STATISTICAL ANOMALY DETECTION IN ETHEREUM TRANSACTION GRAPHS

Kevin Wittek[1], Neslihan Wittek[2], Andrei Ioniță[3], Norbert Pohlmann[1]

The set of transactions that occurs on the public ledger of an Ethereum network in a specific time frame can be represented as a directed graph, with vertices representing addresses and an edge indicating the interaction between two addresses. While there exists preliminary research on analyzing an Ethereum network by the means of graph analysis, most existing work is focused on either the public Ethereum Mainnet or on analyzing the different semantic transaction layers using static graph analysis in order to carve out the different network properties (such as interconnectivity, degrees of centrality, etc.) needed to characterize a blockchain network. By analyzing the consortium-run bloxberg Proof-of-Authority (PoA) Ethereum network, we show that we can identify suspicious and potentially malicious behaviour of network participants by employing statistical graph analysis. We thereby show that it is possible to identify the potentially malicious exploitation of an unmetered and weakly secured blockchain network resource. In addition, we show that Temporal Network Analysis is a promising technique to identify the occurrence of anomalies in a PoA Ethereum network.

1. Introduction

Ethereum is a popular technology in the blockchain space that combines a rich shared-state model (rich referring to the state history being a core part of the system) with a quasi-Turing complete transaction-based state machine [1]. The default Ethereum protocol uses a Proof-of-Work (PoW) consensus mechanism, that shares some similarities with Bitcoin's Hashcash based PoW as proposed by Nakamoto [2] and Back [3]. However Natoli and Gramoli [4] have shown that the inherently forkable nature of PoW based blockchain protocols makes them vulnerable to e.g. double-spending attacks, especially in the context of consortium run blockchain networks. In addition, there has been a rising scepticism about blockchain technology with regards to the energy consumption and sustainability aspects of PoW based protocols, often equalizing blockchain in general with high energy consumption [5].

The Proof-of-Authority (PoA) consensus mechanism is a proposed alternative to PoW for certain blockchain network topologies and use cases. Instead of proving the investment of computing resources, it uses a set of authority nodes (often called validators) that are in charge of creating new blocks, which is called sealing in contrast to mining. Confirmations happen as soon as a certain threshold of authorities agree and sign the respective transactions. Among its advantages are the relatively short block confirmation times, due to fixed block creation times. In fact, the good distribution of authorities in the network accounts for security, especially against malicious 51% attacks. Furthermore, PoA networks are more predictable, as blocks are issued at constant time intervals. PoA is particularly effective for public-permissioned networks.

The bloxberg network [6] is a global blockchain network established by an international consortium of research organisations for scientific purposes. Its mission is to build applications in the network that promote collaboration in all research areas while remaining decentralized and robust to accommodate future requirements of the research community. bloxberg's governance is based on on-chain voting from the consortium members, while the ensuing actions are executed by the Iron Throne holder, a position that is voted for off-chain once a year. bloxberg uses a PoA consensus based on the Authority Round (AuRa) algorithm [7], that ensures availability, consistency and performance, apart from the aforementioned security properties. The bloxberg network provides a faucet application that enables members to acquire bloxberg's cryptocurrency, called bergs (which is functionally equivalent to Ethereum's Ether), to pay for the gas costs to deploy and use their smart contracts and applications. At the same time, bergs are acquired automatically while participating as a validator in the consensus. In fact, in practice, there has been a low demand for bergs from the members as soon as they collected a starting amount, by which their decentralized application (DApps/dApps) could be deployed for the first time.

The faucet application is accessible for human users as a web application and secured against fraud and abuse using Google's reCAPTCHA service in version 2 [8]. However, general operational monitoring of the faucet application, as well as random sample investigations of faucet usage in the past, have demonstrated not only suspicious and potentially malicious patterns but also exploitative faucet usage patterns. The following analyzes show that potentially exploitative activities did indeed occur, while also identifying potential heuristics that can be used for future security monitoring setups.

