unusual patterns or clusters. This process not only highlights potentially malicious entities but also provides a framework that can be generalized for ongoing monitoring of blockchain ecosystems

1.1 ETHEREUM

Ethereum, introduced in 2015 by Vitalik Buterin, is a decentralized blockchain platform that extends the capabilities of cryptocurrencies like Bitcoin by enabling programmable smart contracts. Fundamentally, Ethereum uses accounts to store assets and communicate with the blockchain network. Ethereum is more adaptable than Bitcoin, which is solely a cryptocurrency, because it supports decentralised applications (dApps).

1.2 Overview of Ethereum Accounts

Types of Accounts in Ethereum

1. **Externally Owned Accounts (EOAs)** are user-managed accounts controlled by private keys, representing individuals or organizations that initiate transactions on the Ethereum network. EOAs offer complete ownership, as access to the account is solely dependent on the private key; without it, access is impossible. Unlike smart contracts, EOAs do not contain any logic or code and are primarily used to store assets, transfer Ether (ETH), or interact with smart contracts. EOAs can send and receive Ether or ERC-20 tokens, but each transaction initiated by an EOA requires the payment of a gas fee in ETH. The level of security for an EOA is directly tied to the management of its private key, and if the private key is lost or compromised, access to the account is lost as well, making it crucial to secure the private key.

2. **Smart contracts** are accounts on the Ethereum blockchain connected to self-executing code that performs predetermined activities when specific conditions are met. Unlike externally owned accounts (EOAs), smart contracts cannot initiate transactions on their own; they rely on EOAs or other smart contracts to trigger them. These contracts are written in high-level programming languages like Solidity or Vyper, and their code is deployed on the Ethereum Virtual Machine (EVM). Once deployed, smart contracts operate autonomously, executing in a deterministic manner based on the rules set in their code, without direct human intervention. Their immutability ensures reliability and consistency, as the code cannot be altered after deployment, although certain design patterns can enable future upgrades. Smart contracts are capable of storing information, performing calculations, and interacting with EOAs or other smart contracts, enabling the creation of flexible and composable systems that can call functions across multiple contracts. However, a smart contract cannot independently initiate a transaction; it requires an EOA or another smart contract to begin the process.

EOAs can be configured for multi-signature transactions, enhancing security for high-value transfers. Smart contracts enable decentralized applications (dApps) by maintaining state, emitting events, and allowing upgradability through proxy patterns. Both EOAs and Smart Contracts play crucial roles in Ethereum's ecosystem, with EOAs focused on user interactions and Smart Contracts automating complex tasks.

1.3 Ethereum Account Type Comparison

In the Ethereum blockchain, there are two primary types of accounts: Externally Owned Accounts (EOAs) and Smart Contract Accounts. EOAs, controlled by private keys, allow users to initiate transactions, send Ether (ETH), and interact with smart contracts. They are simple and typically incur lower gas costs compared to smart contract transactions.

On the other hand, Smart Contract Accounts are governed by self-executing code written in programming languages like Solidity or Vyper, and they function autonomously once deployed. These accounts do not have private keys but are controlled by the contract's code, and they require external triggers from EOAs or other smart contracts to initiate transactions.

Smart contracts are immutable after deployment, ensuring reliability, and can store data, perform calculations, and interact with other accounts. While EOAs are used for managing funds and simpler tasks, Smart Contract Accounts automate processes and support more complex interactions in decentralized applications (dApps). Both are essential for the Ethereum blockchain's operation, with EOAs representing users and Smart Contract Accounts handling programmable tasks.

While EOAs are primarily used by individuals or organizations for managing their assets and initiating basic transactions, Smart Contract Accounts empower decentralized applications (dApps) and services such as decentralized finance (DeFi), token transfers, and governance systems. The ability of smart contracts to autonomously execute predefined tasks based on conditions set in their code allows for a wide range of automated and trustless operations. This functionality enhances the Ethereum network's capacity to support a vast ecosystem of decentralized applications, making it a versatile platform for various blockchain-based innovations.

1.4 Transaction Inclusion in Ethereum

Process of adding a transaction into the ethereum chain

When a user initiates a transaction on the Ethereum network, the process begins by creating and digitally signing the transaction using the sender's private key. This signed transaction is then broadcast to the Ethereum network, entering a pool of pending transactions known as the mempool. Miners or validators (in Ethereum 2.0, under the Proof of Stake mechanism) select transactions from the mempool and include them in the next block they propose. To incentivize miners/validators for processing transactions, each transaction requires a gas fee, which also helps determine the priority of the transaction's inclusion in a block. Once the block containing the transaction is validated and added to the blockchain, the transaction is considered confirmed. Over time, as more blocks are appended to the blockchain, the transaction becomes more secure and immutable, further cementing its permanent place in the Ethereum ledger. This decentralized process ensures that all valid transactions are recorded transparently and tamper-proof.

Ethereum transactions take various forms. The simplest transaction type occurs between two Externally Owned Accounts (EOAs), often involving the transfer of Ether from one user to another. Another common transaction type occurs when an EOA interacts with a Smart Contract Account, such as calling a function within a smart contract to perform actions like transferring Ether, voting in a Decentralized Autonomous Organization (DAO), or executing trades on decentralized exchanges. In certain cases, transactions can also occur between two Smart Contract Accounts, usually triggered by a chain of function calls from an initial transaction. Smart Contract Accounts can also send transactions to EOAs, but only in response to an external trigger, as they cannot initiate transactions independently. These various transaction types highlight Ethereum's flexibility and its ability to support a broad range of decentralized applications and use cases.

Chapter-2

2.Anomaly

Anomalies are data patterns or behaviors that significantly deviate from what is expected or considered normal. In the context of networked systems like Ethereum, anomalies often represent rare or unusual activity that may signal potential issues such as fraud, exploitation, network misuse, or other forms of malicious behavior. Unlike traditional centralized systems, blockchain networks operate without a central authority, making anomaly detection a vital component in maintaining the security and integrity of the ecosystem.

In blockchain environments, anomalies can manifest in many forms — unusual transaction volumes, irregular communication patterns between nodes, interactions with inactive or previously dormant accounts, or atypical smart contract executions. These deviations are often early indicators of events such as phishing attacks, reentrancy exploits, front-running schemes, contract self-destruction patterns, or wash trading in decentralized finance (DeFi) platforms.

2.1 Importance of Detecting Anomalies in Ethereum

key reasons why discovering anomalies in Ethereum is essential:

Security and Threat Detection

Blockchain systems are constantly targeted by attackers who exploit smart contract vulnerabilities or conduct fraudulent transactions. Detecting anomalous behavior allows for early identification of potential threats, reducing the time to respond and mitigate attacks.

Network Health and Stability

Unusual transaction behavior may indicate spam or congestion attacks intended to disrupt normal network functioning. Identifying and addressing these anomalies helps preserve the operational integrity of the Ethereum network.

Fraud and Financial Risk Management

In decentralized financial ecosystems, anomalies could point to financial manipulation or unfair practices such as flash loan attacks, pump-and-dump schemes, or rug pulls. Early anomaly detection enables users and platforms to avoid or investigate suspicious activity before substantial loss occurs.

Transparency and Accountability

One of the core tenets of blockchain technology is transparency. Anomaly detection supports this by providing clear insights into abnormal behaviors, empowering auditors, regulators, and developers to hold malicious actors accountable.

Smart Contract Monitoring

Contracts once deployed are immutable, but their execution patterns can be monitored. Anomalies in smart contract calls — such as mass execution, sudden value zero transactions, or new addresses interacting with high frequency — can indicate attempts to exploit contracts or test attack strategies.

2.2 Types of Anomalies in Ethereum

2.2.1 Smart Contract Honeypots

Smart Contract Honeypots are an intriguing defensive mechanism in the blockchain ecosystem. These contracts are designed to look vulnerable and exploitable at first glance, which tempts malicious users to interact with them. However, these vulnerabilities are intentionally placed, and the contract is actually crafted to deceive and trap the attacker. Honeypots operate by embedding hidden logic or fallback functions that only activate under certain conditions, effectively making the exploit fail — sometimes costing the attacker money in gas fees or locking their assets.

Concept and Operation

The core concept of a honeypot is to lure attackers into believing that the contract has a weakness they can exploit, such as the ability to withdraw funds without proper authentication. Once the attacker attempts to exploit this seemingly open flaw, the contract's hidden mechanisms come into play. These mechanisms might involve subtle logic tricks, gas-related limitations, or misleading function names. The result is that while the contract appears to permit malicious actions, it will always prevent them from succeeding. Thus, instead of the attacker gaining an advantage, they fall into a trap — often losing their own funds in the process.

Impact and Ethical Implications

The impact of smart contract honeypots is twofold, offering both benefits and raising ethical questions. On the positive side, they serve as a strong defensive measure, discouraging malicious behavior and providing insights into attacker methods. Honeypots can be valuable educational tools and a way to test the effectiveness of detection systems in the real world. However, their use also enters a gray area ethically. When white-hat hackers or unaware users interact with a honeypot under the belief that they are performing valid tests, they may unintentionally trigger the trap. This can discourage legitimate security research and foster distrust within the developer community. Moreover, if honeypots are not transparently labeled or disclosed, they can be misused to intentionally deceive users for financial gain, crossing the line from defense to exploitation.

2.2.2 Abnormal Contract Creation Patterns

In the decentralized world of blockchain, smart contracts are fundamental to enabling trustless interactions between users and applications. Normally, these contracts are deployed in a transparent, decentralized manner by developers of legitimate decentralized applications (DApps). However, a growing threat involves attackers deploying a large number of near-identical contracts — a pattern known as **Abnormal Contract Creation**. These contracts are designed with malicious intent, often to deceive users, launch phishing schemes, or imitate popular decentralized finance (DeFi) platforms. This technique undermines the safety of blockchain users by creating a web of seemingly legitimate but harmful contracts.

How It Occurs?

The process of abnormal contract creation typically involves the use of **automated scripts or bots** that mass-deploy smart contract clones on the blockchain. These cloned contracts often differ only in trivial parameters such as the address or small configuration changes, making them hard to distinguish from the original. In many cases, these contracts are purposely made to **mimic well-known DApps**, including their interfaces and interaction methods, so that users feel confident interacting with them. The strategy hinges on volume and deception — attackers hope that among thousands of clones, some will attract unsuspecting users.

Risks

The primary risk of abnormal contract creation is that **users may unknowingly interact with malicious contracts**, believing them to be trustworthy versions of legitimate DApps. This can result in users approving transactions that allow the malicious contracts to access or steal their funds. One common scenario involves users connecting their wallets to **fake DeFi interfaces**, which appear genuine but are designed to trick them into granting permissions or sending assets. Once the contract is approved, it may drain tokens or execute unauthorized transactions. Because these fake contracts are deployed en masse, identifying and blacklisting all of them becomes a difficult task, especially for novice users.

Abnormal Contract Creation Patterns represent a significant and evolving threat in the smart contract ecosystem. By mass-deploying nearly identical contracts that mimic legitimate platforms, attackers exploit users' trust and familiarity with popular DApps. This method is particularly dangerous because it doesn't rely on exploiting a technical vulnerability in the contract itself, but rather on **social engineering and visual deception**. To mitigate this risk, users must be educated on verifying contract sources, using trusted interfaces, and reviewing transaction details carefully. On a broader level, blockchain communities and developers need to invest in tools and standards that detect and flag suspicious deployment patterns, ensuring a safer and more trustworthy decentralized environment.

2.2.3 Flash Loan Pattern Exploits

Flash loans are a unique and powerful feature in decentralized finance (DeFi) that allow users to borrow large sums of digital assets without any collateral. The only condition is that the loan must be borrowed and repaid within the same transaction block. While flash loans were initially developed to support efficient arbitrage and liquidity balancing in DeFi ecosystems, they have increasingly become a tool for **sophisticated exploits**. Attackers take advantage of the atomic nature of flash loans to manipulate DeFi protocols and extract profit with little to no upfront capital — making them a major security concern.

How It Occurs?

