TU WIEN Informatics

A Systematic Investigation of Illicit Money Flows in the DeFi Ecosystem

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Software Engineering & Internet Computing

by

Sebastian Luzian, BSc
Registration Number 01327570

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Matteo Maffei
Assistance: Bernhard Haslhofer

Vienna, 10th November, 2022

Sebastian Luzian

Matteo Maffei

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Sebastian Luzian, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 10. November 2022

Sebastian Luzian

Danksagung

An dieser Stelle möchte ich all jenen danken, die mich bei der Fertigstellung dieser Arbeit und in meinem Studium unterstützt haben.

An erster Stelle gilt mein Dank meinem Betreuer Bernhard Haslhofer und seinen Kollegen vom Complexity Science Hub. In zahlreichen Meetings haben wir die Forschung, die für die Fertigstellung dieser Arbeit notwendig war, diskutiert und vorangetrieben. Ich habe immer hilfreiches Feedback und Unterstützung erhalten und jedes Treffen, das wir hatten, hat wesentlich zur Fertigstellung dieser Masterarbeit beigetragen.

Ein großes Dankeschön geht auch an alle, die mich während meines Studiums generell unterstützt haben. Insbesondere möchte ich mich bei meiner Familie bedanken, ohne deren Unterstützung die Fertigstellung dieser Arbeit nicht möglich gewesen wäre.

Auch möchte ich mich bei allen meinen Freunden bedanken, die mich während meiner Zeit in Wien unterstützt haben. Ohne die Unterstützung von euch allen hätte das Schreiben dieser Arbeit und auch das Studium selbst nur halb so viel Spaß gemacht.

Acknowledgements

At this point, I would like to thank all those, who supported me in the completion of my studies and this thesis.

First and foremost, my thanks go to my supervisor Bernhard Haslhofer, and his colleagues from the Complexity Science Hub. In numerous meetings, we discussed and advanced the research that was necessary to complete this thesis. I have always received helpful feedback and support and every meeting we had contributed significantly to the completion of this Master‘s thesis.

A big thank you also goes to everybody that generally supported me during my studies. In particular, I would like to thank my family without whose support completing the thesis would have never been possible.

I would also like to thank all my friends who supported me during my time in Vienna. Without the support from all of you, writing this thesis, as well as studying in general would have only been half the fun.

Kurzfassung

Mit der immer populärer werdenen Blockchaintechnologie und deren Anwendungsgebieten, traten in den letzten Jahre auch immer wieder betrügerische Aktivitäten in den Vordergrund. Durch die oft inhärent bereitgestellte Pseudo-Anonymität von Blockchains, sowie ein Mangel an Regulierung der sich darauf befindenden Applikationen werden betrügerische Tätigkeiten bedingt. Hinzu kommt die relative Neuheit dezentraler Transaktionssysteme und deren weitreichende Möglichkeiten.

Dezentrale Protokolle, Computerprogramme in der Form von Smart Contracts (SCs), zählen zu den neusten Bausteinen innovativer Finanzsysteme. Dieses neue Paradigma der Dezentralisierung, besser bekannt als Decentralized Finance (DeFi), hat sich in den letzten Jahren zu einem der am schnellsten wachsenden Segmente der Kryptobranche entwickelt. Eine Vielzahl an DeFi Plattformen welche das Investieren, Leihen und Verwalten von Vermögenswerten ermöglichen, entstanden auf verschiedenen Blockchains. Die Ethereum Blockchain beheimatet eine Vielzahl an DeFi Protokollen, stellt eine robuste Infrastruktur für SCs bereit und weist hohe Nutzungsraten auf. Die Tatsache, dass DeFi Protokolle auch Finanztransaktionen ohne Identifikationsanforderungen ermöglichen, machen sie attraktiv für den Transfer illegaler Gelder, welche aus betrügerischen Aktivitäten stammen.

Ziel dieser Arbeit ist es, durch quantitative Methoden zu erforschen wie betrügerische Addressen mit dem DeFi-Sektor in Zusammenhang stehen. In einer Websuche sammeln wir Ethereum-Adressen, die mit betrügerischen Aktivitäten in Verbindung stehen. Die gefundenen Adressen und ein Datensatz von DeFi Protokollen werden verwendet um Transaktionsdaten des Ethereum-Netzwerks anzureichern. Wir untersuchen das resultierende Netzwerk in einer Graphdatenbank und heben Transaktionsflüsse sowie deren Beziehung zu DeFi-Protokollen hervor. In einer Transaktionsanalyse vergleichen wir verschiedene betrügerische Aktivitäten und deren Verbindung zu DeFi-Protokollen.

Unsere Analysen zeigen, dass besonders Decentralized Exchanges (DEXs) betrügerisch erworbene Gelder erhalten. Uniswap (ein Open-Source-Protokoll für die Bereitstellung von Liquidität und den Handel mit Ethereum Request for Comments (ERC-20) Tokens) ist besonders stark betroffen. In einer genaueren Analyse zeigen wir daher auch wie Tokens innerhalb des Uniswap Protokolls getauscht werden, und welche ERC-20 Tokens bevorzugt von betrügerischen Adressen verwendet werden. Durch eine Untersuchung von Message Traces erforschen wir wie sich betrügerische Addressen dezentrale Protokolle zunutze machen, und wie Transaktionen innerhalb des Protokolls ablaufen.

Abstract

With the increasingly popular blockchain technology and its areas of application, illicit activities have repeatedly shown to be a problem in recent years. The lack of regulation, combined with the fact that pseudo-anonymity is intrinsically provided by most blockchains and their applications, makes them popular platforms to launder money coming from illegal activities like scams, exploits and hacks. The relative novelty of decentralized transaction systems and their far-reaching possibilities additionally add to this phenomenon.

Decentralised protocols in the form of Smart Contracts (SCs) - computer programs that run on the blockchain - are one of the many innovative building blocks of new financial systems. This new paradigm of decentralisation, better known as Decentralized Finance (DeFi), has become one of the fastest-growing segments in the crypto industry in recent years. A variety of DeFi platforms that enable investing, lending and managing assets have emerged on different blockchains. Ethereum is one of these blockchains. It provides a robust infrastructure for SCs, has a big user base and is home to a variety of protocols. The fact that DeFi protocols also enable financial transactions without identification requirements makes them attractive for the transfer of illicit funds originating from fraudulent activities.

The aim of this work is to provide a better insight into how fraudulent transaction activities on the Ethereum blockchain are related to DeFi protocols, through quantitative methods. In a web search, we collect a dataset of Ethereum addresses related to fraudulent activities. The found addresses and a dataset of DeFi protocols are used to enrich transaction data of the Ethereum network. We examine the resulting network in a graph database and highlight transaction flows and their relation to DeFi protocols. In a transaction analysis we compare different fraudulent activities and their connection to DeFi protocols.w

Our analysis shows that Decentralized Exchanges (DEXs) are common receivers of fraudulent funds. Uniswap (an open-source protocol for providing liquidity and trading Ethereum Request for Comments (ERC-20) tokens is particularly affected. In a more detailed analysis, we therefore also show how tokens are exchanged within the Uniswap protocol, and which ERC-20 tokens are preferentially used by fraudulent addresses. By examining message traces, we gain insight into how fraudulent addresses take advantage of decentralised protocols and how transactions within the protocol take place.

Contents

CHAPTER 1

Introduction

1.1 Motivation

In recent years, cryptocurrencies and blockchain-based tokens have had a revolutionary impact on the traditional banking system. Built on Distributed Ledger Technology (DLT), decentralized protocols in the form of Smart Contracts (SCs) [Sza96] have become *the* new building blocks for innovative financial systems. This new paradigm of decentralization, known as Decentralized Finance (DeFi) has become a fast-growing segment of the financial markets. DeFi platforms provide a multitude of applications ranging from lending and borrowing services for tokens or cryptocurrencies to platforms that allow speculating on price movements or earning interest in bank-like accounts. Due to cryptographic techniques, these financial systems - and the different services they provide - do not require trusted parties like banks and other intermediaries anymore and centralized regulation is no longer common [But14].