[1] Institute for Internet Security, Westphalian University of Applied Sciences, Neidenburger Straße 43, D-45897 Gelsenkirchen, Germany
[2] Faculty of Psychology, Department of Biopsychology, Ruhr University Bochum, Universitätsstraße 150, D-44801 Bochum, Germany
[3] Fraunhofer Institute for Applied Information Technology FIT, Fraunhofer Society for the Advancement of Applied Research, Schloss Birlinghoven 1, D-53757 Sankt Augustin, Germany

2. Model

The Ethereum transaction ledger can be modelled as a directed multigraph [9] - [11], containing edges with identity (identity properties of edges are block number and transaction value), with multiple edges being allowed but not required. The nodes of the graph represent Ethereum addresses. A transaction from address A to address B creates a directed edge from node A to node B. The graph G is, therefore, an ordered pair G = (N, E), with N being the set of nodes and E being the set of ordered pairs of nodes, i.e. edges, representing a transaction between those nodes. Looking at the ledger at a certain block number results in a certain graph. By analyzing the evolution of the graph over time, certain events can be inferred and clusters of transactions that happened in a short period of time can be identified.

Bai et al. [12] constructed three different types of graphs with the goal to uncover fundamental properties of Ethereum transaction: user-to-user graphs (UUG), contract-to-contract graphs (CCG) and user-contract graphs (UCG). UUGs describe a directed graph where the direction of the edges is dictated by the transfer of Ether between externally owned accounts (EOAs). For CCG the edges represent a creation or call action towards a smart contract, while UCG reveals how externally owned accounts use smart contracts in Ether transfers. For analysing the graph dynamics sliding windows and incremental windows are employed. It is observed that a sliding window of 180 days is suitable for analysis, as 70% of the nodes have a lifetime of below 180 days. The granularity is of about a quarter of the window size, i.e. 45 days. The incremental window expands from 180 days to 1260 days with the same granularity. As far as degree distribution is concerned, it was observed that about a quarter of the nodes (23.58%) have transactions with a single address, while 97.45% have transactions with less than ten addresses. Furthermore, patterns of interaction between node triplets are identified and counted. It was found that closed triplets, i.e. 3-node graphs where the unoriented edges describe a triangle and hence signify that all pairs of nodes are in a relationship, are negligible relative to the open triplets, the rest of the 3-node graphs.

3. Observation

For this paper, all transactions since the genesis block (which was sealed in January 2019) up until the current block number 9527285 (which was sealed in August 2020) are analyzed. In this period of time, a total number of 9527286 transactions have been recorded (see Fig. 1), which results in a transaction frequency of approximately 0.2 transactions per second.

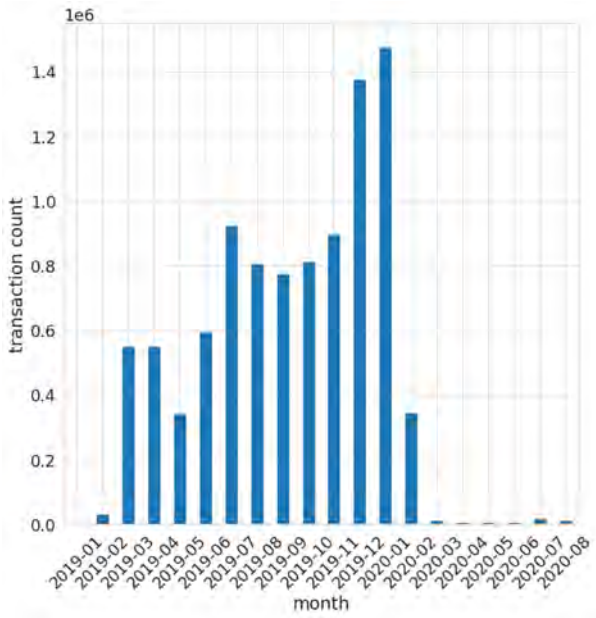

Fig. 1: Transaction count in bloxberg since the genesis block up until August 2020

The drastic decrease in monthly transactions beginning in February 2020 is due to the fact that the bloxberg network did perform the Ethereum Istanbul hard fork in this period of time which also deactivated the use of an internal monitoring Smart Contract, that was used to observe and monitor the behaviour of validators. If we filter those transactions, we get a much more accurate overview of the network traffic (see Fig. 2). When these transactions are excluded, a total number of 114256 transactions occur in the same period of time, which results in a transaction frequency of approximately 0.002 transactions per second.