A typical flash loan exploit follows a well-structured pattern. First, the attacker takes out a flash loan, instantly gaining access to large volumes of tokens. Using these tokens, the attacker then **manipulates a DeFi**

protocol, such as by targeting a **vulnerable price oracle** or **liquidity pool**. This manipulation is often designed to **artificially inflate or deflate asset prices**, creating a temporary imbalance. The attacker then executes a series of actions — often involving arbitrage trades or swaps — to **extract profits** from the manipulated conditions. After successfully profiting from the exploit, the attacker **repays the flash loan within the same transaction**, completing the cycle without leaving any outstanding debt. Because all operations are executed in a single transaction block, the attack happens extremely quickly, making it hard to detect or prevent in real-time.

Risks

Flash loan exploits pose **serious risks to the stability and security of DeFi platforms**. One of the immediate consequences is the **sudden loss of liquidity** from the targeted protocol, as funds are either drained or redistributed based on manipulated conditions. In several high-profile cases, attackers have managed to **steal millions of dollars in mere seconds**, leaving DeFi projects struggling to recover and maintain user trust. Beyond financial losses, these attacks **undermine the credibility of decentralized finance** by exposing vulnerabilities in core systems like oracles and automated market makers (AMMs). Repeated incidents also discourage new users and investors from participating in the DeFi ecosystem, fearing the potential for loss through complex, hard-to-detect exploits.

Flash loan pattern exploits highlight the dual-edged nature of innovation in decentralized finance. While flash loans serve a legitimate purpose and enable efficient on-chain operations, their misuse has become a **major threat vector**. The ability to perform powerful, atomic financial actions without upfront capital is appealing not only to arbitrage traders but also to malicious actors. As these attacks grow in complexity and frequency, it becomes imperative for DeFi developers to **strengthen protocol design**, implement **oracle safeguards**, and improve real-time monitoring mechanisms. Educating users and creating community-driven alerts can also help mitigate the damage. Ultimately, the goal is to preserve the utility of flash loans while defending against their abuse — ensuring a secure, reliable, and trustworthy DeFi environment.

2.3 Anomaly Detection in Ethereum Blockchain Transactions

Blockchain networks like Ethereum offer transparent transaction systems, but transparency alone does not prevent fraud. Detecting anomalies in transaction behavior is crucial for identifying potential threats, scams, or suspicious activity. This report presents two types of anomalies identified through graph-based analysis of Ethereum transactions.

Anomaly 1: Degree vs Weighted Degree Mismatch

This anomaly arises when there is a mismatch between a node's transaction frequency and the value of those transactions. It is based on comparing structural (in/out degree) and weighted (in/out value) metrics.

1.1 High In-Degree, Zero Out-Degree

Some smart contracts receive many transactions (high in-degree) but do not send any (zero out-degree). This behavior suggests blackhole contracts, such as the *Parity Wallet Freeze* in 2017, where funds were locked and could not be retrieved. EOAs showing this pattern may be savings wallets, but smart contracts without outflow are suspicious.

1.2 Degree Exists, Weighted Degree is Zero

In many cases, EOAs send zero-value transactions to smart contracts for signaling purposes. However, when such zero-value transactions occur frequently between two nodes within seconds (e.g., 5–10 transactions in 1 second), they resemble phishing signals or dust attacks. These spam-like bursts are not normal and are flagged as anomalies.

1.3 Both Degrees Present, But Weighted Values Are Zero

When both in-degree and out-degree are non-zero, but the weighted values are zero, it indicates repeated interactions with no real ether exchanged. This is usually non-productive behavior and may point to bots or deceptive activity.

Anomaly 2: EOAs Acting as Smart Contracts (Dual Behavior)

This anomaly highlights addresses that behave both as Externally Owned Accounts (EOAs) and smart contracts—something that's not technically possible under normal conditions.

2.1 Observed Behavior

Externally Owned Accounts (EOAs), which are user-controlled, and smart contracts, which are code-based, are generally considered distinct types of Ethereum addresses. However, certain addresses exhibit anomalous behavior by rapidly switching between EOA and smart contract types. These addresses are seen initiating bursts of zero-value transactions between different nodes within one second. After these rapid exchanges, these addresses suddenly receive large ether transfers, suggesting the presence of manipulation or suspicious activity. This type of behavior raises concerns about the integrity of the blockchain and the potential for fraud.

2.2 Risks

The observed behavior of addresses flipping between EOA and smart contract types introduces several risks. **Type confusion** occurs when security systems, which rely on distinguishing account types for access control, are bypassed. This allows attackers to exploit vulnerabilities in systems that rely on these distinctions. **Obfuscation** is another significant risk, as attackers can hide malicious smart contracts behind the guise of EOAs, making it difficult for users and automated systems to detect potential threats. Lastly, **trust exploits** emerge when these deceptive addresses mimic trusted EOA identities, leading to a situation where unsuspecting users might interact with fraudulent accounts, ultimately compromising their assets or exposing them to further malicious actions.

Chapter-3






METHODOLOGY

3.1 PREPROCESSING:

The dataset was downloaded from xblock pro website.This is my initial dataset.

blockNum	timestamp	transactionId	from	to	toCreate	fromIsCon	toIsCon	contraValue	gasLimit	gasPrice	gasUsed	callingFunc	isError	eip2718ty	baseFeePe	maxFeePe	maxPriorityFeePerGas
46147	1.44E+09	0x5c504ec	0xa1e438f	0x5df9b87	None	0	0	31337	21000	5E+13	21000	0x	None	None	None	None	None
46169	1.44E+09	0x19f1df2	0xbd08e0f	0x5c12a8e	None	0	0	1.99E+19	21000	9.1E+11	21000	0x	None	None	None	None	None
46170	1.44E+09	0x9e6e19f	0x63ac545	0xc93f225	None	0	0	6E+20	21000	5E+11	21000	0x	None	None	None	None	None
46194	1.44E+09	0xc9378f5	0x037dd0f	0x7e7ec15	None	0	0	1E+20	21000	1E+12	21000	0x	None	None	None	None	None
46205	1.44E+09	0x570ce1f	0x3f2f381	0x4bd5f0e	None	0	0	8.04E+20	21000	5E+11	21000	0x	None	None	None	None	None
46214	1.44E+09	0xe17d4df	0xa1e438f	0xc9d4035	None	0	0	31337	21750	5E+13	21748	0x74796d5	None	None	None	None	None
46217	1.44E+09	0x2ec3825	0xc8ebccc	0xc8ebccc	None	0	0	0	21000	6.53E+10	21000	0x	None	None	None	None	None
46219	1.44E+09	0xe891897	0xa1e438f	0x5df9b87	None	0	0	31337	21800	5E+13	21748	0x74796d5	None	None	None	None	None
46220	1.44E+09	0x35d4f3d	0xf0cf0af5	0xb608771	None	0	0	1E+20	21000	6.42E+10	21000	0x	None	None	None	None	None
46230	1.44E+09	0x417387f	0x1c68a6f	0xc8ebccc	None	0	0	5E+19	21000	7.13E+10	21000	0x	None	None	None	None	None
46235	1.44E+09	0x80f3170	0xfd2605a	0x073f70b	None	0	0	1E+16	21000	7.06E+10	21000	0x	None	None	None	None	None
46237	1.44E+09	0x3a1be27	0xbbed465	0xbfd8db4	None	0	0	4.41E+12	21000	7.05E+10	21000	0x	None	None	None	None	None
46239	1.44E+09	0xc0c1c72	0x8ce4945	0x15e34af	None	0	0	1E+20	21000	1E+12	21000	0x	None	None	None	None	None
46240	1.44E+09	0xd4ff148	0x136d4bf	0xc8ebccc	None	0	0	1E+21	21000	8.14E+10	21000	0x	None	None	None	None	None
46242	1.44E+09	0x8e2ba7c	0x4d92795	0x99c2361	None	0	0	1E+18	21000	7.47E+10	21000	0x	None	None	None	None	None

Here, after downloading it, we have splitted the dataset day wise by converting the timestamp to Unix 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.

 2016-02-14	23-04-2024 11:32	Microsoft Excel Co...	2,547 KB
 2016-02-15	23-04-2024 11:32	Microsoft Excel Co...	3,461 KB
 2016-02-16	23-04-2024 11:32	Microsoft Excel Co...	3,543 KB
 2016-02-17	23-04-2024 11:32	Microsoft Excel Co...	3,528 KB
 2016-02-18	23-04-2024 11:32	Microsoft Excel Co...	3,966 KB

To reduce the storage of these large files, I have mapped the from and to address to numerical values, like a specified number maps to a specified hash.

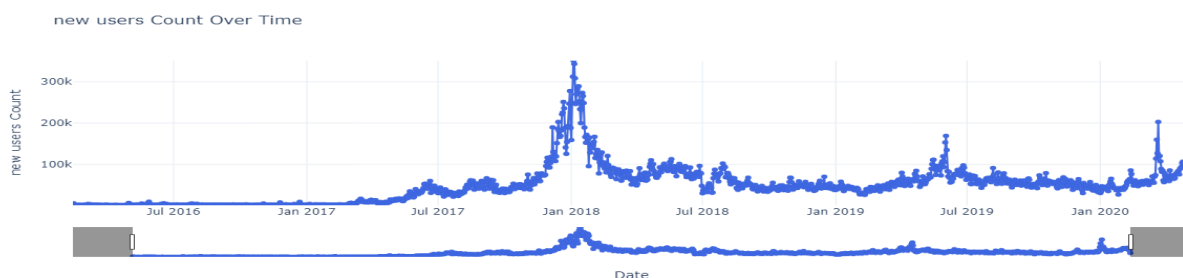
from	to		
0x39fa8c5f2793459d6622857e7d9fbb4bd91766d3	0xc083e9947cf02b8ffc7d3090ae9aea72df98fd47	1	2
0x32be343b94f860124dc4fee278fdbcdb38c102d88	0xdf190dc7190dfba737d7777a163445b7fff16133	3	4
0x2a65aca4d5fc5b5c859090a6c34d164135398226	0x819f4b08e6d3baa33ba63f660baed65d2a6eb64c	3	145
0x2910543af39aba0cd09dbb2d50200b3e800a63d2	0x9e486ad335492959c38a0740cb66c55ad30bb4f0	3	66
0x10d5bfff7879b7eb5192b3374338bb834981910a8	0xc6c764fc6c1e1211d2b4a06ef2170f660a4512fa	3	66
0x2a65aca4d5fc5b5c859090a6c34d164135398226	0x53e0551a1e31a40855bc8e086eb8db803a625bbf	3	146
0x2a65aca4d5fc5b5c859090a6c34d164135398226	0x51033f1a1a59cb6a1bf6ca2087a53bd202ac1c83		
0x120a270bbc009644e35f0bb6ab13f95b8199c4ad	0x3dc12a32a5abf477e2ec91f6218d0a96150fef99		
0x2a65aca4d5fc5b5c859090a6c34d164135398226	0xf4f2c15602b084cae84ea603f75527de19705aa1		

Also I have removed unnecessary columns from the dataset to reduce the storage further.