While these advancements towards decentralization certainly have great potential in the banking industry [OEC20], many possibilities for fraudulent activities have emerged in decentralized financial systems as well [HRM+21, TM20, Wro21, SS21]. As DeFi protocols allow to carry out financial transactions without identification requirements, this makes them attractive for the transfer of illicit funds coming from illegal activities [Wro21].

"As the demand for cryptocurrencies increases, it provides opportunities for criminals to hide behind the presumed privacy and anonymity. Identifying these cryptocurrency-related crimes have posed challenges for law enforcement due to the cross-border nature of transactions, the use of evasion technology to mask the identity of users, and inconsistent regulations." [KC20]

The lack of regulation, combined with the built-in properties of pseudonymity and irreversible transactions makes DeFi a popular ecosystem for illegal activities like scams,

exploits and hacks. Victims of scams often realize too late, that their transactions can never be undone, and their money is lost forever. Exploits target the lack of security in the newly formed space and try to obtain funds by force or social engineering. Hacks are often only discovered after substantial damage was done, and in most cases, there is no way to recover the stolen funds. As hiding one's identity in a decentralized space is very simple, it is easy for newcomers to construct fraudulent schemes as there are only limited ways for law enforcement to ever finding out about their operations. Luring people to send cryptocurrency in hope of good investments or quick potential profits, without the fear of being caught leads to an increasing number of entities developing new techniques and threats to extract money from unknowing victims. Without any regulatory measurements that could actually prevent these entities from doing so, this trend will likely continue. As DLT's main purpose is to enable anonymous transactions between participants, cryptocurrencies like Ethereum are also widely used to launder money coming from various types of criminal activities including theft, fraud, extortion, drug trafficking, trafficking of counterfeit goods, human trafficking and even child pornography [Eur21b].

To gain a better understanding of how illegally obtained funds move in the newly formed decentralized space, we give an insight into how DeFi applications are used by fraudulent entities in order to manage, invest and swap funds relatively anonymously. By doing so we will provide a better understanding of how different DeFi protocols on the Ethereum blockchain are connected to a range of illegal activities.

1.2 Aims

While the research in the area of cryptocurrency scams has made great contributions in the past [BLL+21, XWL+20, CGC+20, WLZZ21], there is still a lot of knowledge to gain, when decentralized financial applications are involved. As DeFi protocols have no centralized entity that could help people restore their assets [Har19], they became a popular target for scammers and hackers. These fraudulent entities not only purposefully attack DeFi protocols and their underlying technology, but they also use the very same protocols to swap, transfer and invest their stolen funds in order to make them untraceable or to gain even bigger profits. Inconsistent regulations and easily implementable evasion techniques further make it difficult for regulatory entities to take action. Decentralized financial applications are a particularly affected area of fraudulent activities, as it is one of the emerging application fields of decentralized protocols.

The goal of this thesis is to gain a better understanding of how illicitly obtained funds are sent around in the decentralized ecosystem, and how DeFi protocols on Ethereum are utilized to store and manage funds. We purposefully chose to concentrate our efforts on the Ethereum blockchain, as most decentralized protocols are built on Ethereum [CB20]. In our thesis, we therefore show how fraudulent entities on the Ethereum blockchain are connected to each other, as well as to DeFi protocols. We present a reproducible method that allows us to gain a better understanding of how illicit transactions and

DeFi protocols on the Ethereum network are related, and how and to which extent these protocols are used by fraudulent entities.

To show how fraudulent activities and their transaction behavior are connected to the decentralized financial space, and how much value is involved, we enrich aggregated transaction data of the Ethereum Network with two datasets. First, we enrich the network with metadata of fraudulent addresses. The fraudulent addresses, associated with various illegal activities, are found by web search and are grouped by their respective category of fraud. Secondly, we enrich the data with addresses of different DeFi protocols along with their types and other metadata.

By combining the aggregated transaction data with the sets of fraudulent addresses and DeFi addresses, we construct a network with all the needed metadata for us to investigate transaction activities between illicit entities and DeFi protocols.

Based on this network's topology we conduct an empirical analysis to show different network metrics, as part of a network study. We show how fraudulent entities are connected to each other and how fraudulent entities are connected to DeFi protocols. By analyzing the network´s transactions we highlight entities or groups of entities related to fraudulent transactions, and show to which protocols these fraudulent actors transfer the most value. By looking at different DeFi protocols we show the cumulative amount of fraudulent funds received over time, as well as possible revenues of different illegal activities.

Our work also investigates what protocols are of particular interest for fraudulent entities, and how different DeFi protocols are used in order to manage and trade crypto tokens for other assets or fiat currency. Common similarities regarding transaction behaviour are highlighted, and differences between different types of decentralized protocols are shown. For this case, we provide a detailed analysis of fraudulent transactions connected to the most used DeFi protocol and show how this specific protocol is used to swap the blockchain's native currency to different tokens, which can then be managed and utilized.

Following a top-down approach (first investigating the network as a whole, then going into more detail about the network's connections, and finally investigating different transaction patterns) we answer the following research questions:

RQ 1: What types of fraudulent activities are conducted on the Ethereum blockchain, and how are participating nodes connected to the DeFi sector?

RQ 2: What amounts of funds belonging to illicit activities can we identify flowing to DeFi? Are there certain protocols, that are targeted more heavily in comparison to other protocols? What type of protocols are affected most?

RQ 3: In which ways are illicitly obtained funds being transferred to DeFi protocols? Are there identifiable transaction schemes that can be observed?

1.3 Overview

In this section, we give a brief overview of our thesis. The remainder of the thesis is structured in the following way:

Chapter 2 Background: In this chapter, we introduce the Ethereum blockchain, its account models, aspects of state and transaction, SCs as well as tokens and token systems. We also describe different types of DeFi protocols and criminal activities on the Ethereum blockchain.

1. We cover the basics of the Ethereum blockchain as well as its account models.
2. We show how state and transactions work on Ethereum.
3. We take a look at SCs and provide an overview of how decentralized protocols can be built, and why they are so promising.
4. We highlight tokens and token systems so we can understand what they are used for and why decentralized protocols heavily make use of them.
5. Different types of DeFi protocols are shown and an insight into various criminal activities on the Ethereum blockchain is given.

Chapter 3 Methodology: In this chapter, we show in more detail how our data was gathered, how our network was constructed, and how the different analyses of network topology, transactions, and traces are conducted.

1. By conducting a web search, we collect illicit Ethereum addresses that were included in databases or blacklists of wallet providers as well as different on-chain analysis tools.
2. Using the 6869 found addresses, we enrich the network's aggregated transaction data to identify fraudulent addresses and their properties.
3. We additionally enrich the network's data to identify a set of 1407 DeFi addresses of 4 different types (assets, derivatives, dex, and lending) belonging to different decentralized protocols such as AAVE, Uniswap, Sushiswap, and others.
4. In a network study, we describe the constructed network and give a basic summary of the network's statistics.

Chapter 4 Results: In the results section we show how fraudulent entities are connected to the DeFi space. For this, we draw comparisons between different fraudulent activities as well as DeFi protocols in a transaction analysis. In a more detailed trace analysis, we show which trace patterns occur the most, and how illicitly obtained funds move inside the most heavily used protocol.

1. As results of the network study, we show statistics regarding the neighbours of fraudulent nodes and DeFi protocols as well as local clustering coefficients. We also show the number of fraudulent addresses directly connected to DeFi protocols of different categories.

