Ethereum transaction graph analysis

Conference Paper · December 2017

DOI: 10.23919/ICITST.2017.8356459

2 authors, including:

Aspen Olmsted
Fisher College
81 PUBLICATIONS **56** CITATIONS

SEE PROFILE

Ethereum Transaction Graph Analysis

Wren Chan
Department of Computer Science and Engineering
New York University
New York, NY
wc1453@nyu.edu

Aspen Olmsted
Department of Computer Science
College of Charleston
Charleston, SC 29401
olmsteda@cofc.edu

***Abstract*—Cryptocurrency platforms such as Bitcoin and Ethereum have become more popular due to decentralized control and the promise of anonymity. Ethereum is particularly powerful due to its support for smart contracts which are implemented through Turing complete scripting languages and digital tokens that represent fungible tradable goods. It is necessary to understand whether de-anonymization is feasible to quantify the promise of anonymity. Cryptocurrencies are increasingly being used in online black markets like Silk Road and ransomware like CryptoLocker and WannaCry. In this paper, we propose a model for persisting transactions from Ethereum into a graph database, Neo4j. We propose leveraging graph compute or analytics against the transactions persisted into a graph database.**

Keywords-Ethereum; transaction graph; graph compute; graph analytics; Neo4j;

I. INTRODUCTION

Ethereum is the second largest cryptocurrency platform after Bitcoin by market capitalization, $24 billion to $53 billion (as of August 1, 2017). The currency used in Ethereum is called ether (ETH) whereas the currency used in Bitcoin is also called bitcoin (BTC). Ethereum relies on public blockchain where consensus is maintained by proof-of-work like Bitcoin. A proof-of-work system is one where miners maintain the blockchain by competing to solve computationally intensive problems (which are easy to validate). Ethereum provides the following features on top of what is offered by Bitcoin:

- Ethereum Virtual Machine (EVM) which provides a runtime environment for smart contracts.
- Smart contracts that act as stateful decentralized applications that run on EVM implementations to enforce contract instructions. A taxonomy of smart contracts was proposed by Bartoletti and Pompianu [1] to be divided into five categories: financial, notary, game, wallet, and library.
- Digital tokens that represent digital assets where the issuance is governed by smart contracts. The digital tokens can be treated as currencies like ETH and BTC and can be issued through initial coin offerings (ICOs) akin to IPOs.

Since anonymity is a key promise of cryptocurrency platform and due to the popularity of Bitcoin, much of the existing research deal with weaknesses in Bitcoin and recommendations to fix them. In this paper, we will work on applying the approaches from Bitcoin onto Ethereum. Section II will describe the related works involving the use of a graph database to persist and analyze transactions. Section III explains the motivation for applying graph analytics on Ethereum. Section IV will cover what was implemented, Section V will cover the results, and Section VI will cover the conclusions and future work.

II. RELATED WORKS

Spagnuolo et al. [6] created a modular framework, BitIodine that took transaction data and information scrapped from the web to "test several real-world use cases" including discovering connections from a known address and investigating payment for ransomware.

- The transaction data is put through a component called Clusterizer which tries "to find groups of addresses that belong to the same user" based on two heuristics based on how Bitcoin transactions take as input, the output from prior transactions (formally called unspent transaction output). Nick [8] provides a succinct write-up of several heuristics that can be applied for clustering in his thesis.
- The Grapher component creates two graphs, transaction graph and user graph from the transaction data and output from Clusterizer. The transaction graph has addresses as vertices and transactions as edges. The user graph has "users" (effectively cluster of addresses) as vertices and aggregated transactions as edges.
- The Classifier component reads the two graphs created from the Grapher, enriches it with labels that are scrapped from the web and persists the result of the resulting classifications into a database. It provides a reverse index against the transactions, addresses, and users effectively.

Fleder et al. [2] created a similar pipeline to BitIodine which appears to be due to sharing the model for transactions and users from Reid and Harrigan [7]. Unlike Spagnuolo et at [6], Fleder et al. weren't as clear about what underpins the figures provided or any source code to replicate what was done. The PageRank algorithm was mentioned as a guide to finding interesting users in the user graph due to similarities with graph constructed by search-engines to rank websites. When PageRank algorithm was applied for transactions over a single day (October 25, 2013), Fleder et al. found the transactions corresponding to the FBI's seizure of BTC from Silk Road addresses.

Haslhofer et al. [4] created a solution, GraphSense which implements what it calls an address graph. Based on the description of the address graph, it corresponds to the transaction graph that Spagnuolo et al. [6] used for BitIodine. GraphSense creates what it calls an address cluster ("user" in BitIodine) based on heuristics that aren't mentioned in the paper. Like BitIodine, GraphSense also applies tags based on contextual information scrapped from the web. GraphSense leverages GraphX and GraphFrame API of Apache Spark for graph construction and analytics.

III. Motivation

As the leading platform for execution of smart contracts, it is important to determine whether the promise of anonymity holds for Ethereum. Like many of the papers written concerning anonymity of Bitcoin, we're aiming to find potential weaknesses and recommendations for fixing them. The paper is focused on whether an attacker can de-anonymize addresses from graph analytics against transactions on the blockchain.

IV. Research and Implementation

Due to the time and disk space required for downloading the Ethereum blockchain, we've opted for loading transactions from REST API provided by Etherscan block explorer for our initial implementation (https://etherscan.io). The REST API allows for us to query for all transactions to and from an address of an account. ICOs and digital token transfer were not considered as part of the implementation.