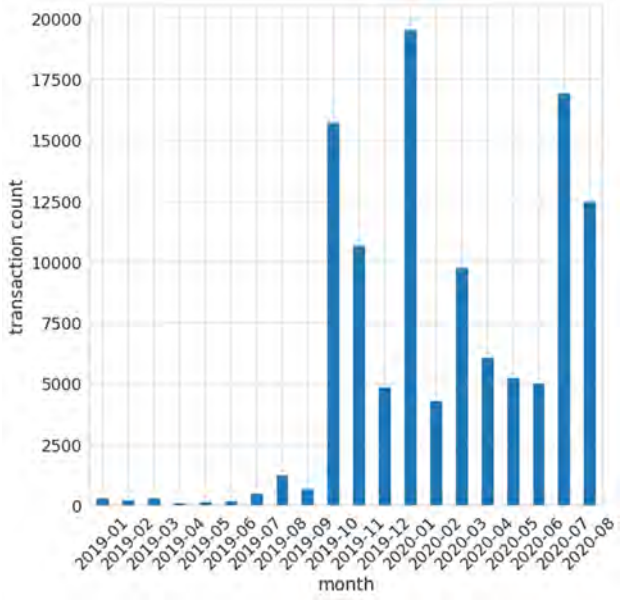

Fig. 2: Filtered transaction count in bloxberg since the genesis block up until August 2020

If we compare this to a public Ethereum network such as the Ethereum Mainnet, which possesses a transaction frequency of approximately 1.3 transactions per second in its initial 2 years beginning in 2015 [13], bloxberg can be considered a low traffic

network. It is important to consider this for further analyzes.

Of all addresses receiving funds from the faucet account (which accounts for 235 addresses in total), 38.3 % receive a single transaction (which accounts for 90 addresses in total), with 19.6 % receiving over 10 transactions (which accounts for 46 addresses in total). In general, the distribution shows the absolute peak for low transaction numbers, with high transaction numbers being seemingly suspicious (see Fig. 3).

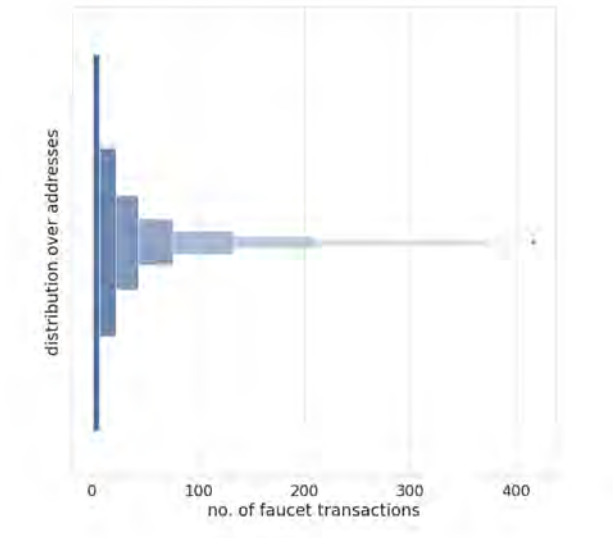

Fig. 3: Distribution of faucet transaction over addresses in bloxberg

If we focus our observations exclusively on those suspicious addresses, we can still see a peak distribution for lower transaction counts, while we also see further local maxima in the distribution for higher transaction counts (see Fig. 4).