1. By analyzing the given transaction data, we show which protocols are used more frequently. We find that decentralized exchanges do not only have a high amount of relationships with other nodes, but we also find that they receive the biggest amounts of illicitly obtained funds.

2. In a more detailed trace analysis, we show which trace patterns occur the most, and how illicitly obtained funds move inside the most heavily used protocols. We also show how funds are converted to different tokens with the use of these protocols.

3. We show the most common traces related to the most used protocol and give an insight into how Ether is swapped within this protocol.

4. Last but not least, we check common similarities inside of trace patterns and see if there are any particular tokens that are heavily used by fraudulent entities.

Chapter 5 Discussion: In this chapter, we will summarize our insights, show limitations to our analysis and talk about possible future extensions of the work conducted in this thesis. **Chapter 6 Conclusion**: We finish the thesis with a conclusion of our methods and findings by giving a concise summary of our results.

CHAPTER 2

Background

The field of cryptocurrencies, which was essentially introduced to the public by the creation of Bitcoin, is still relatively young. Although a similar ledger by the name of KARMA existed even 5 years prior to Bitcoin, Vishnumurthy et al. [VCS03] only used their blockchain in a peer-to-peer filesharing application, where they prevented actors to freeload on resources. When Bitcoin was introduced around 2009 [Nak08], it had a broader general scope, as it was the first ledger to have proof-of-work-secured tokens that could not only be applied to one use case, but rather it became the first global, decentralized transaction ledger with many of the application fields we talked about earlier. After the introduction of Bitcoin, it didn't take long until other distributed ledgers started to show up as well.

One of those ledgers was Ethereum which was publicly launched on July 30, 2015. With the possibility to exchange funds in a non-authorized way plus the possibility to execute arbitrary code on the blockchain, Ethereum quickly grew to the second largest blockchain, counting as many as 500,000 active addresses every day, as of February 28, 2022.

In the following chapters, we give a brief introduction to Ethereum and explain the main concepts which are particularly relevant to understanding this thesis: accounts, state, transactions, blocks, SCs as well as message calls and token systems. We also take a glimpse at how transactions are stored on the blockchain to understand how the data we want to investigate in this thesis can be obtained. Further, we explain what exactly defines a DeFi protocol and why this sub-part of protocols comprises the main interest of our investigations.

2.1 Ethereum

People familiar with blockchain and cryptocurrency concepts most certainly also have heard of the Ethereum blockchain and the same-named native currency Ethereum

(ETH) (also denoted as Ξ). Ethereum is a second-generation blockchain and its SC capabilities make it popular for the creation of decentralized applications. With a market capitalization of over $300,000,000,000 Ethereum is the second largest blockchain after Bitcoin, according to market cap, and currently trading at $2,605.19 per Ethereum token.

2.1.1 Basics

Before the Ethereum blockchain was launched, there were as few as 18 other cryptocurrencies in existence yet[1], of which Bitcoin and Litecoin appeared first in 2009 and 2011. Two years later, in November 2013, Vitalik Buterin published a white paper [But14] explaining the fundamental concepts of Ethereum. Following Buterins early work, there were 7 more people, that are today considered co-founders of Ethereum: Charles Hoskinson, Gavin Wood, Anthony Di Iorio, Amir Chetrit, Jeffrey Wilcke, Mihai Alisie, and Joseph Lubin. They helped realize the project in the years to come until the Ethereum blockchain went live on July 30, 2015. Ethereum is not only the blockchains name and the name of the blockchain's native currency, but rather Ethereum is an actively developed open-source project[2] that attempts to build generalized technology on which all transaction-based state machine concepts may be built [Woo14].

As the main focus of this thesis is transactional behaviour and investigating fraudulent accounts, we will purposefully leave out some details about Ethereum, as they are not necessary to understand the scientific work we will conduct. What we will look at are the basic principles and terms, that are used, as well as all the parts that are necessary in order to understand the main concepts of this work.

Ethereum is a peer-to-peer network like Bitcoin, that uses an alternative protocol for decentralized applications. As Ethereum's specification was partly derived from the ideas of Bitcoin, it is also often referred to as an alt-coin (alternative coin). In this section we will mainly focus on five main parts which are relevant for this thesis, being accounts, state & transactions, Smart Contracts as well as message calls (sometimes also referred to as internal transactions), and token systems. One of the main goals of Ethereum is to facilitate transactions between consenting entities, without the need for a trustful environment. This means, that two individuals can come to a specific agreement, without having to trust each other in any way, because the agreement will be enforced autonomously [Woo14]. Ethereum, taken as a whole, can also be viewed as a transaction-based state-machine, that begins with a genesis state, which over time transforms to another state whenever a transaction is executed [Woo14]. This world-state is referred to as σ. From this world-state σ, we can deduce information on account balances, transactions that led to this state, as well as any other information that can be represented by a computer [Woo14].

[1] https://en.wikipedia.org/wiki/List_of_cryptocurrencies
[2] https://github.com/ethereum

2.1.2 Accounts

In Ethereum accounts are stateful entities comprised of 4 different fields:

- **nonce**: represents the number of transactions sent from this address or the number of contract creations made by this account
- **balance**: a value referring to the number of Wei (the smallest subdenomination of ETH) owned by an address
- **storageRoot**: a 256-bit hash that encodes the content in the storage of the account
- **codeHash**: a hash of EVM code that this account stores