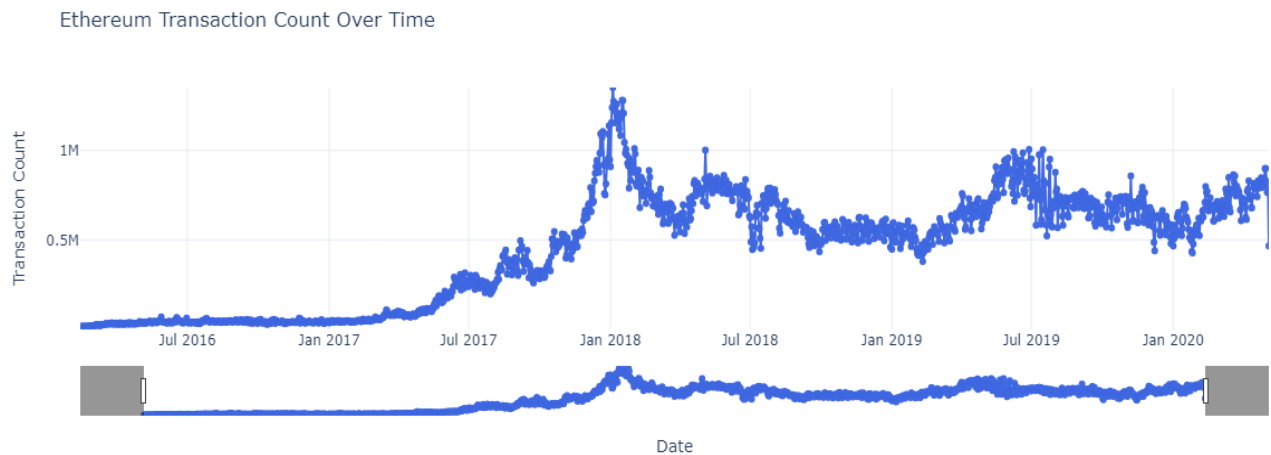
1	timestamp	date	time	from	to	fromiscontract	toiscontract	value
2	1455404053	2016-02-13	22:54:13	1	6706	0	1	10000000000000000000
3	1455404053	2016-02-13	22:54:13	2	3896	0	0	4371949800000000000
4	1455404058	2016-02-13	22:54:18	3	77	0	0	1048850240000000000
5	1455404078	2016-02-13	22:54:38	4	6707	0	0	10000000000000000000
6	1455404078	2016-02-13	22:54:38	5	78	0	0	10000000000000000000
7	1455404078	2016-02-13	22:54:38	3	70	0	0	1046558020000000000
8	1455404080	2016-02-13	22:54:40	3	55	0	0	1041761350000000000
9	1455404100	2016-02-13	22:55:00	6	658	0	0	1048847840000000000
10	1455404100	2016-02-13	22:55:00	3	58	0	0	1041460650000000000
11	1455404108	2016-02-13	22:55:08	5	6708	0	0	1000000000000000000
12	1455404157	2016-02-13	22:55:57	7	64	0	0	1150000000000000000
13	1455404157	2016-02-13	22:55:57	8	83	0	0	10067832935795451000
14	1455404157	2016-02-13	22:55:57	9	25	0	0	3019522700000000000

Now, I have extracted new users created each day by verifying if the node is already present in the network before that day or not, if not then it counts as a new node created on that specific day.

And I have plotted a graph of new users count vs day.



Similarly, I have extracted the transaction count of each day and plotted a graph for visualization purposes as shown below.



Anomaly Detection Methodology

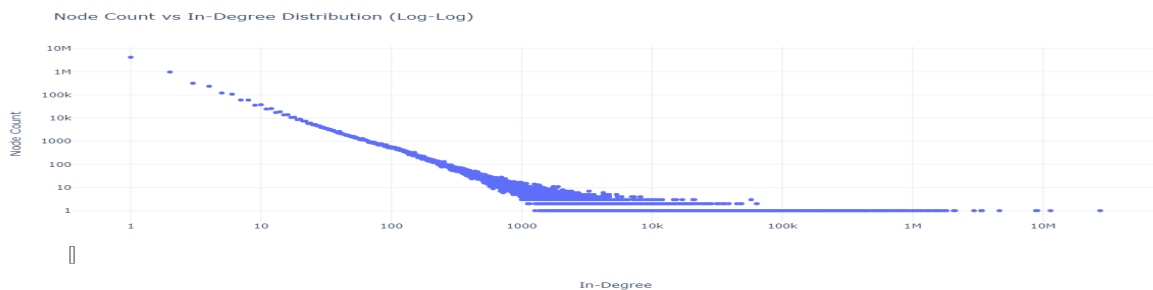
3.2 Method Used for Detecting Anomaly one

After that, I have created two files, one for degree and another for weighted degrees. Like the first file will be having node indegree outdegree and the second one will be of similar pattern but of weighted degrees.

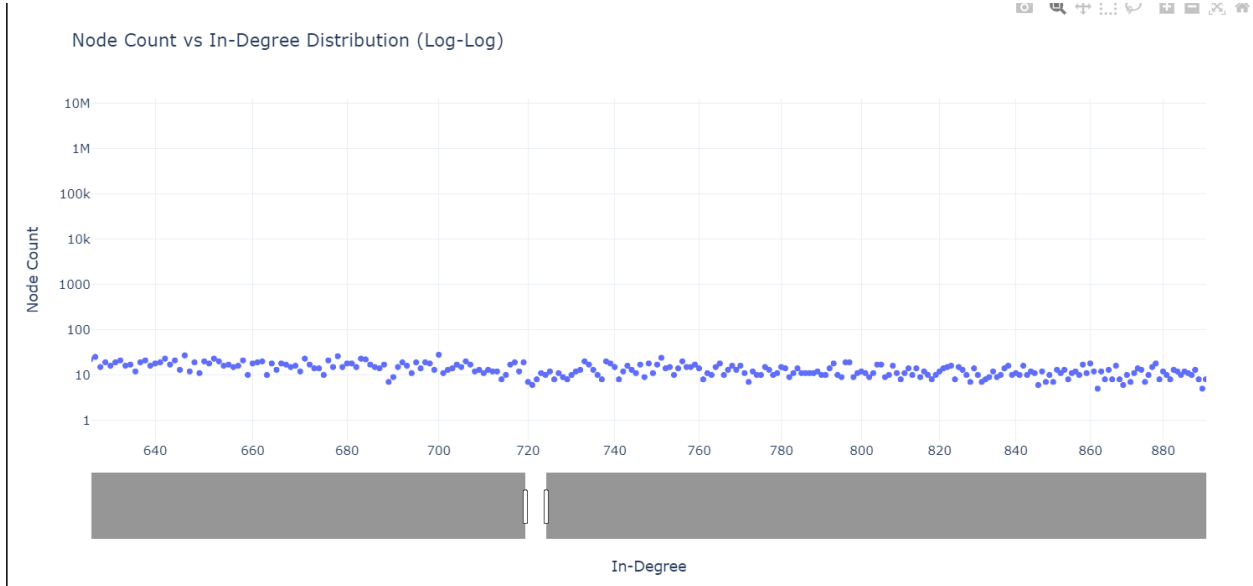
Now, extract the nodes which are having $\text{indegree} > 0$ and $\text{outdegree} = 0$ into a file. Also extract another file with $\text{indegree} = 0$ and $\text{outdegree} > 0$.

Plot the degree distribution graphs of these nodes to catch useful insights if present.

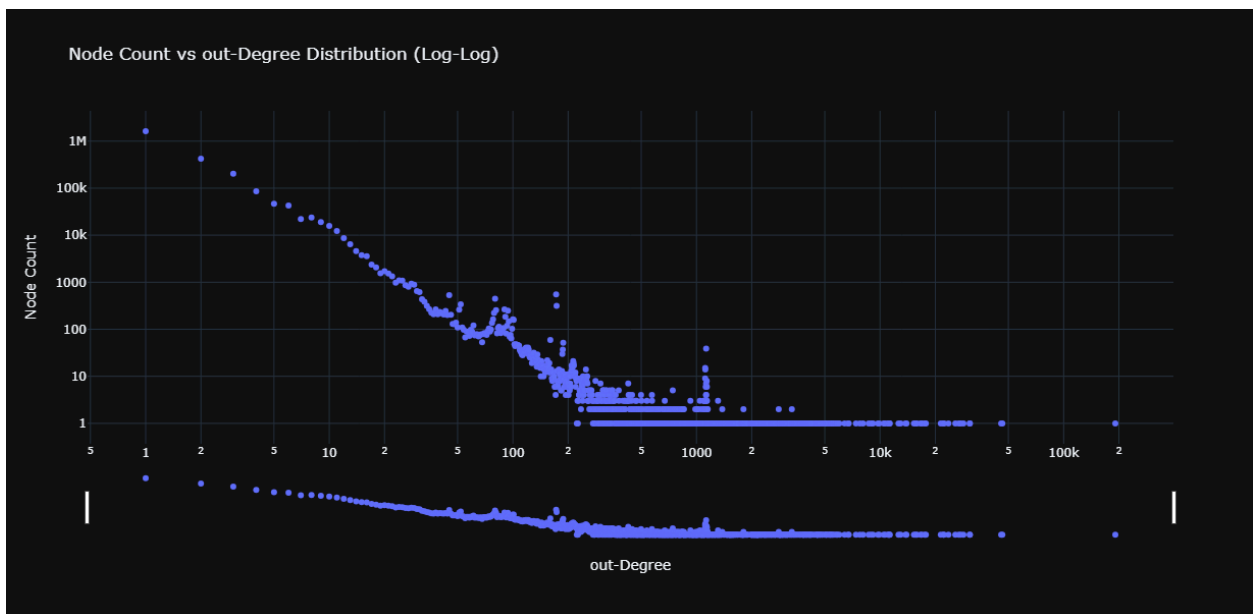
This is the plot where the indegree is not zero and outdegree is zero:



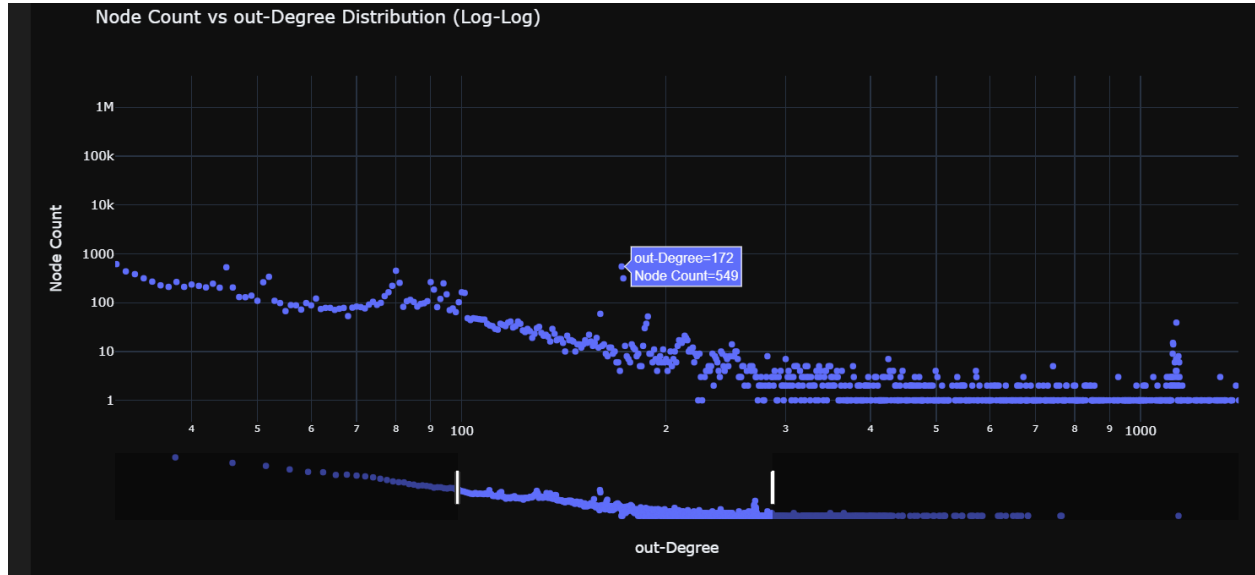
This is the small part zoomed from the above plot



This is the plot where the indegree is zero and outdegree is not zero:



This is the small part zoomed from the above plot



Also, extract nodes having weighted in degree>0 but weighted out degree=0 into a file and another file with the weighted in degree=0 but weighted out degree>0.

Now we have extracted 4 files in total, so , start comparing the respective weighted and normal degrees of the same class. Like if weighted indegree is zero and indegree is not zero and same for our degree and weighted out degree.

Here a new file is created for these kind of situations

```
node_id in_degree out_degree weighted_in_degree weighted_out_degree
295 1 2172 0 29610304644891568766976
841 1 7 0 140499067114375968
1833 0 21 0 0
1913 0 3 0 0
1920 2 837 3500000000000000000 0
2173 0 3 0 0
2888 1 2 500000000000000000 0
2951 0 3 0 0
3109 0 2 0 0
3141 2 2 2800000000000000000 0
3528 0 10 0 0
3545 4 1 1892966833026286848 0
4507 0 1 0 0
4942 1 1 1000000000000000000 0
5033 1 3 1000000000000000000 0
_*
```

Now for every file created either for degree, weighted degrees or mismatched file I have added the nodes contract status for each of them.

```
node_id in_degree out_degree weighted_in_degree weighted_out_degree contract
295 1 2172 0 29610304644891568766976 0
841 1 7 0 140499067114375968 0
1833 0 21 0 0 0
1913 0 3 0 0 0
1920 2 837 3500000000000000000 0 0
2173 0 3 0 0 0
2888 1 2 5000000000000000000 0 0
2951 0 3 0 0 0
3109 0 2 0 0 0
3141 2 2 28000000000000000000 0 0
3528 0 10 0 0 0
3545 4 1 1892966833026286848 0 0
4507 0 1 0 0 0
4942 1 1 1000000000000000000 0 0
```

Based on this, I have considered three conditions for anomaly detection:

In-degree > 0 and weighted in-degree = 0 – Nodes receive connections but show no actual incoming activity.