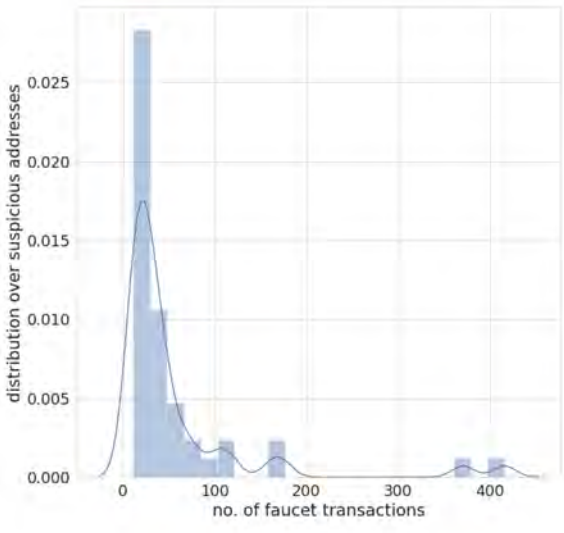

Fig. 4: Distribution of faucet transaction over addresses in bloxberg for addresses with faucet transactions count > 10

These observations with regards to faucet transaction distribution characteristics assured us in the hypothesis, that a high number of faucet transactions for a single address can be a potential heuristic for identifying suspicious addresses.

4. Results

Using the seemingly arbitrary but promising heuristic of faucet transactions counts with a cut-off value of 10 transactions, 46 suspicious addresses have been identified for which we compared the resulting transaction network graph with the transaction network graph of regular addresses. Note that transactions by validators, as well as by the faucet address, have been excluded.

We defined the connectivity of a node in the transaction graph N_{con} as the ratio between connected nodes, i.e. neighbours, N_n and the number of ingoing and outgoing edges E_{sum}:

$$N_{Con} = N_n / E_{sum}$$

We can observe a highly different connectivity distribution when comparing suspicious and regular accounts (see Fig. 5). To confirm this interpretation, a one-way between-subjects ANOVA revealed a significant difference of node connectivity between suspicious and regular addresses. ($F(1, 1281) = 120$, $p < 0.001$).

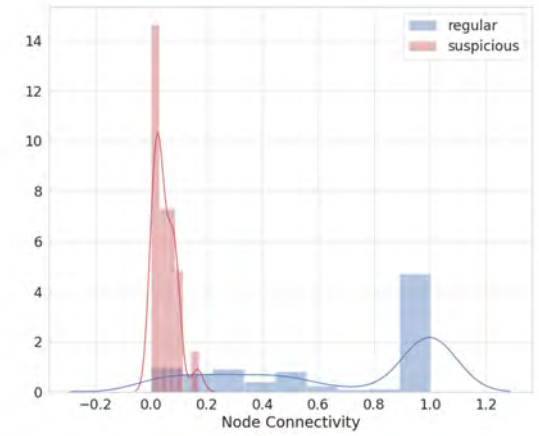

Fig 5: Comparison of node connectivity distribution between regular and suspicious addresses

Therefore we can conclude that those suspicious addresses show a transaction behaviour that differs from regular accounts, leading us to further heuristics for better classification of suspicious addresses. Consecutively, we performed a manual forensic analysis of the top 10 suspicious accounts with the highest number of faucet transactions. Simultaneously, the list of suspicious addresses has been forwarded to the bloxberg consortium, with one address in the top 10 being identified as a false positive (FP). In addition, another of the addresses seemed to indicate an FP, while for the others, 4

different patterns of exploitation could be identified: Manual, Automated, Mule & Mule Receiver.

Manual describes a seemingly manual faucet usage by a human user. Automated shows similar characteristics to Manual, but the transaction timing hints at a machine aided triggering of faucet transactions. Mule & Mule Receiver can be considered a pattern tuple, with the Mule accumulating a certain amount of bergs, which are then transferred to the Mule Receiver in a single transaction. This behaviour pattern is interesting since it shows more creativity and potential goal orientation in the exploit.

5. Temporal Network Analysis

While the previous analyzes showed promising results employing statistical techniques without considering time-based dynamics, the manual forensic analysis revealed different time-based transaction patterns which were identifiable for a human observer. It is therefore reasonable to further investigate possible techniques that allow for more precise and accurate detection of malicious activities.

Temporal network analysis has been traditionally employed in fields such as social network analysis [14], trade patterns [15] - [17] and even network neuroscience [18]. However, the time-related nature of blockchain transactions seems to make a blockchain transaction graph well suited for temporal network analysis as well.