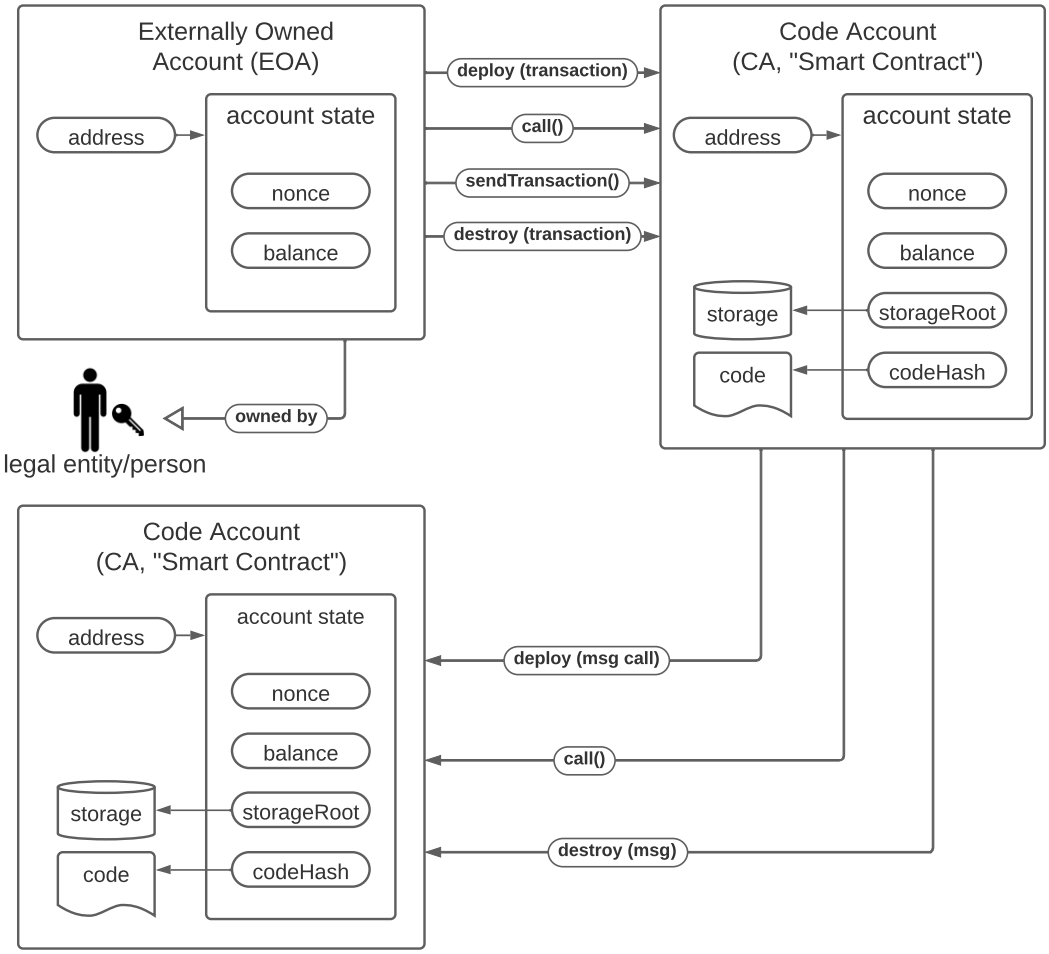


Figure 2.1: Externally Owned Accounts and Code Accounts

In Ethereum, there are two different account types: accounts that have no Ethereum Virtual Machine (EVM) code associated with them, and accounts that do. In practical terms, we refer to these two different types of accounts as Externally owned accounts (EOAs) and Code accounts (CAs), informally also referred to as SCs. In Section 2.1.5 we will investigate the functionality of SC in more detail. An Ethereum address that identifies an EOA (also often referred to as a user account) is comprised of 160 bits, meaning that an account address contains 40 hexadecimal digits:

Example: 0xb794F5279a39494cE8396eA0bfffBA7413579268

Most of the time, the address is prefixed with the common identifier for the hexadecimal representation "0x". CAs have the same format as EOAs and are indistinguishable from them, given only the address is known. But contract accounts are distinguishable from user accounts by checking if the codeHash field is the hash of an empty string e.g. $\sigma[a]_c = KEC(())$. If the hash is not the hash of an empty string, then there is code stored within this account, and the account therefore must be a CA. User accounts are associated with their respective private key, such that only authorized users can access EOAs (given they know the private key that is associated). Contract accounts do not have any association with private keys, rather their key is based on the code the contract contains itself, and anybody can interact with it. The address of a contract is computed from information that is based on the creator of the contract.

2.1.3 State & Transactions

In this section, we will describe the two most important aspects of Ethereum: state and transactions. State and transactions are the two core concepts that the Ethereum blockchain builds upon. This is why Ethereum is often also referred to as a transaction-based state machine. A transaction is a cryptographically signed instruction, that when executed on one state will result in a new state. This behaviour is formally defined in the Ethereum yellow paper as: $\sigma' = \Upsilon(\sigma, T)$ where Υ denotes the Ethereum state transition function, with T being the transaction and σ the current state onto which transaction T will be executed. The σ' notation denotes the post-transactional state.

Of course, not every state change is also automatically valid, and in fact, invalid state changes occur all the time. Fortunately, for the scope of this work, we will not have to worry about how invalid state transitions are handled, as only already verified states end up on the blockchain. As the transaction data we investigate in this thesis comes from a valid state of the blockchain it means that all the transactions that led to this state must also be valid.

If a valid state transition denoted by $\sigma_{t+1} \equiv \Upsilon(\sigma_t, T)$ occurs, it has a chance of being included in the next block of the blockchain. If the transaction is included in the next block is dependent on different factors. How the creation of blocks (mining) functions as well as how blocks are built is briefly explained in Section 2.1.4, although this functionality of the Ethereum blockchain is not too relevant for the work in this thesis. For our use case, it is sufficient to know, that transactions represent valid state transitions, and that

states can save arbitrary data such as account balances for example. Figure 2.2 shows a simplified version of how transactions are included in the blockchain.

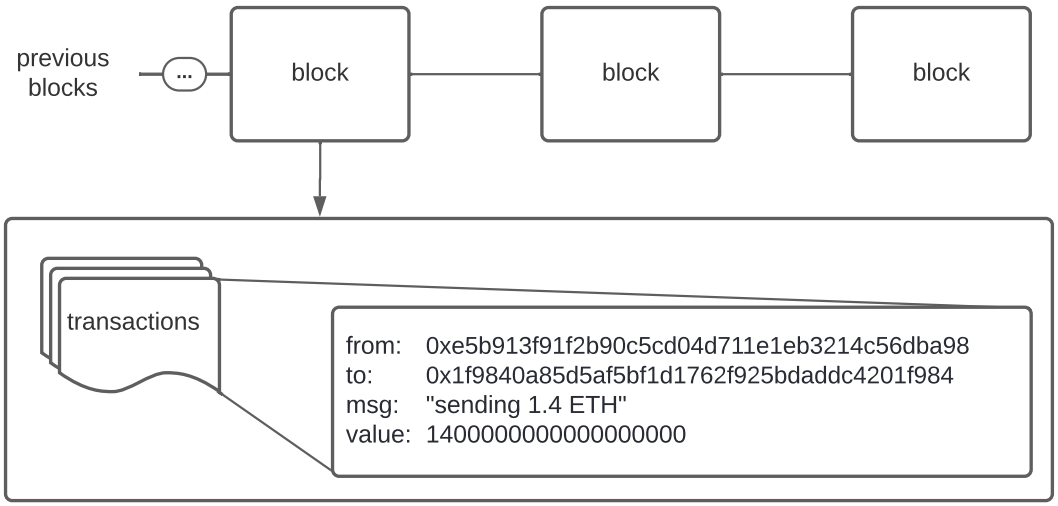


Figure 2.2: How transactions are included into the blockchain

The Ethereum yellow paper states that there are two different types of transactions:

- transactions which result in message calls, and
- transactions which result in the creation of new accounts with code (this is also referred to as contract creation)

We will talk about the creation of contracts and the resulting code accounts in the section Smart Contracts, but for now, let us investigate the transactions that are denoted as message calls:

In the case of executing a message call, there are a few required parameters that are of interest for our work. These parameters consist of the sender (s), transaction originator (o), recipient (r), available gas (g), value (v), and gas price (p), d which stands for the data that can be specified as input for this specific call, as well as e which describes the depth of the message-call/contract-creation stack. For us, only the sender, the recipient as well as the sent data, and the transferred value are of interest, as those are the main parts of a transaction we need to know, in order to investigate it. In case of executing a transaction referring to contract creation, an additional byte array i is also sent along. The additional parameter i describes the initialization EVM code, that is used to instantiate a callable contract on the Ethereum blockchain.

2.1.4 Blocks & Mining

A block consists of a collection of transactions that are run on a current state. The blocks themselves lay the foundation for the blockchain, the ledger recording the series of transactions by utilizing cryptographic hash functions such that prior blocks can be referenced uniquely. We can denote a block B as $B \equiv (B_H, B_T, B_U)$ where B_T is a series of transactions $(T_0, T_1, ...)$, B_H is the block header that contains a variety of information such as the reference to its parent, a timestamp and many other things. B_U refers to a list of ommer block headers (ommer blocks are blocks that have a parent equal to the present block's parent's parent). As we know that there are many transactions in one block there is also a block-level state transition function Π which is written as follows:

$$\Pi(\sigma, B) \equiv \Omega(B, \Upsilon(\Upsilon(\sigma, T_0), T_1)...)$$

Where Ω is the block-finalization state transition function for block B. Ω is also responsible for rewarding block creators with a mining reward and should incentivize participants of the blockchain to put computational effort into creating new valid blocks (mining). As block creation is a substantial part of how the Ethereum blockchain functions, we wanted to mention it here shortly, but note that it has no further significance in the scope of our thesis.

Now that we have a simplified overview of transactions and how transactions that were triggered by external actors are stored in blocks, we will also take a closer look at how message calls, calls which can also be triggered by code execution of a contract, are handled. We will also look at how CAs work and show how traces of executable SC functions can be investigated for an even more detailed analysis.