Out-degree > 0 and weighted out-degree = 0 – Nodes initiate connections but perform no outgoing activity.

In-degree, out-degree > 0 and both weighted in-degree, weighted out-degree = 0 – Nodes are structurally active but entirely inactive in actual interactions.

High In-Degree and High Weighted In-Degree (Normal Receiver):

This node receives many transactions with substantial ether, typical of DeFi apps, DEXs, or exchange wallets. Such high-value, high-activity behavior reflects utility and trust, not an anomaly.

High Out-Degree and High Weighted Out-Degree (Normal Sender):

The node sends many high-value transactions, often seen in EOAs, exchanges, or whales. It reflects active engagement with visible outflows, which is standard ecosystem behavior.

Both Degrees and Weighted Degrees Are High (Healthy Activity):

Nodes with many valuable incoming and outgoing transactions are likely smart contracts or staking platforms. This balanced activity is typical of operational, productive participants and is not anomalous.

Low Degree and Low Weighted Degree (Dormant or New Node):

Nodes with few low-value transactions may be new, test wallets, or casual users. Such low engagement is expected and not inherently suspicious.

Non-Zero Degree and Low Weighted Degree (Microtransactions/Gas Token Interactions):

Frequent low-value transactions may indicate micro-rewards, faucets, or gas token usage. These valid, small-scale interactions are common in testing or low-stake applications.

High Out-Degree and Zero In-Degree (EOA Dispersing Funds):

A node that only sends funds might be a cold wallet, treasury, or airdrop contract. Such fund distribution patterns are legitimate and often traceable to known sources.

High Degree and Moderate Weighted Degree (Frequent Interactions, Moderate Value):

Nodes engaging frequently but not with large values are common in social apps or games. This pattern reflects normal behavior in low-stake or high-frequency scenarios.

In-Degree > Out-Degree, and Weighted In > Weighted Out (More Receiving):

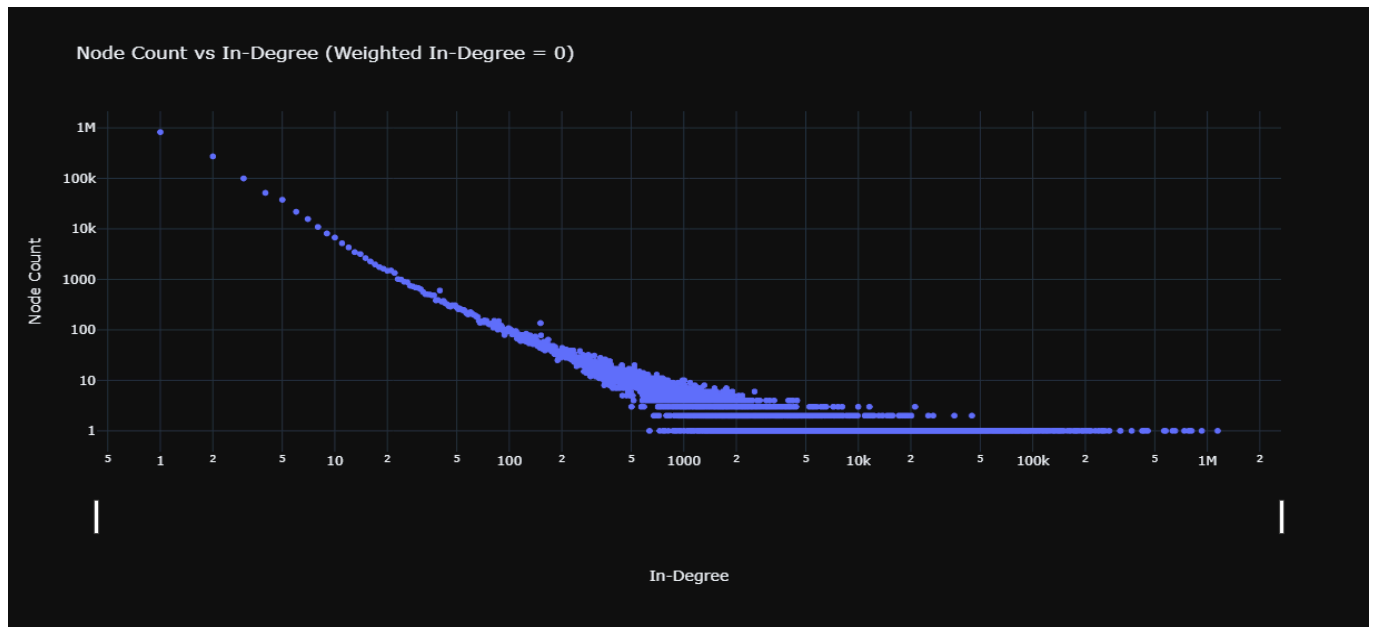
Nodes accumulating more than they send could be savings wallets or DAO treasury addresses. Receiving-focused behavior is expected in fund collection or storage cases.

Equal Degree, Unequal Weighted Degree (Asymmetric Transfer Amounts):

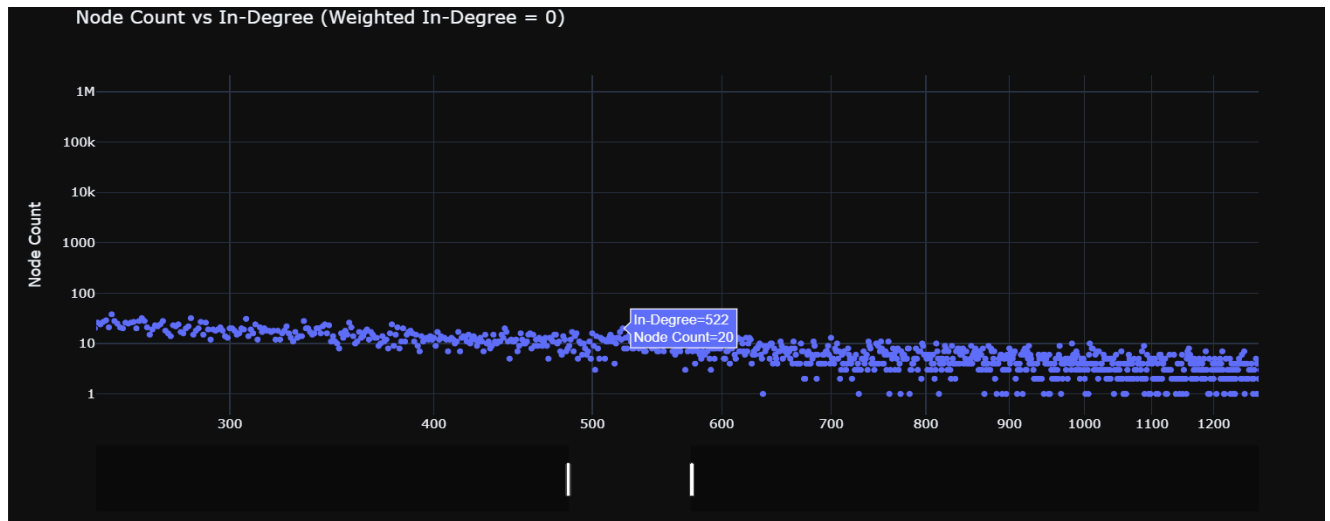
Even when send/receive counts match, differing values may reflect business or contract logic. Such asymmetry is common in applications with structured deposit and withdrawal flows.

Here , $\text{indegree} > 0$ and $\text{weighted indegree} = 0$ is our case 1.

I have plotted the degree distribution for this with log log scale

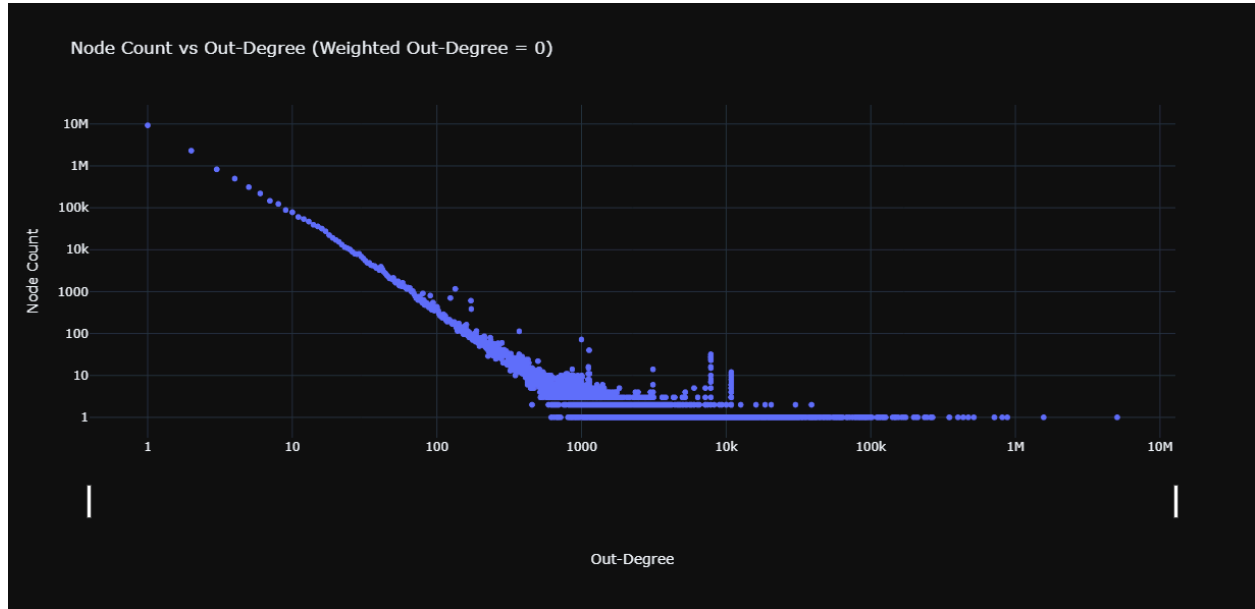


This is the small part zoomed from the above plot

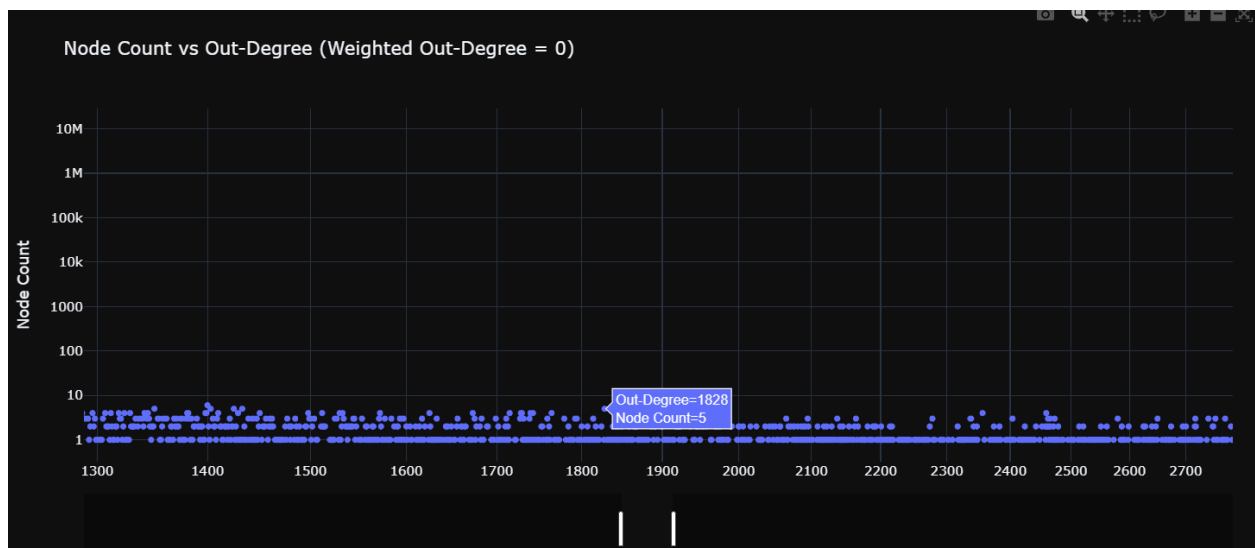


Out degree > 0 and $\text{weighted out degree} = 0$ is our case 2.