Figure 1.Graph Model

The graph model seen in Figure 1 is used in Neo4j where we have blocks that contain many "normal" transactions which in turn have calls to multiple "internal" transactions. "Normal" transactions in this context refer to transactions that are included directly in the blockchain and therefore signed by the corresponding private key. "Internal" transactions aren't stored in the blockchain and are the results of contracts being executed in the EVM. Both "normal" and "internal" transactions, in turn, have value in ETH being transferred to and from accounts. Accounts in this model cover normal accounts that are analogous to Bitcoin addresses and contract accounts which are tied to an instance of smart contract.

The graph created for this paper started with initial addresses that are known to be associated with hacks from Etherscan (Gatecoin in May 2016 and Coindash in July 2017):

- 0x04786aada9deea2150deab7b3b8911c309f5ed90
- 0x6a164122d5cf7c840d26e829b46dcc4ed6c0ae48

From the initial addresses, we proceeded to load additional neighboring addresses that are part of the same transactions (for up to three hops away). The load logic was set to avoid loading transactions from accounts that have high in and out-degree transactions. Such accounts could be tumbler service, exchanges or gambling smart contracts which would make our results more difficult to interpret.

V. Results

Through some simple Cypher queries (Neo4j's graph query language), we could see some transactions from the Gatecoin hack ending up in addresses that are associated with cryptocurrency exchanges (Changelly, ShapeShift, Bitfinex, Bittrex, and Poloniex) based on tagging done by Etherscan as seen in Figure 2.

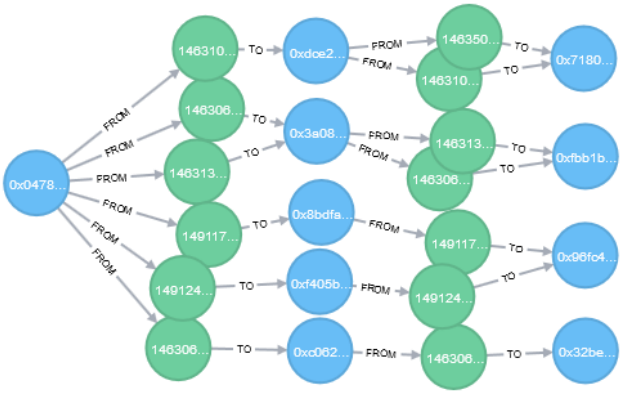

Figure 2.Transactions from Gatecoin Hack – Accounts in blue and transactions in green

VI. Conclusions / Future Work

Due to limitations of what was loaded into the graph database, our insight was limited. Ethereum doesn't rely on unspent transaction outputs to be inputs for a new transaction, unlike Bitcoin. This means that the heuristics for clustering addresses into users mentioned in Section II aren't applicable. An extension of the work performed for this paper would be to take the entire Ethereum blockchain and load it into Neo4j. We can subsequently leverage Neo4j extensions or Apache Spark to perform graph analytics such as PageRank algorithm used by Fleder et al. [2]. Aside from graph analytics, we can apply web scrapping to associate addresses with tags such as those provided by Etherscan or user handles from forum posts that mention an address.

There are several studies that rely on network layer traffic to de-anonymize transactions in Bitcoin. One study completed by Biryukov et al. [3] relied on a custom Bitcoin peer to capture the propagation of transactions to fingerprint the entry node where a transaction likely originated from. In effect, these studies function as a side-channel attack that exploited how the Bitcoin platform operated underneath the covers to be able to group transactions together and perform proactive tagging. An attempt to replicate the methodology for these studies would require a

deep understanding of the Ethereum platform and would likely yield better results compared to graph analysis.

The observation that the stolen ETH went into addresses associated with cryptocurrency exchanges suggests that it may be beneficial to link the Ethereum transaction graph with Bitcoin transaction graph. This would provide additional insight to see if ETH was converted into BTC by the hackers and whether there is an associated BTC address.

REFERENCES

[1] M. Bartoletti and L. Pompianu, "An empirical analysis of smart contracts:platforms, applications, and design patterns," *arXiv preprint arXiv:1703.06322*, 2017.

[2] M. Fleder, M. S. Kester and S. Pillai, "Bitcoin Transaction Graph Analysis," *arXiv preprint arXiv:1502.01657*, 2015.

[3] A. Biryukov, D. Dhovratovich and I. Pustogarov, "Deanonymisation of Clients in Bitcoin P2P Network," in *2014 ACM SIGSAC Conference on Computer and Communications Security*, Scottsdale, Arizona, USA, 2014.

[4] B. K. R. F. E. Haslhofer, "O Bitcoin Where Art Thou? Insight into Large-Scale Transaction Graphs," in *SEMANTICS 2016*, Leipzig, 2016.

[5] R. Boyd, "Graph Compute with Neo4j: Built-in Algorithms, Spark & Extensions," 9 March 2016. [Online]. [Accessed 10 July 2017].

[6] M. Spagnuolo, F. Maggi and S. Zanero, "BitIodine: Extracting Intelligence from the Bitcoin Network," in *FC 2014: Financial Cryptography and Data Security*, Christ Church, Barbados, 2014.

[7] F. Reid and M. Harrigan, "An Analysis of Anonymity in the Bitcoin System," in *Security and Privacy in Social Networks*, New York, Springer, 2013, pp. 197-223.

[8] J. D. Nick, "Data-Driven De-Anonymization in Bitcoin," ETH-Zürich, Zürich, 2015.