2.1.5 Smart Contracts & Message Traces

Smart Contracts, often also referred to as distributed apps (or short dApps), have become very popular in the last years. But it was quite some time ago when the term Smart Contract was first introduced by Nick Szabo in the late 1990s when he wrote and published the article "Smart Contracts: Building Blocks for Digital Markets" [Sza96]. His idea was to use a distributed ledger to store real-world contract information. The contracts should be embedded in the hardware and software we deal with, in such a way as to make breach of contract expensive for the breacher [Sza96]. In his view, there were some major objectives that would be addressed by creating cryptographically secured protocols in opposition to having contracts that have to be secured by physical force (like arrest or confiscation). His vision was to create contracts that would be observable, verifiable, immutable, and enforceable [Sza96].

In simpler terms, SCs on Ethereum are computer programs deployed onto the blockchain. They are pieces of code that can be triggered to execute, they are able to hold data and make transactions to other SCs such that they update their state according to the specification that was programmed into them when the contract was initially deployed.

The exact execution model of SCs would go beyond the scope of this thesis, nevertheless, we want to briefly show some applications of SCs as Ethereum allows for SC possibilities in various fields.

For example, SC can be used to fund projects and cryptographically make sure that the funding will only reach the project's managers in case the funding is fully completed, and is returned to the investors otherwise. Banks can use them to issue loans as well as to automate payment processes. Insurance companies can use contracts to help them process claims, and there are a lot more options when it comes to applicable fields and use cases. One of the spaces using the capabilities of SCs extensively is DeFi.

On the Ethereum blockchain, SCs are initialized by setting EVM code as a parameter when creating a new account. Contract creation can either fail or successfully alter the state in which the contract will persist from this point on. On contract creation, different restrictions can be set. This mechanism known as modifiers allows us to change the behaviour of functions inside a SC. Modifiers can, for example, be used to make sure not every user can interact with the contract in the same way. We can think of this mechanism as giving certain privileges to different accounts. This feature of restriction is needed for a variety of software-engineering-specific nuances, as the code set on contract creation is immutable and cannot be changed thereafter. For example, it makes it possible for privileged entities to update their software by using certain patterns, for example, proxies.

After a contract successfully has been created, everybody owning an account can now interact with the contract by the means of message calls. For a message call, there are a few parameters that are of particular interest for our thesis:

- **transaction originator** EOA address from which the initial message call originated
- **transaction_hash** a unique identification hash for the transaction this message belongs to
- **sender** the address (CA or EOA) that initiated the message call
- **recipient** (usually) the account whose code is to be executed
- **value** the value that is associated with the trace
- **input data** the input data of the message call
- **function_signature** the signature of the function that is called (we can derive this signature from the input data)

In particular, we can look at all the different addresses that sent and received message calls, in order to trace all function calls, not only calls executed by EOAs but also from SCs themselves. In cases where SCs call functions of other SCs, an execution chain of message calls can be constructed.

These so-called message traces emerge as contracts call functions of other contracts in order to execute some logic. By calling code that resides in other contracts, arbitrary code can be executed, side effects can occur and data or value transfers can be triggered. Because of their different applicability, traces may contain hundreds of message calls, as a result of a single transaction triggered by an EOA.

Message calls from SCs to other SCs are also often referred to as internal transactions. In Figure 2.3 we can see a trace of a message call of length 8. It consists of 4 direct sub-traces (0, 1, 2, 3), one of them having 3 sub-traces itself (3|1, 3|2, 3|3). The edge denoted as tx is the initial transaction from an EOA triggering all the other message calls. By looking at the first 4 bytes of the input data which is provided in the message call (the so-called function signature), we can check which exact function of a SC was called in each step of the trace. In the figure, we can see the function signatures of every message call appended to each edge of the graph. The most important trace properties of the same trace in its raw tabular form can be seen in table 2.1. We can see that the first transaction going to the address of the Uniswap V2 Router is associated with a value transfer of 2 ETH, and the function that is called is $swapExactETHForTokens(...)$. In this manner, we can follow the whole trace of the transaction giving us a better understanding of what happens on the blockchain, not only on a transactional value-transfer level but also on a logical, semantic level.

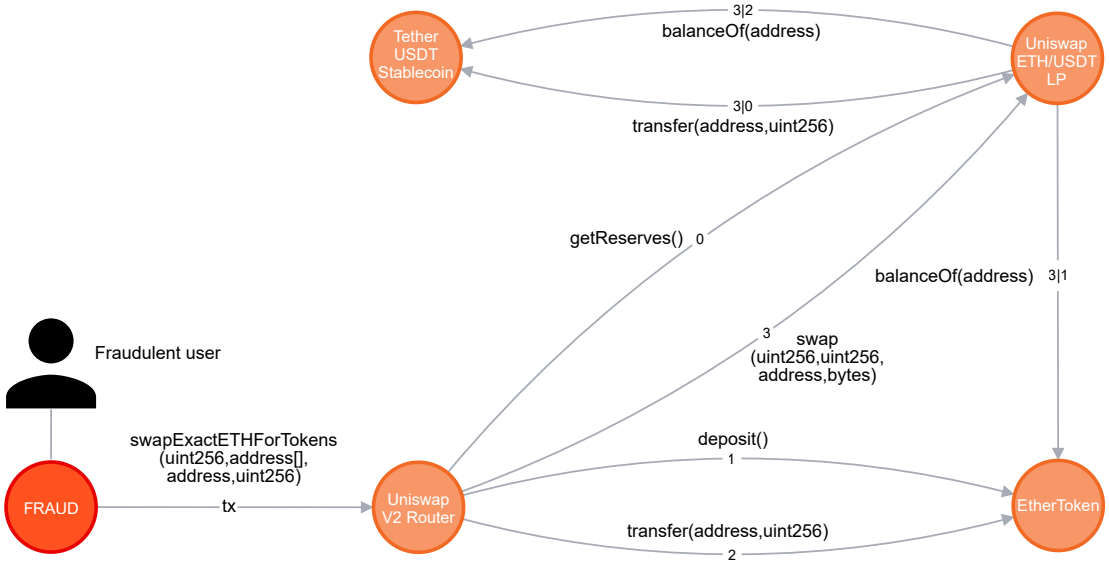


Figure 2.3: Trace showing a swap from ETH to USDT visualized as graph

tx_id	from	to	value	hex_signature	text_signature
tx	0xd5edf...	0x7a250...	2.0	0x7ff36ab5	swapExactETHForTokens(...)
0	0x7a250...	0x0d4a1...	0	0x0902f1ac	getReserves()
1	0x7a250...	0xc02aa...	0	0xd0e30db0	deposit()
2	0x7a250...	0xc02aa...	0	0xa9059cbb	transfer(...)
3	0x7a250...	0x0d4a1...	0	0x022c0d9f	swap(...)
3|0	0x0d4a1...	0xdac17...	0	0xa9059cbb	transfer(...)
3|1	0x0d4a1...	0xc02aa...	0	0x70a08231	balanceOf(...)
3|2	0x0d4a1...	0xdac17...	0	0x70a08231	balanceOf(...)

Table 2.1: Message trace from figure 2.3

2.1.6 Tokens and Token Systems

The definition of what a token exactly is, and what a token is used for depends on context, and there are different views on what should be labeled as a token and what should be labeled as cryptocurrency.