I have plotted the degree distribution for this with log log scale



This is the small part zoomed from the above plot

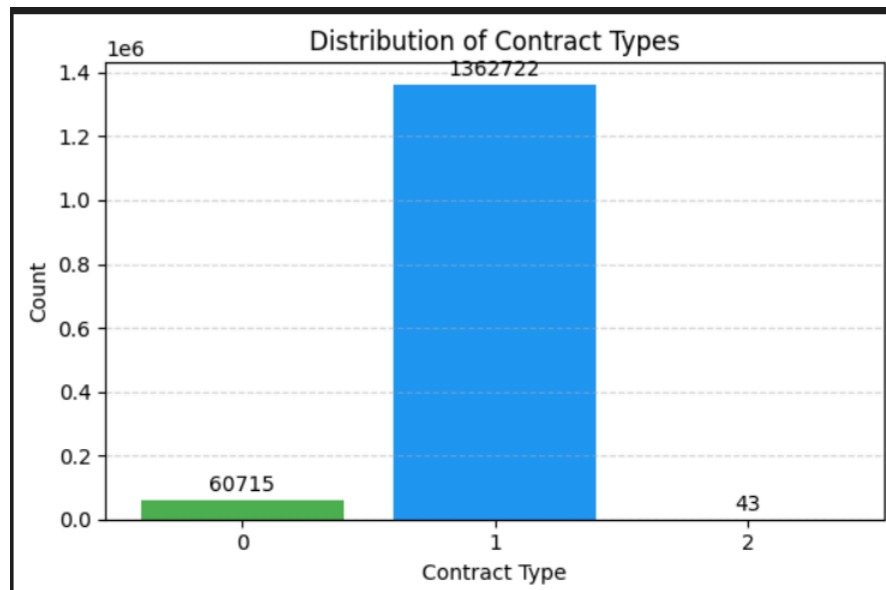


Nodes which satisfy both case 1 and case 2 are case 3.

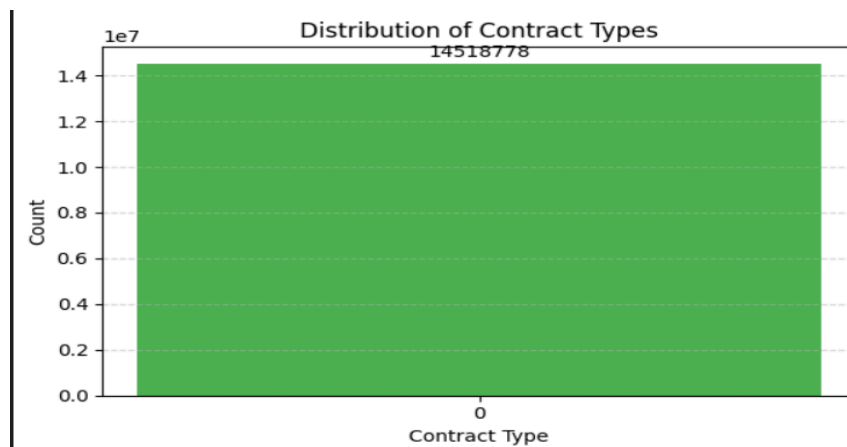
We have extracted these three cases from the mismatch-contract file.

Now, for these three cases I have plotted a bar graph between account type (smart contract,EOA, flipping account) vs node count in that particular case.

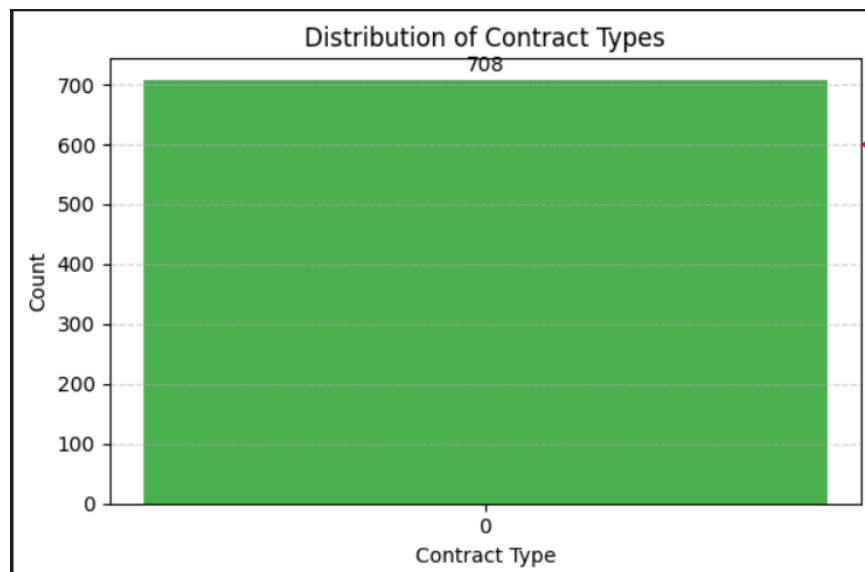
Case1 : indegree>0 and weighted indegree=0 plot is



case2: Out degree >0 and weighted out degree=0 plot is



Common nodes :indegree and out degree >0 and weighted indegree and weighted outdegree=0



As there is no threshold for identifying the anomalies, I have assigned scores for each one of the feature as shown below

- both in and out degree>0 but both weighted in and out degree =0 then the score is 2.

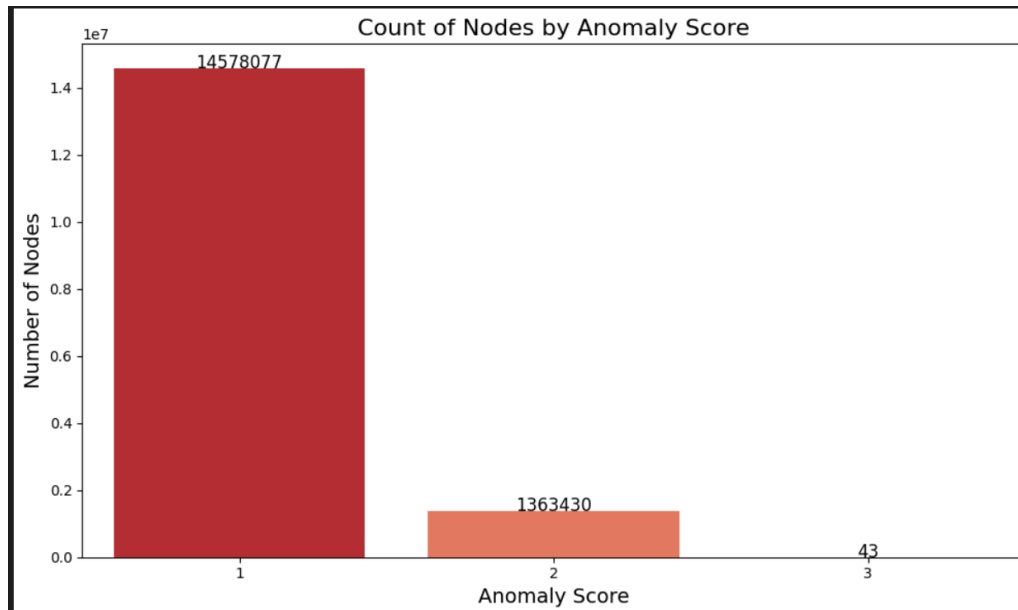
- if in degree>0 and weighted in degree =0 then the score is 1.

- if out degree>0 and weighted out degree=0 then the score is 1.

And after this i am adding the contract value to the score that has given for the three conditions that is if a node is given score as 2 then it is the smart contract the total score is the 2+1=3

Now after calculating the scores of the respective nodes, mark nodes with score of 2 and 3 as top anomalous nodes of the data.

Here is the bar graph saying that how many nodes got what scores



Now after calculating the scores of the respective nodes, mark nodes with score of 2 and 3 as top anomalous nodes of the data.

The transactions of these top anomalous nodes are extracted and plotted the transactions count vs day for these node to catch insights which might be useful

Burst Detection in Ethereum Transactions

In the context of Ethereum transaction data, a burst refers to a sudden and unusually high spike in activity — like when a specific node suddenly becomes much more active than usual.

For example, suppose Node A typically participates in around 10 to 20 transactions per day. Over time, this gradually increases — 25, 30, 40, up to 80. This pattern shows a steady rise in activity.

Now consider a 17-day transaction history like this:

[10, 12, 14, 15, 18, 20, 22, 24, 25, 30, 35, 40, 42, 45, 60, 80, 120]

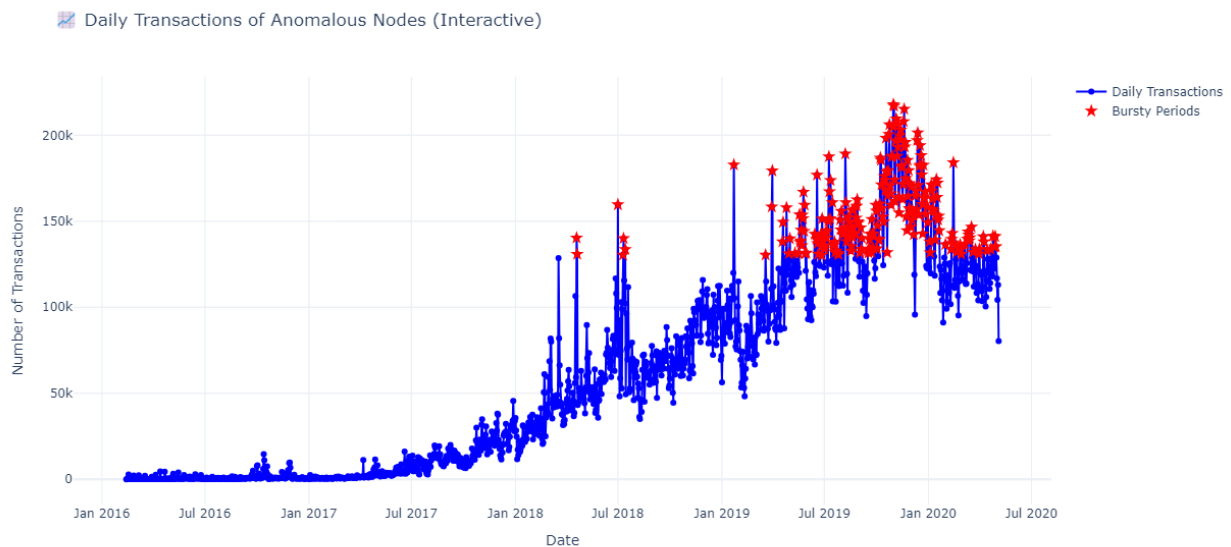
Most days fall below 80 transactions, and only the last value, 120, stands out as much higher. When we calculate the 95th percentile, it comes out to roughly 80 — meaning 95% of the days had fewer than 80