Teneto [19] is an open-source Python package for temporal network analysis, which allows the generation of temporal network objects, the computation of various graph measures and provides specialized functions for neuroimaging use cases. Temporal network objects can be constructed from either NumPy arrays [20], pandas DataFrames [21], or built-in Python data structures, such as dictionaries or lists of edges. Centrality measures provide a way to quantify the attributes of a node in the network, such as temporal degree centrality, temporal betweenness centrality, temporal closeness centrality or a burstiness coefficient. Other measures such as burstiness, node neighbourhood, and edge properties are available as well.

Teneto has built-in plotting capabilities when used in conjunction with matplotlib [22]. A time unit can hold multiple transactions when the time axis is split into intervals. In order to eliminate empty columns, we have discretized time and assigned it the transaction index. In its simplest form, a time unit therefore corresponds to a transaction. The rows ids have been simplified too and have been assigned the addresses' index. For the bloxberg transaction set, Fig. 6 shows the plotted temporal network for the first 25 transactions. When the time unit was set to span 10 transactions, bloxberg's first 200 transactions were represented as in Fig. 7.

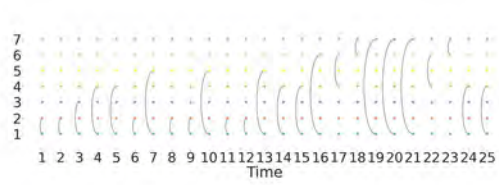

Fig. 6: Temporal network of the first 25 bloxberg transactions with time unit size equal to a single transaction

While the plotting of a temporal network might be helpful for network exploration and pattern discovery for a human analyst, future work is needed to identify classification functions for anomaly detection and malicious behaviour, which consecutively can be applied on a bigger scale via automated methods. With regards to faucet exploitation, the burstiness coefficient seems to be a promising property for further investigations.

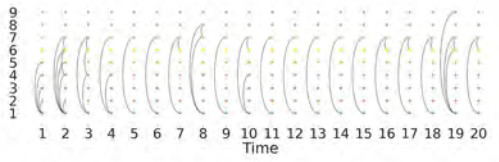

Fig. 7: Temporal network of the first 200 bloxberg transactions with time unit size of 10

6. Discussions and Mitigations

We've shown that by applying a statistical approach for analyzing the transaction graph of a PoA Ethereum network, potentially malicious and exploitative activities can be detected. Since this approach provides rather broad heuristics, it has to be supported by additional manual forensics work. However, future implementations might incorporate additional heuristics originating from temporal network analysis, which could lead to better automated classification results.

All findings have been shared with the bloxberg consortium and possible mitigations and countermeasures were discussed. Although bloxberg is not fully decentralized, since it is PoA based, the degree of decentralization makes it functionally impossible to block malicious actors from using the network, even after being identified. However, they could be restricted from using the faucet application, thereby cutting them off from collecting further funds. Future countermeasures are planned to include better monitoring and alerting of malicious faucet behaviour, with subsequent automated updates of a deny list for malicious addresses as part of the faucet application.

Acknowledgements

We like to thank all members of bloxberg for the operation and governance of the bloxberg infrastructure.
This work was partially supported by the Ministry of Economic Affairs, Innovation, Digitalisation and Energy of the State of North Rhine-Westphalia as part of the connect.emscherlippe project at the Westphalian University of Applied Sciences in Gelsenkirchen.