A token can generally be seen as some unit managed by a token contract. Although the management by contract is not necessary, e.g. alt-coins are also often labeled as tokens. As there is no exact definition of what a token can describe, and what its utilities are, we want to mention two main takes on crypto tokens:

1. Any cryptocurrency is a token because - technically - all crypto assets can be described as such.[Blo22]
2. Tokens are crypto assets that run on top of another cryptocurrency's blockchain.[Blo22, Fra22]

Although Bitcoin and Ethereum are more commonly referred to as cryptocurrencies instead of tokens, basically all other alt-coins (alternative coins) are mostly addressed as tokens, whether they are native tokens running on their own blockchain or not. Since we only investigate data inside the Ethereum ecosystem we can go with the second approach and state that tokens are a special form of SCs, running on top, and making use of, the Ethereum blockchain. Far more important than actually coming up with a solid definition for what a token exactly is, is the variety of different use cases that tokens and token systems provide.

Crypto-tokens mainly emerged after the launch of Ethereum in 2015, as by the use of its general-purpose programming language it became very easy to create decentralized applications and also tokens that these applications would use. There is a variety of different tokens, such as DeFi tokens, governance tokens, Non fungible tokens (NFTs), security tokens, and utility tokens serving different purposes.

As the Ethereum whitepaper states: "Token systems have many applications ranging from sub-currencies representing assets such as USD or gold to company stocks, individual

tokens representing smart property, secure unforgeable coupons, and even token systems with no ties to conventional value at all, used as point systems for incentivization." [But14]

The primary building blocks of all tokens on the Ethereum blockchain are SCs. Basically all token contracts have at least one thing in common, which is the possibility for entities to transfer tokens between each other. A simple implementation of a token transfer can be seen in Listing 2.1.

```
function transfer(address receiver, uint amount) public returns (bool) {
  require(amount <= balances[msg.sender]);
  balances[msg.sender] = balances[msg.sender] - amount;
  balances[receiver] = balances[receiver] + amount;
  emit Transfer(msg.sender, receiver, amount);
  return true;
}
```

Listing 2.1: Basic transfer function of a token contract

Of course, this implementation is a very basic example of a token transfer and as seen in the listing, there is no other use-case tied to this token, besides the actual transfer. As we have a very wide range of what a SC can actually define, there is also a wide range of functions that a token can exhibit. DeFi tokens for example implement functions used in traditional financial systems such as lending, saving, and trading tokens in a variety of ways. Governance tokens give holders the possibility to participate in votes regarding different issues, for example, an upgrade to a decentralized application. Access-control features can also be implemented using governance tokens. NFTs are used to represent ownership rights for real-world or virtual assets and are heavily used to issue digital artworks and in-game items [PKPD22, FP21]. Security tokens are assets for example used to represent partial ownership of a company like traditional shares.

With such a variety of different tokens, there emerged the need for a unified specification, as to make sure tokens could also interact with each other, also on different blockchains, in a standardized way. For this purpose different token standards emerged over time:

- **ERC-20** The Ethereum Request for Comments (ERC-20) token standard was proposed in November 2015 and implements an Application Programming Interface (API) that provides functionalities such as token transfers, getting the current token balance of an account, getting the total supply of all tokens available and third party spending approvements. Over 500,000 ERC-20 compatible tokens are deployed on the Ethereum blockchain, making it a viable standard.
- **ERC-721** The ERC-721 token standard is a non-fungible token standard, meaning that contrary to the ERC-20 specification where all tokens are the same, this standard ensures, that all tokens are different. Therefore tokens from this standard can be used to represent ownership over digital or physical assets [ESES18].

- **ERC-777** ERC-777 defines a backward compatible standard with ERC-20 specification that allows for more advanced interactions with tokens. With this standard operators can for example send tokens on behalf of other addresses.
- **ERC-1155** The so-called Multi-Token Standard provides functionalities for contracts to manage multiple tokens at once. It was originally introduced in 2018, as it became clear that the ERC-20 and ERC-721 standards were responsible for a lot of redundant bytecode on the Ethereum blockchain, plus they had limited functionality in some cases [RCC+18].

The ERC-20 token standard is by far the most commonly used standard on the Ethereum blockchain, as it was the first standard that became widely accepted, and a lot of other standards build upon it today. The ERC-20 token standard enables tokens to easily interact with wallets, exchanges, and a range of SCs, also in the DeFi space, that use ERC-20 functionality to ensure compatibility.

In Listing 2.2 we can see the standard ERC-20 methods, that an ERC-20-compatible SC must implement. Because we know what the different standardized functions are designed for, they will provide us with insight into what different message traces do internally, when one of these functions is called.

```
function name() public view returns (string)
function symbol() public view returns (string)
function decimals() public view returns (uint8)
function totalSupply() public view returns (uint256)
function balanceOf(address _owner) public view returns (uint256 balance)
function transfer(address _to, uint256 _value) public returns (bool success)
function transferFrom(address _from, address _to, uint256 _value) public returns (bool success)
function approve(address _spender, uint256 _value) public returns (bool success)
function allowance(address _owner, address _spender) public view returns (uint256 remaining)
```

Listing 2.2: ERC-20 Methods

For example, if we see a *transfer()* function being called in an ERC-20-compatible token, we can conclude that there were tokens transferred from the owner of the contract to some other entity. This will become useful for us as we will try to understand what different message traces do in chapter 4.3.

2.2 Decentralized Finance (DeFi)

In this chapter, we will explain more thoroughly what contracts are considered as *decentralized finance* and why DeFi is an important topic to investigate when it comes to illicit transaction behaviour.

If we think about financial transactions today, what probably comes to most people's minds, is some form of bank. With the help of banks, they can send and receive money to and from others. The banks are responsible for handling all the legal conditions, they guarantee that our money will be safe, and they make our money accessible via debit and credit cards, as well as ATM machines and Internet portals, where we can easily view and manage our holdings. Banks are intermediaries, that provide us with the services we need in our economic world every day. They easily let us connect with others, without having to establish trust ourselves, and they handle transactions in a reliable and fast manner. Although banks provide great benefits, they often also enjoy substantial power in shaping the economic landscape, and they can leverage their power to maximize self-interests, raising concerns over their monopoly power [CB20].

With the growing concerns over regulatory power in the last decades, blockchain technology and its applications make the perfect substitute for large-scale financial institutions. With decentralized platforms and cryptographically backed trust mechanisms, blockchain-based financial systems can eliminate the need for intermediaries, and help to make financial services more accessible, transparent and maintainable. DeFi is a collection or a subset of SCs that facilitate functions and frameworks, such that people from all around the world can interact with these contracts. People can use the different contracts to save, invest, access, and manage their money in a decentralized way. According to Werner et al. [WPG+21] there are four main properties that a perfect DeFi system should exhibit. It should be:

1. **Non-custodial**: participants have full control over their funds at any point in time
2. **Permissionless**: anyone can interact with financial services without being censored or blocked by a third party
3. **Openly auditable**: anyone can audit the state of the system, e.g., to verify that it is healthy
4. **Composable**: its financial services can be arbitrarily composed such that new financial products and services can be created (similar to how one is able to create new Lego models based on a few basic building blocks)

It is these properties, that inherently differentiate decentralized systems from their centralized counterparts. Over time centralized institutions have accumulated a lot of power, and now that their systems are established, it is hard to restructure the monopoly position of such big players. Their internal structure is often not visible to outsiders, and it is hard to monitor and correctly regulate centralized systems. DeFi systems on the other hand are transparent and there are (theoretically) no single authorities that are in control of regulation.

As DeFi systems have a great variety of applications, the total value locked in DeFi had first reached $25 billion in December 2020 and has since then reached over $100 billion

by the end of 2021 only to come back down about $75 billion as of March 2022. The most prominent application areas where DeFi has already set foot are assets, exchanges, lending services, derivatives, and payment services. These are only some of the main use cases that DeFi offers, but there are many more, and it would take a long time to list every single one of them, as there are hundreds of different protocols already deployed on various blockchains. Therefore we will only look at the most prominent application areas, mentioned above, and explain how DeFi managed to replace or at least supplement the given centralized systems we are so used to.