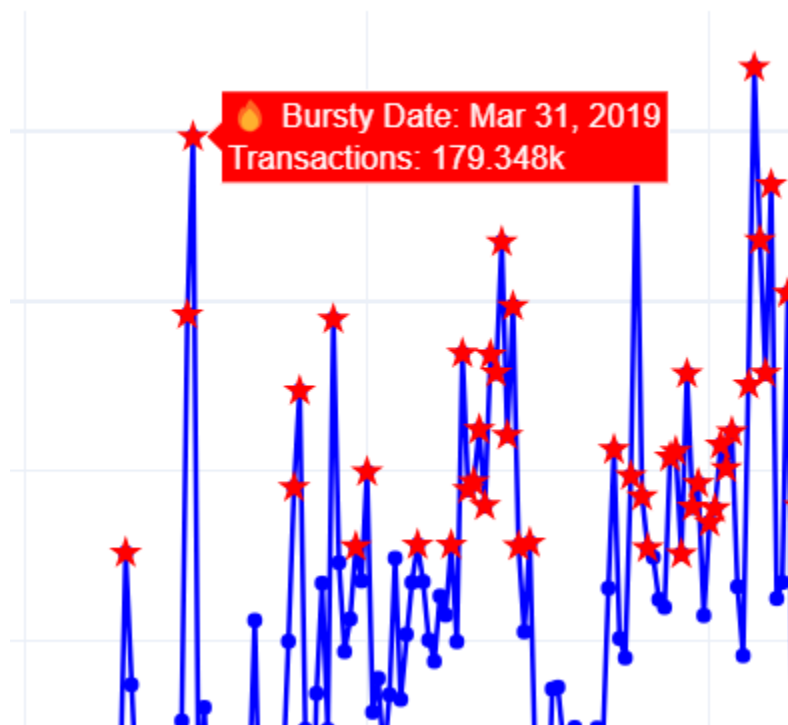
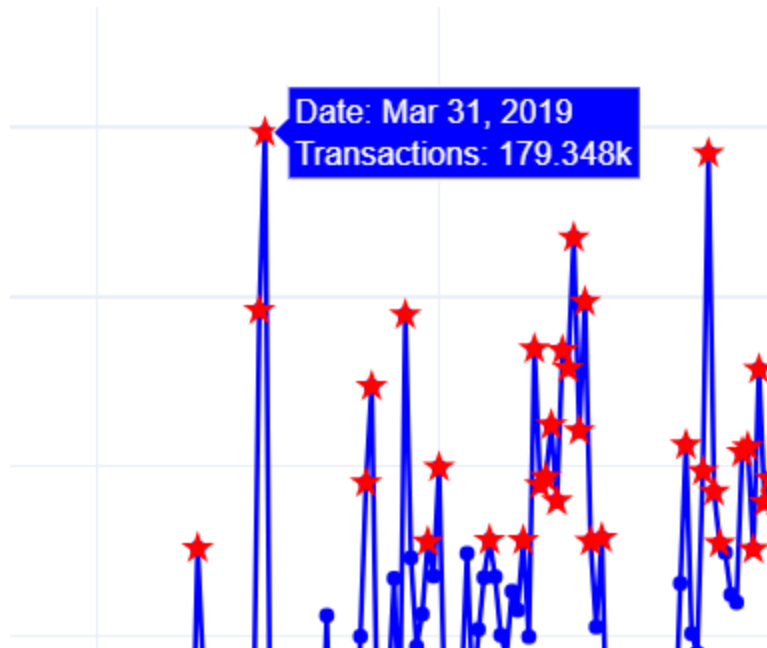
transactions.

So, any day with a value above 80 (like 120) is considered a bursty day — a rare spike that could indicate something significant, such as an event, attack, or sudden demand.

3.2.1 Results

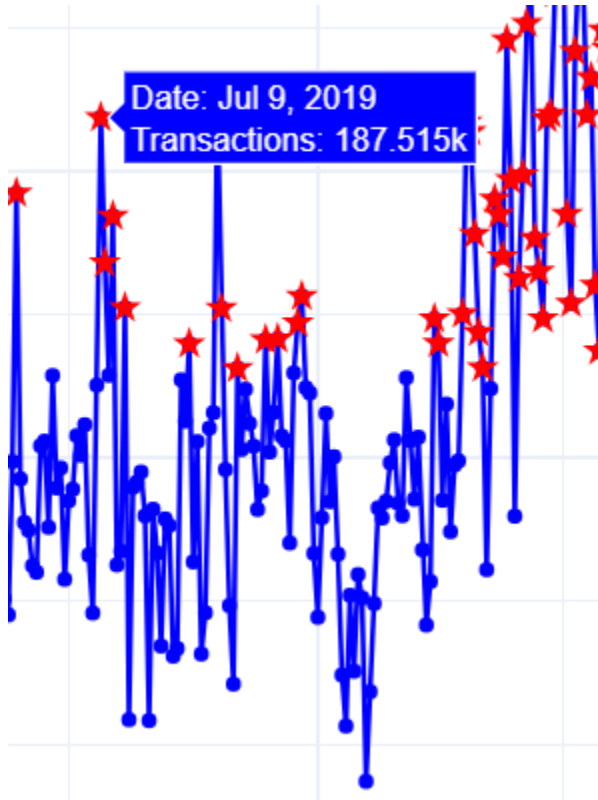
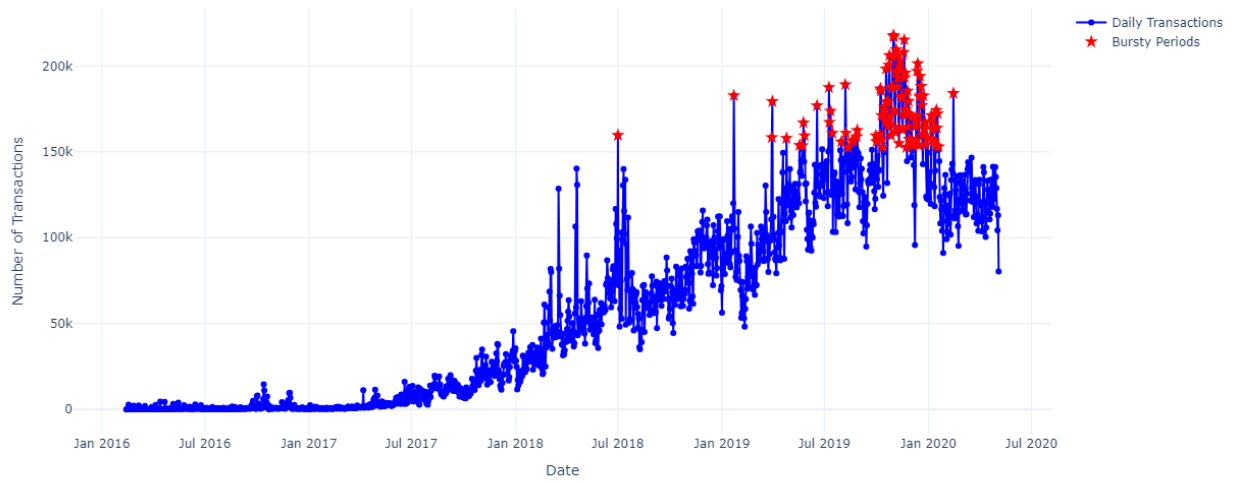
Here i have plotted the no of transactions vs date for those anomaly nodes only and burst threshold is 85 percent

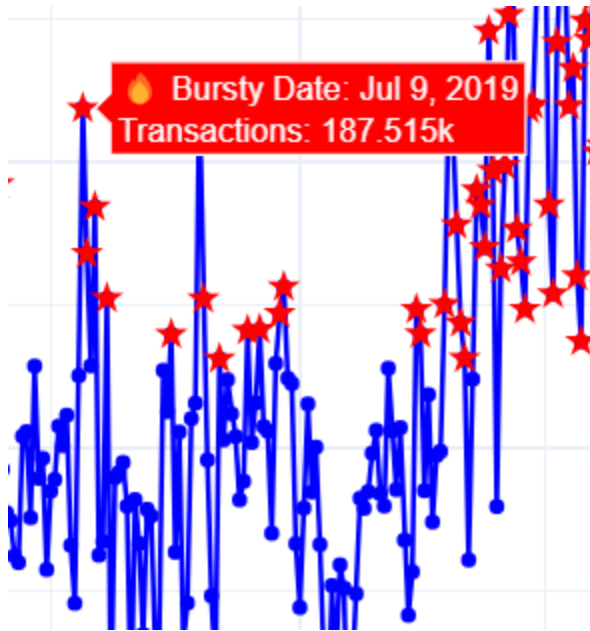




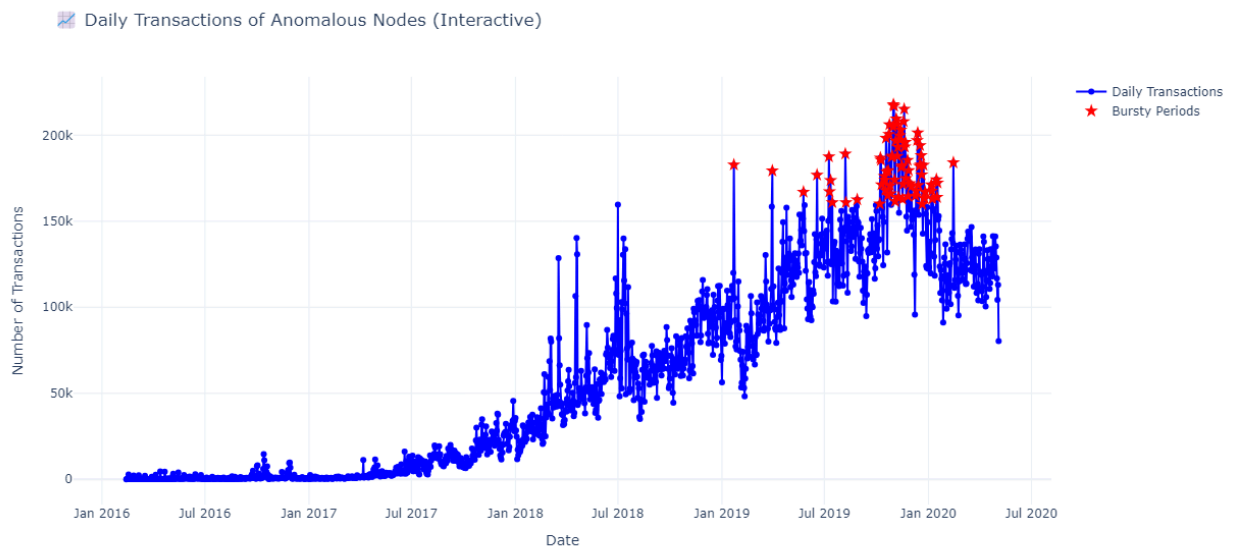
Here i have plotted the no of transactions vs date for those anomaly nodes only and burst threshold is 93 percent

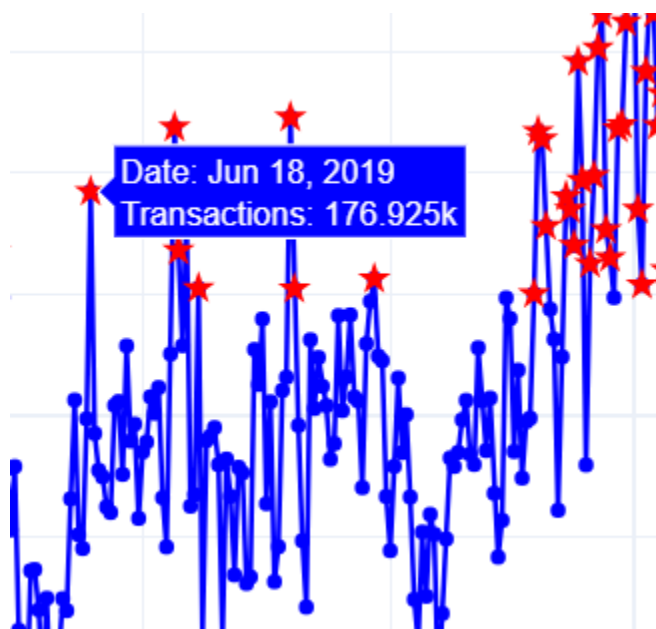
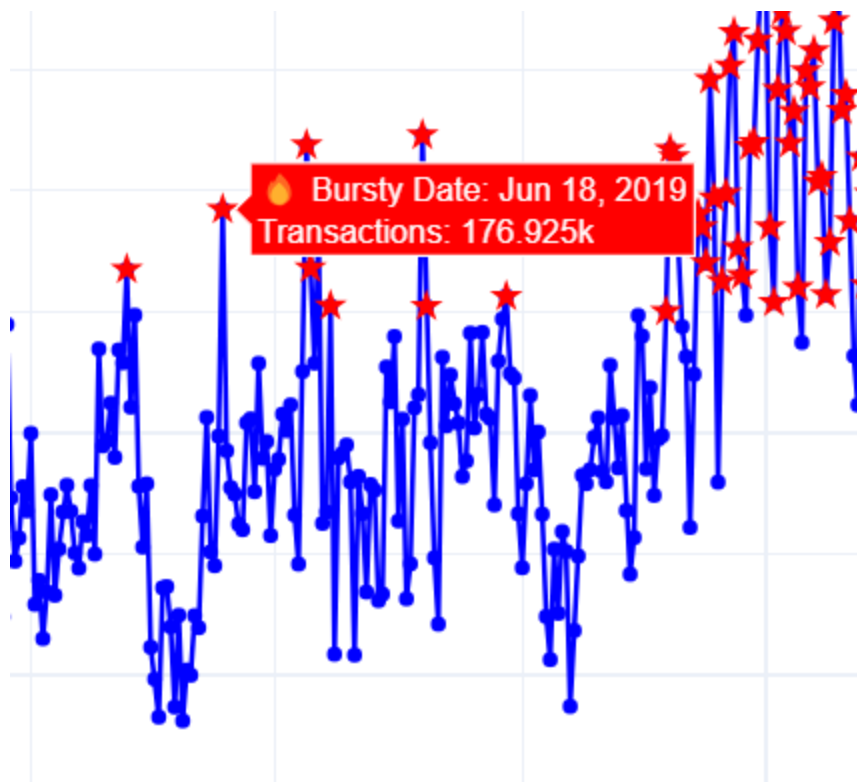
Daily Transactions of Anomalous Nodes (Interactive)