References

[1] G. Wood, “Ethereum: A Secure Decentralised Generalised Transaction Ledger,” EIP-150 REVISION.
[2] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” 2008.
[3] A. Back, “Hashcash - A Denial of Service Counter-Measure”, 2002.
[4] C. Natoli and V. Gramoli, “The Balance Attack or Why Forkable Blockchains are Ill-Suited for Consortium,” in 2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Jun. 2017, pp. 579–590, doi: 10.1109/DSN.2017.44.
[5] C. Schinckus, “The good, the bad and the ugly: An overview of the sustainability of blockchain technology,” Energy Research & Social Science, vol. 69, p. 101614, Nov. 2020, doi: 10.1016/j.erss.2020.101614.
[6] F. Kleinfercher, S. Vengadasalam, J. Lawton, “Bloxberg - The Trusted Research Infrastructure”, Whitepaper 1.1, Last Accessed: Aug. 28, 2020 [Online]. Available at https://bloxberg.org/wp-content/uploads/2020/02/bloxberg_whitepaper_1.1.pdf.
[7] Aura Consensus Protocol Audit, Last Accessed: Aug. 28, 2020. [Online] Available at: https://github.com/poanetwork/wiki/wiki/Aura-Consensus-Protocol-Audit.
[8] “reCAPTCHA,” reCAPTCHA. https://www.google.com/recaptcha/about/ (accessed Aug. 30, 2020).
[9] F. Harary, Graph theory, 15. print. Cambridge, Mass: Perseus Books, 2001.
[10] J. L. Gross and J. Yellen, Graph theory and its applications, 2nd ed. Boca Raton: Chapman & Hall/CRC, 2006.
[11] S. V. Pemmaraju and S. S. Skiena, Computational discrete mathematics: combinatorics and graph theory with Mathematica. Cambridge, U.K. ; New York: Cambridge University Press, 2003.
[12] Q. Bai, C. Zhang, Y. Xu, X. Chen, und X. Wang, “Evolution of Ethereum: A Temporal Graph Perspective“, arXiv:2001.05251 [cs], Jan. 2020, Zugegriffen: Aug. 25, 2020. [Online]. Verfügbar unter: http://arxiv.org/abs/2001.05251.
[13] etherscan.io, “Ethereum Daily Transactions Chart | Etherscan,” Ethereum (ETH) Blockchain Explorer. http://etherscan.io/chart/tx (accessed Aug. 30, 2020).
[14] J. Tang, M. Musolesi, C. Mascolo, and V. Latora, “Temporal distance metrics for social network analysis,” in Proceedings of the 2nd ACM workshop on Online social networks, New York, NY, USA, Aug. 2009, pp. 31–36, doi: 10.1145/1592665.1592674.
[15] B. L. Dutta, P. Ezanno, and E. Vergu, “Characteristics of the spatio-temporal network of cattle movements in France over a 5-year period,” Preventive Veterinary Medicine, vol. 117, no. 1, pp. 79–94, Nov. 2014, doi: 10.1016/j.prevetmed.2014.09.005.
[16] H. H. K. Lentz et al., “Disease Spread through Animal Movements: A Static and Temporal Network Analysis of Pig Trade in Germany,” PLOS ONE, vol. 11, no. 5, p. e0155196, May 2016, doi: 10.1371/journal.pone.0155196.
[17] X. Fan, X. Li, J. Yin, and J. Liang, “Temporal Characteristics and Spatial Homogeneity of Virtual Water Trade: A Complex Network Analysis,” Water Resour Manage, vol. 33, no. 4, pp. 1467–1480, Mar. 2019, doi: 10.1007/s11269-019-2199-2.
[18] W. H. Thompson, P. Brantefors, and P. Fransson, “From static to temporal network theory: Applications to functional brain connectivity,” Network Neuroscience, vol. 1, no. 2, pp. 69–99, Jun. 2017, doi: 10.1162/NETN_a_00011.
[19] William Hedley Thompson, granitz, Vatika Harlalka, and lcandeago, wiheto/teneto: 0.5.0. Zenodo, 2020.
[20] S. van der Walt, S. C. Colbert, and G. Varoquaux, “The NumPy Array: A Structure for Efficient Numerical Computation,” Computing in Science Engineering, vol. 13, no. 2, pp. 22–30, Mar. 2011, doi: 10.1109/MCSE.2011.37.
[21] W. McKinney, “Data Structures for Statistical Computing in Python,” Proceedings of the 9th Python in Science Conference, pp. 56–61, 2010, doi: 10.25080/Majora-92bf1922-00a.
[22] Thomas A Caswell et al., matplotlib/matplotlib: REL: v3.2.2. Zenodo, 2020.