- **Assets**: DeFi assets are tokens that utilize SCs to provide additional features to regular assets. On the one hand, the tokens can be seen as simple investments, but on the other, the token could be responsible for community-driven decision-making, participating in stake pools, or yield farming, which involves lending your tokens to gain yield by interest rates or transaction fees. DeFi assets often have multiple usecases, and they come in a lot of different forms. DeFi protocols of the asset category are for example Badger[3] or Convex[4] among many others.

- **Decentralized Exchanges (DEXs)**: A DEX let you trade different tokens for other tokens at any time you like. It is like exchanging currencies on a regular exchange, with the difference, that the market will not be closed like a regular market. DEXs function 24 hours a day, every day of the year. Examples of well-known DEXs are Uniswap[5], Sushiswap[6] or 0x[7], but there are many more.

- **Lending and Borrowing Services**: In decentralized systems, for somebody to get a loan, it is not necessary to provide personal information. Neither the borrower nor the lender have to identify themselves and they don't even have to know each other. Still, they can come to a mutual agreement on what terms they want to borrow or lend their money. The main advantage of the decentralized setting is that you are not only having access to funds in the custody of your personal bank, but rather to all the available funds at that given time. No more does it matter, what your location is, or what lending programs your bank offers. By having a lot of people offer loans on their own terms, loans become accessible for everybody, and interest rates can improve as there is more competition in the market. Even loans without collateral (flash loans) are possible in DeFi, which would basically be impossible in the current centralized setting. Some known lending and borrowing services are Aave[8], Compound[9] and Maker[10].

[3] https://badger.com/
[4] https://www.convexfinance.com/
[5] https://uniswap.org/
[6] https://sushi.com/
[7] https://www.0x.org/
[8] https://aave.com/
[9] https://compound.finance/
[10] https://makerdao.com/

- **Derivatives**: Derivatives work in a very similar way as their centralized counterparts do. Derivatives are contracts deriving their value from underlying assets, commodities, or indices. They are mostly used to hedge the risk associated with crypto-exposure, but also for speculation. Again, the beneficial part of decentralization is, that everybody has access to these derivatives, and they can even be created by anybody. Synthetix[11] and dYdX[12] are among the most common protocols.

- **Payment Services**: Payment services also work very likewise to their centralized counterparts. If two entities want to exchange value, there needs to be a fast system that handles the transaction, in such a way, that transferred value can be used immediately after it was sent. Specifically focusing on simple transactions, there have emerged multiple protocols and networks, that solely focus on improving transaction throughput whilst not lacking in privacy and security. Celar[13], Connext[14] or Matic[15], just to mention a few, all provide services in this sector of DeFi.

As DeFi has emerged as one of the biggest application fields of cryptocurrencies, there is also an ever-growing amount of scams and illegal activities that involve the above-mentioned protocols and their provided functionality. DeFi, often being loosely regulated, is a welcoming opportunity for fraudulent entities to launder their money using these protocols. There are a lot of different criminal activities related to Ethereum and protocols in general, which we cover in a bit more detail in the next section.

2.3 Criminal activities on the Ethereum blockchain

As the 2021 Crypto Crime Report from Chainalysis states, illicit activity represented 2.1% of all cryptocurrency transaction volume in 2019, or roughly $21.4 billion worth of transfers. In 2020, the illicit share of all cryptocurrency activity fell to just 0.34%, with one of the reasons being, that the overall economic value almost tripled in 2019. Nevertheless this 0.34% still accumulates to about $10.0 billion in illicit transaction volume [Cha21]. The Crypto Crime Report has also shown, that DeFi is becoming a more and more prominent destination of stolen cryptocurrency in the last 2 years and that "DeFi protocols received 17% of all funds sent from illicit wallets in 2021, up from 2% the previous year. That translates to a 1,964% year-over-year increase in total value received by DeFi protocols from illicit addresses, reaching a total of $900 million in 2021" [Cha22].

The history of criminal activities related to cryptocurrencies basically started as soon as the first cryptocurrency came around. As technology and society have progressed, so have criminal activities and their applications. The connectedness of the Internet makes it an

[11] https://www.synthetix.io
[12] https://dydx.exchange/
[13] https://www.celer.network/
[14] https://www.connext.network/
[15] https://polygon.technology/

easy task to spread malicious information all around the globe, while staying relatively anonymous. Fraud, scams and pyramid schemes, all of which existed prior to the Internet, have now become activities, that can target an even bigger audience ever more easily. The predominant use case that cryptocurrency provides in relation to illegal activities is the process of laundering money, such that it is hard to trace where funds originally came from. As the European Union Agency for Law Enforcement EUROPOL states, "Recent years have seen cryptocurrency increasingly used as part of criminal activities and to launder criminal proceeds. Criminals have also become more sophisticated in their use of cryptocurrencies. In addition to using cryptocurrencies to obfuscate money flows as part of increasingly complex money laundering schemes, cryptocurrencies are increasingly used by criminals as a means of payment or as an investment fraud currency." [Eur21a]

To better understand different illicit schemes and how these are orchestrated, we will briefly explain the main activities that make out some of the illegal proceedings connected to cryptocurrencies.

2.3.1 Scams & Fraud

As a lot of institutions point out, scams and fraud are not quite the same thing, but successful scams or frauds almost always lead to monetary loss for the victims. Scamming is considered to be the little brother of fraud as it is not quite as sophisticated in orchestration and also potential revenues are often considered to be lower. A scam may be perpetrated by a small group or even a single entity, often making use of false promises. The sophistication of fraud is considered to be higher, often conducted by a group of insiders, and the possible amounts involved in frauds are considered to be higher also. Nevertheless, the two terms are quite often used interchangeably. As the annual report on the criminal use of cryptocurrency by Chainalysis shows, scamming is the most frequently identified illicit activity in the crypto space. Their reporting shows that over half of all the detected activities (54%) were related to scams, which accounted for $2.6 billion in 2021. [Cha21]

When it comes to trading and investing, most retail investors rely on trusted exchanges, to get their hands on cryptocurrencies. Fiat money is transferred to a bank account that belongs to the exchange, and in return, the investor gets some cryptocurrency, based on the current exchange rate. But exchanges are not the only way to get your hands on cryptocurrencies, and they are also not the only way to trade crypto assets for fiat money or other currencies. Peer-to-peer (P2P) trading is often used to transfer funds between two or more entities without the need of any intermediaries. As P2P basically skips the identification process and can be used by anybody whenever they want, it is the predominant way of transacting money pseudonymously from one entity to another. As there is no real way to identify a person or entity behind an arbitrary address, without sophisticated investigations, P2P transactions are often used in all kinds of scams [ByB22].

A common scheme is to promote tokens by using various websites and social media to

trick people into investing in certain tokens. People are lured by high potential profits and fake communities that back up those frauds. The founders of the tokens often artificially inflate the price and then leave with the majority of the money before investors realize that the promised use case or investment opportunity was just a stunt to obtain their money. This method is known as **exit scams**, as the fraudsters "exit" with a lot of money, often resulting in the burst of the price bubble. Other schemes rely on hacked social media accounts to promote free giveaways of coins or tokens if you send a certain amount of tokens in the first place. These so-called **giveaway scams** are a form of advance fee fraud, a category of crime that covers myriad versions of the same basic premise: I have something valuable to give you; to release it, send me a payment ([GSD04] as cited in [Mac22]). Of course, there are no free coins to be obtained in a giveaway scam, and the amount of cryptocurrency sent to the address responsible for doubling or tripling the victim's coins is just lost once the transaction is executed.