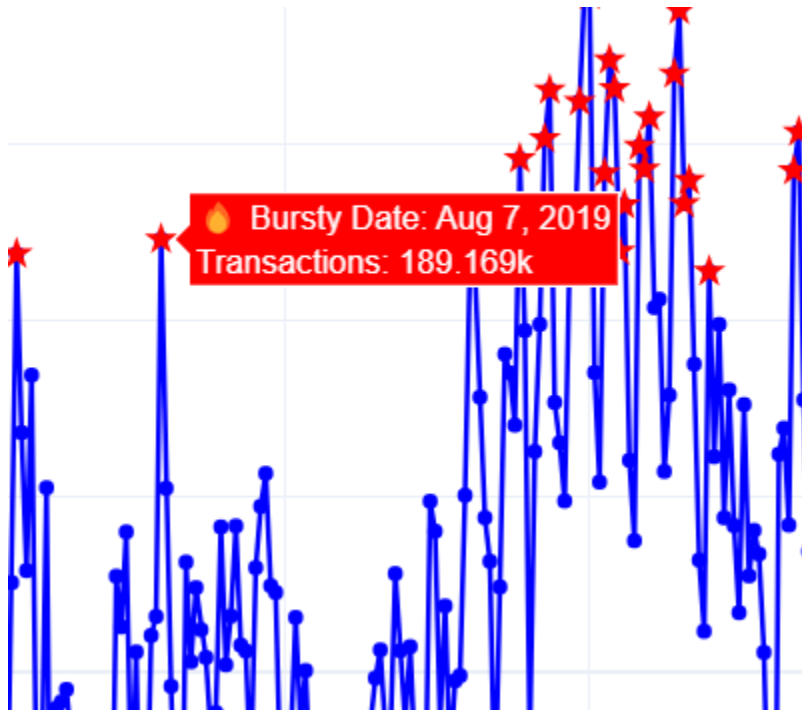
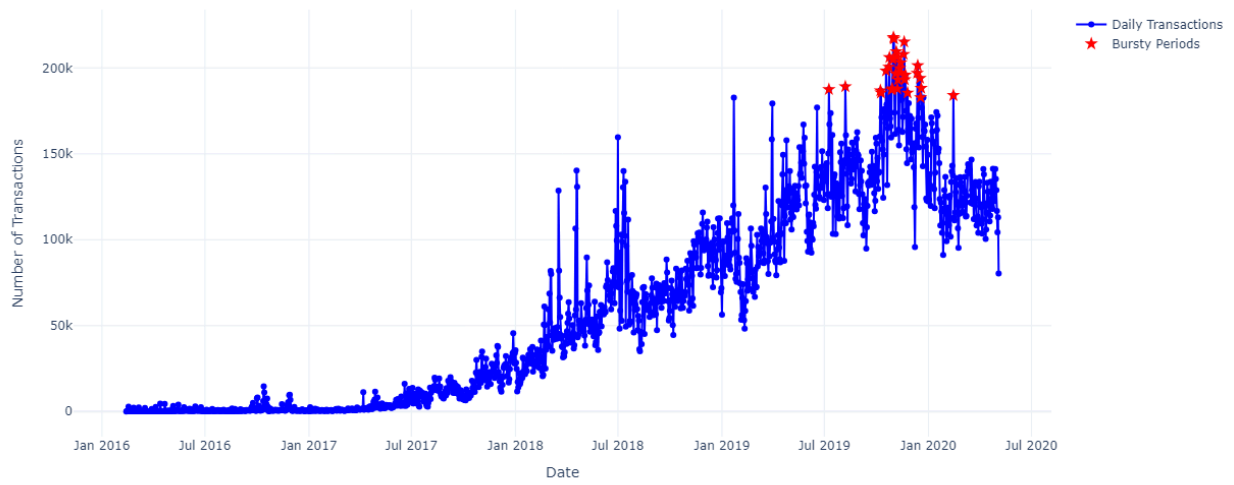
Here i have plotted the no of transactions vs date for those anomaly nodes only and burst threshold is 95 percent

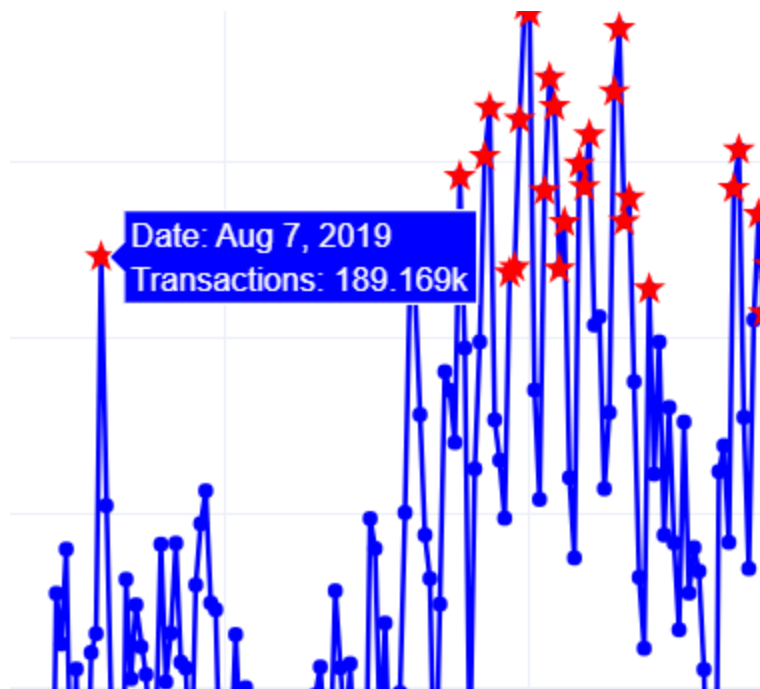




Here i have plotted the no of transactions vs date for those anomaly nodes only and burst threshold is 95 percent

Daily Transactions of Anomalous Nodes (Interactive)





3.3 Method Used for Detecting Anomaly one

Now for this anomaly, I have extracted nodes that have dual nature of account type.

Node	Contract_Type
6733	2
265660	2
274298	2
274307	2
343726	2
368581	2
406867	2
410730	2
410900	2
417618	2
426376	2

After that I have extracted each and every transaction for these nodes.

1507160965	2017-10-04	23:49:25	1138346	6733	0	1	0
1507161009	2017-10-04	23:50:09	5526324	6733	0	1	0
1507161212	2017-10-04	23:53:32	7000364	6733	0	1	0
1507161212	2017-10-04	23:53:32	5526324	6733	0	1	0
1507161351	2017-10-04	23:55:51	1138346	6733	0	1	0
1507161351	2017-10-04	23:55:51	7004236	6733	0	0	2870000000000000000
1507161629	2017-10-05	00:00:29	5526324	6733	0	1	0
1507161667	2017-10-05	00:01:07	1138346	6733	0	1	0
1507161726	2017-10-05	00:02:06	7004236	6733	0	0	2750000000000000000
1507161842	2017-10-05	00:04:02	5526324	6733	0	1	0
1507162057	2017-10-05	00:07:37	7003065	6733	0	0	3000000000000000000
1507162262	2017-10-05	00:11:02	1138346	6733	0	1	0
1507162602	2017-10-05	00:16:42	6636194	6733	0	1	0
1507162867	2017-10-05	00:21:07	1138346	6733	0	1	0

Before flipping the state of the account, I have checked whether any high value transfer is done through it or not and extracted and also noted how many times a particular node has flipped.

Burst Detection per Node:

To identify bursts in node behavior, we define a **burst** as a situation where a node flips multiple times **within a one-hour window**. If several flips happen close together with **less than or equal to one hour between them** we group them as part of the same burst.

The system goes through the **flip timestamps for each node**, and whenever it finds flips that occur within an hour of each other, it clusters them into a single burst.

For each detected burst, we store the following details:

Node ID

Burst start time

Burst end time

Number of flips in that burst

To better understand and analyze this, we visualize these bursts as **scatter plot points**, where:

The **X-axis** represents **time**

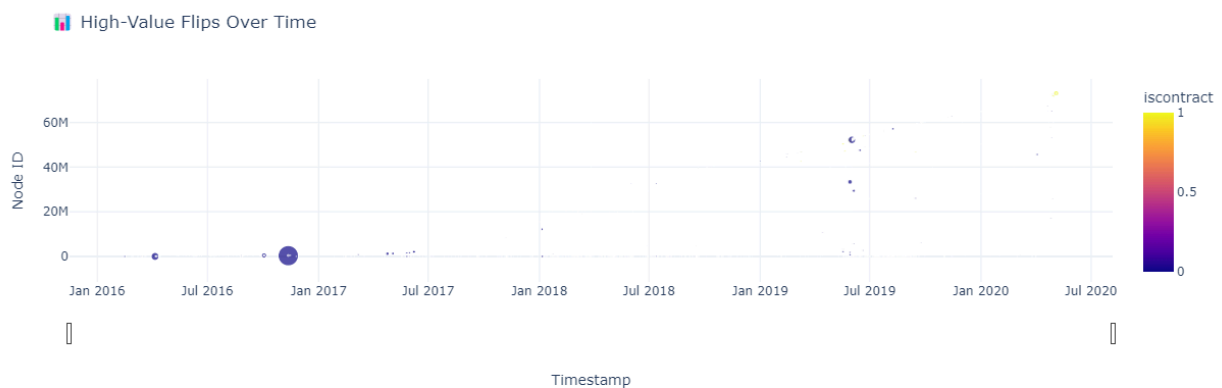
The **Y-axis** represents **node ID**

The **size of the marker** reflects how many flips happened in that burst

And when you **hover over a point**, it shows the burst's full details

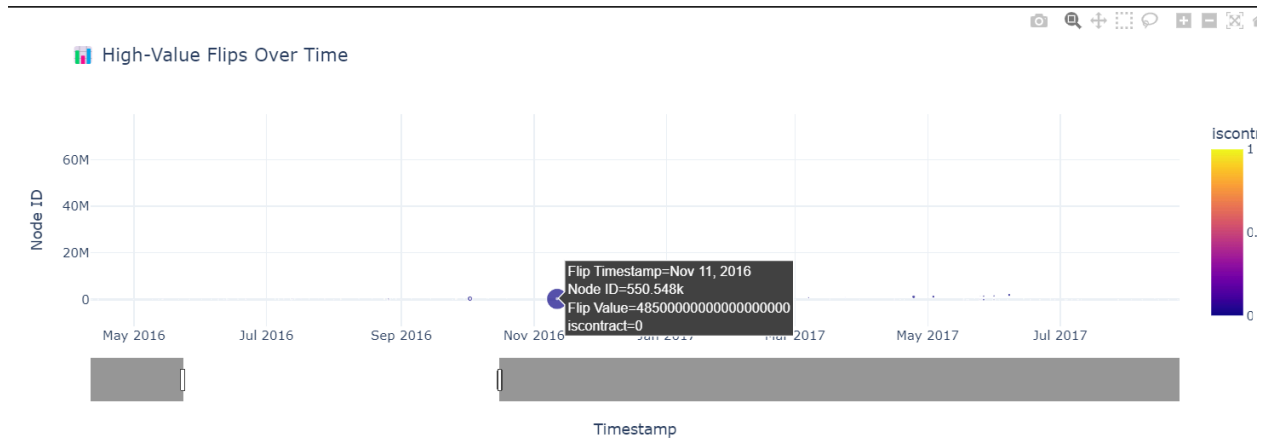
This approach helps quickly spot nodes with abnormal or sudden flip activity, making it easier to investigate potentially interesting or suspicious behavior.

3.3.1 Results

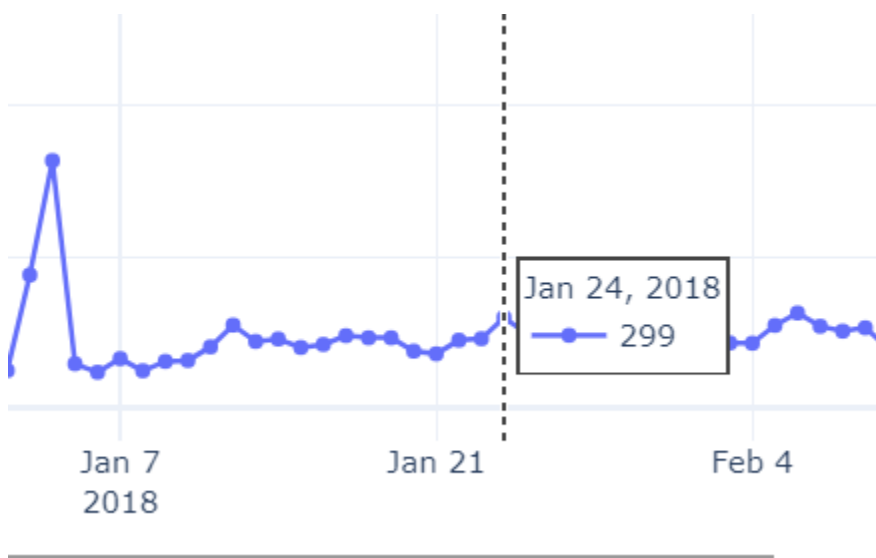
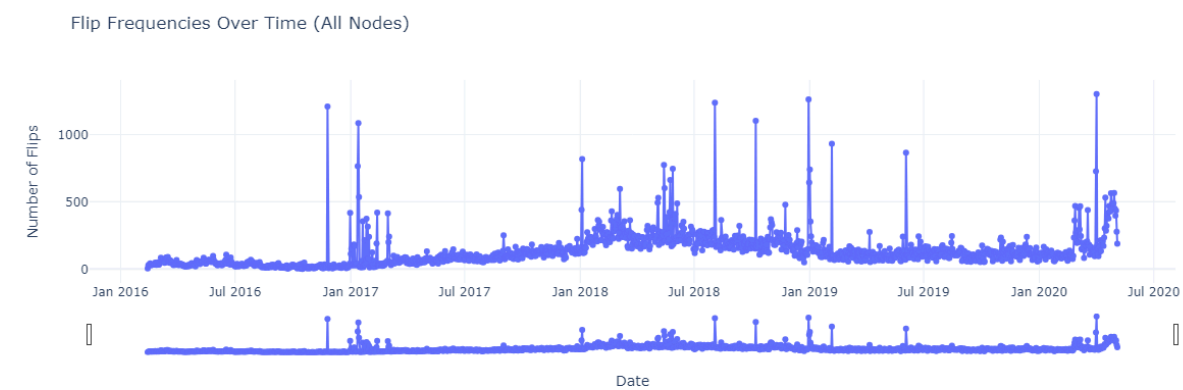


Zoomed part of the graph

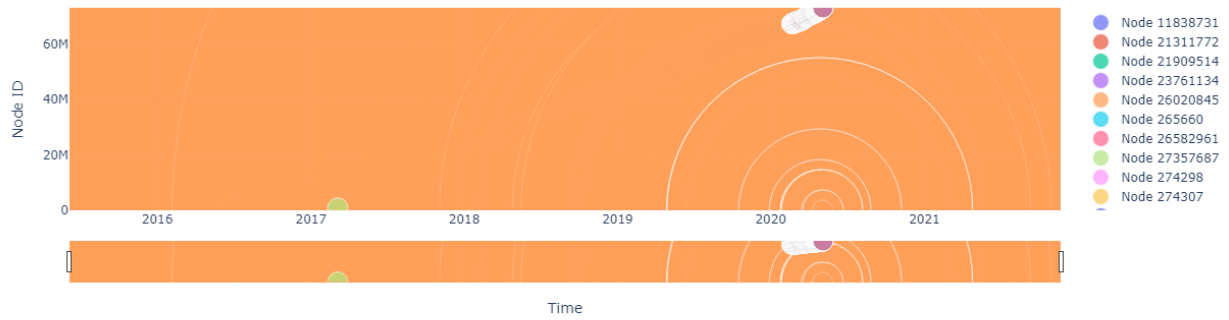
Here, plot will be of dot format but the size of dot will depend on the value transferred during flip.



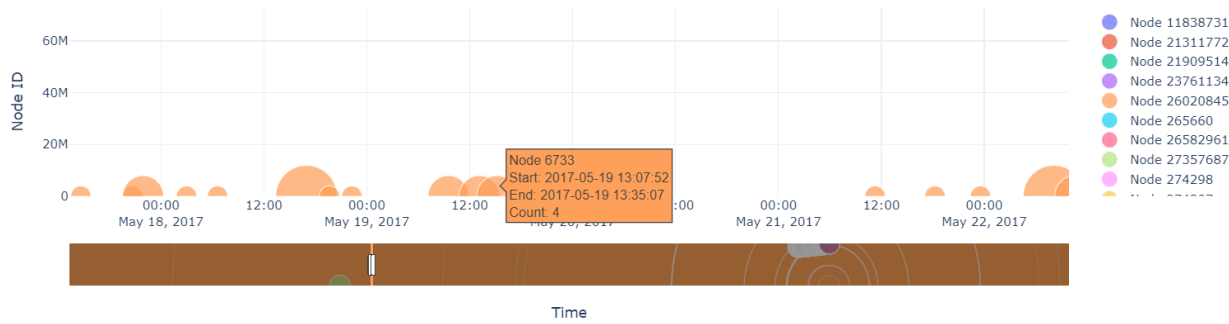
Now i have extracted the number of flips happened overall in a day and plot the graph vs day



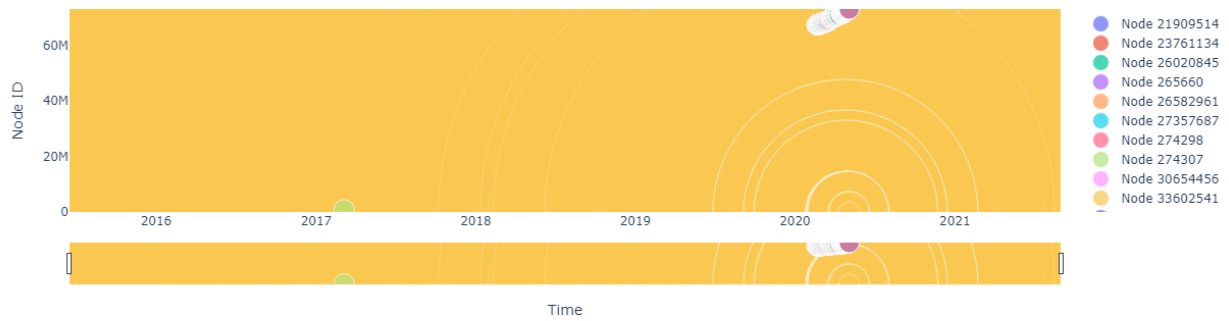
Burst Flip Events per Node

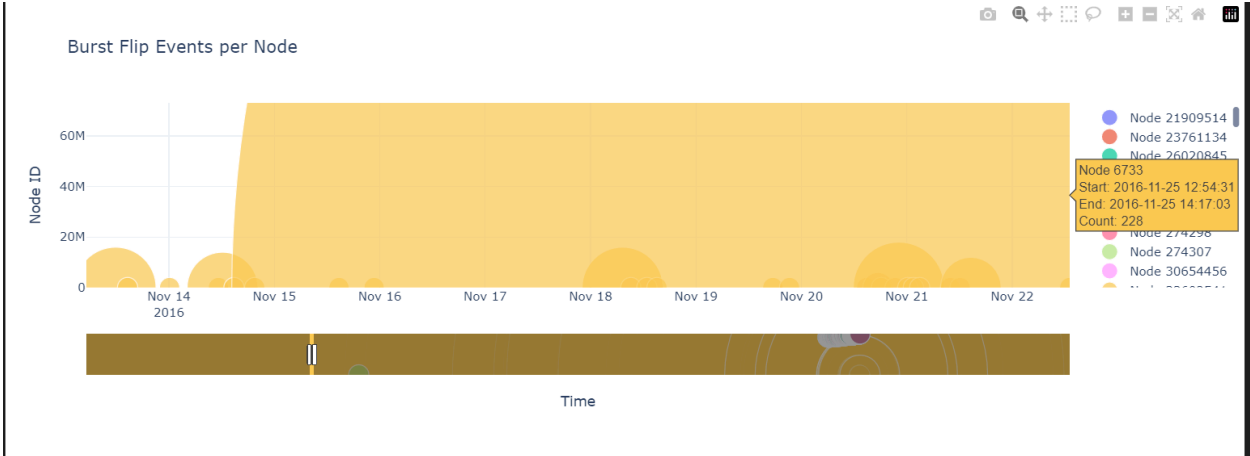


Burst Flip Events per Node



Burst Flip Events per Node





Chapter-4

4.1 Conclusion

These two anomalies uncover subtle yet potentially malicious behaviors that can significantly compromise the trust and overall functionality of the Ethereum network. By conducting a detailed analysis of transaction structures and behavioral patterns, it becomes possible to flag fraudulent or deceptive accounts at an early stage. Such insights are invaluable for identifying common threats, including phishing schemes, dust attacks, and scenarios involving locked funds. Ultimately, these findings contribute to the development of more robust fraud prevention mechanisms, enhancing the resilience and security of the Ethereum ecosystem.

4.2 Future Work

Future work in this area aims to enhance detection capabilities and deepen behavioral analysis. One promising direction is the integration of machine learning models to automate the classification of suspicious activity, enabling real-time threat detection. Another important area is the analysis of smart contract code to extract more detailed insights into malicious logic or hidden behavior patterns. Additionally, leveraging external data sources such as Etherscan and Chainalysis will be crucial for validating flagged anomalies, ensuring the accuracy and reliability of findings through corroboration with trusted third-party data.

4.3 References

- Destefanis, Giuseppe, et al. "Smart contracts vulnerabilities: a call for blockchain software engineering?." *2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE)*. IEEE, 2018.
- Yuan, Qi, et al. "Detecting phishing scams on ethereum based on transaction records." *2020 IEEE international symposium on circuits and systems (ISCAS)*. IEEE, 2020.
- Pham, Thai, and Steven Lee. "Anomaly detection in the bitcoin system-a network perspective." *arXiv preprint arXiv:1611.03942* (2016).
- Lin, Dan, et al. "T-edge: Temporal weighted multidigraph embedding for ethereum transaction network analysis." *Frontiers in Physics* 8 (2020): 204.
- Chithanuru, Vasavi, and Mangayarkarasi Ramaiah. "Proactive detection of anomalous behavior in Ethereum accounts using XAI-enabled ensemble stacking with Bayesian optimization." *PeerJ Computer Science* 11 (2025): e2630.
- <https://academy.binance.com/en/glossary/externally-owned-account-eoa>
- <https://research.checkpoint.com/2024/ethereums-create2-a-double-edged-sword-in-blockchain-security/#:~:text=The%20Technical%20Side%20of%20CREATE2&text=In%20the%20context%20of%20Ethereum,the%20new%20contract%20is%20determined>.
- Liu, Lin, et al. "Blockchain-enabled fraud discovery through abnormal smart contract detection on Ethereum." *Future Generation Computer Systems* 128 (2022): 158-166.
- Patel, Vatsal, Lei Pan, and Sutharshan Rajasegarar. "Graph deep learning based anomaly detection in ethereum blockchain network." *International conference on network and system security*. Springer, Cham, 2020.
- Liu, Lin, et al. "Blockchain-enabled fraud discovery through abnormal smart contract detection on Ethereum." *Future Generation Computer Systems* 128 (2022): 158-166.
- Wang, Wei, et al. "Contractward: Automated vulnerability detection models for ethereum smart contracts." *IEEE Transactions on Network Science and Engineering* 8.2 (2020): 1133-1144.
- Praitheeshan, Purathani, et al. "Security analysis methods on ethereum smart contract

vulnerabilities: a survey." *arXiv preprint arXiv:1908.08605* (2019).