Another subcategory of scams are **pump and dump** schemes in various forms, where groups of people promote to artificially raise the price of a coin by buying lots of coins at a certain time. Just after the collective buying should start, the promoters then sell large amounts of the agreed-upon cryptocurrency, in which they had been invested way before everybody else got on board. This often dumps the price and leaves the scammed buyers with prices well below their buying price, and no way to regain their losses in the near future. Pump and dump schemes are illegal, however, cryptocurrency exchanges are, for the moment, unregulated and difficult to police and those involved often escape prosecution [B+18]. Other fraud schemes involve promoting investment opportunities or trading signals or even sophisticated simulated trading environments just to get a hold of personal information, subscription fees, or initial money deposits. Basically, the possibilities for fraud are endless, and in combination with pseudonymity, it certainly is an easy opportunity to make money in an illicit way.

2.3.2 Exploits

Since cryptocurrency has been around, multiple protocols have become targets of sophisticated hacks, whether it was the exploit of certain bugs or social engineering, that led to huge amounts of stolen funds. DeFi-related hacks have accounted for 76% of all major hacks in 2021, and users have lost more than $361 million to attacks on DeFi platforms in 2021, as CipherTrace states in their *Cryptocurrency Crime and Anti-Money Laundering Report* in August [Cip21].

In 2016 the hack of the Decentralized Autonomous Organization (DAO) made headlines. Somebody had discovered an exploit in the Smart Contract code that was deployed to the Ethereum blockchain. At its peak, the DAO contract held over 12 million Ethereum, of which the attacker managed to drain 3.6 million, today worth over $160 million. The hack only occurred merely one year after the blockchain was launched in 2015, and because of the huge amount of stolen funds, the Ethereum community made the controversial decision to hard-fork the Ethereum blockchain. The original chain with the stolen funds is today known as Ethereum Classic, while the blockchain that got reset to a point in

time before the hack occurred, is today known as Ethereum. But we don´t have to look all the way back to 2016 to find exploits that gained large amounts of tokens. Just in the beginning of 2022 alone, over 50 exploits have been reported in the REKT database of DEFIYIELD[16]. As the Ethereum blockchain is still *the* go-to SC platform it is no surprise, that there are numerous big Ethereum-related exploits listed just in the last few years.

In August 2021 the Poly Network project, a DeFi application providing cross-chain transactions, got targeted by an undisclosed attacker, who drained over $600 million in different cryptocurrencies from a Smart Contract to external wallet addresses. The exploit was conducted by exploiting a security flaw, that allowed the hacker to execute transactions only reserved for privileged entities. In the aftermath of the exploit, the attacker surprisingly returned all of the funds, earning him the name "Mr. White Hat".

Also in 2021, a Play-to-earn NFT platform - Vulcan Forged - also reported it had suffered an attack, and a big amount of the token's total supply (nearly 9%), worth $140 million at that time, was stolen. To the luck of investors, the platform was capable of returning the stolen tokens to everybody that was affected, but not all projects are this fortunate.

Two months earlier, Cream Finance - another DeFi project - also reported major losses as an attacker exploited a SC by using a flash loan attack. These attacks work by creating an arbitrage opportunity by using lent tokens in a favorable way to exploit artificially created price discrepancies. This attack shows, that even after continuous security improvements of SCs there can still be exploitable parts, as this was the third time Cream Finance was hacked in 2021, losing $37 million in February, $29 million in August, and $130 million in the latest attack.

BadgerDAO, Compound Labs, EasyFi, and many others are all among projects in the DeFi space, that suffered exploits responsible for multi-million dollar losses. And the list does not stop there. Attentive readers might have noticed, that we only mentioned protocols, that reside on the Ethereum blockchain. Looking at the full picture shows an even more devastating reality of a total of $3 billion in lost funds, of which only 30.9% happened on the Ethereum blockchain, or had ETH tokens involved. It also shows, that only about $700 million were ever returned, of which the hacker of the Poly Network is responsible for $600 million. As of 28th of March 2022, they listed 2730 malicious acts of which 934 targeted Ethereum-based protocols. 134 entries are listed as exploits. Also, the biggest losses are attributed to exploits. We can clearly see, that DeFi has become one of the main targets of hackers and malicious entities, especially as financial services are the ones where the money resides, giving attackers an outlook to make huge profits when targeting DeFi protocols.

2.3.3 Ponzi Schemes

In the 1920s, an Italian crook named Charles Ponzi caught investor's attention by promising extraordinarily high returns within just a few months. What he claimed was

[16] `https://defiyield.app/rekt-database`

an investment in international mail coupons, turned out to be one of the biggest scam schemes known to that date. In its essence, a Ponzi scheme pays earlier investors with investments of later investors [WAH12], without making any or only little legitimate earnings itself. Therefore Ponzi schemes need a constant flow of new money with which earlier investors can be paid. The non-transparent schemes often gained popularity among new investors, as the potential success of the investment is proven and promoted by early adapters. The Ponzi schemes work as long as newer and bigger investors can be found, but sooner or later all Ponzi schemes meet the same fate of rapid collapse, unable to pay the promised profits.

Recruiting new investors was certainly a time-consuming task back in 1920, but since then a lot has changed, and today the Internet provides the basis for easier recruitment as well as easier promotion of fraudulent schemes all around the globe. Although Ponzi schemes as well as their detection and analysis on the Ethereum blockchain are well-researched [CZC+18, BCCS20, YJX+21, ZYL+22], SCs implementing Ponzi schemes show up in the hundreds. Persisted on the blockchain, Ponzi schemes written as SCs have the appearance, as if the constant flow of new investors is even more certain than in the real world, as the contract will exist on the blockchain forever, with no way for regulatory entities to remove it. This creates a false sense of continuity and the suggestion that the scheme can go on forever. This sense of continuity combined with the promise of quick returns even leads sophisticated investors to join these schemes. Even though they understand the fraud - they still hope to profit by joining early [MHC12]. With SCs often no bigger than 200 lines of code, anybody can become the new Charles Ponzi.

```
pragma solidity ^0.5.0;
import "@openzeppelin/contracts/math/SafeMath.sol";

contract Doubler {

    using SafeMath for uint;

    address payable public owner;

    struct User {
        address payable addr;
        uint amount;
    }

    User[] public users;
    uint public currentlyPaying = 0;
    uint public totalUsers = 0;
    uint public totalWei = 0;
    uint public totalPayout = 0;
    bool public active;

    constructor() public {
        owner = msg.sender;
        active = true;
    }

    function close() public{
        require(msg.sender == owner, "Cannot call function unless owner");
        require(active == true, "Contract must be active");
        require(address(this).balance > 0, "Must have balance > 0");
        owner.transfer(address(this).balance);
        active = false;
    }
```

```
    function join() external payable{
        users.push(User(msg.sender, msg.value));
        totalUsers += 1;
        totalWei += msg.value;

        owner.transfer(msg.value.div(10));
        while (address(this).balance > users[currentlyPaying].amount.mul(2)) {
                uint sendAmount = users[currentlyPaying].amount.mul(2);
                users[currentlyPaying].addr.transfer(sendAmount);
                totalPayout += sendAmount;
                currentlyPaying += 1;
        }
    }
}
```

Listing 2.3: Ponzi Contract in Solidity

Listing 2.3 shows a Ponzi Contract written in the Solidity programming language. The original contract can be found at: `https://github.com/alexroan/EthereumPonzi/blob/master/contracts/Doubler.sol`

As it is fairly easy to write a Ponzi scheme, they come in different shapes and sizes, but they all have one thing in common: All of them have a limited lifetime with an average of a little more than a year [BCCS20], and late investors always lose their money by participating. As Bartoletti et al. [BCCS20] have shown in their study, there are hundreds of Ponzi schemes on the Ethereum blockchain, which totaled thousands of payments. Most of the schemes today are abandoned and not used anymore, leaving late investors with losses, and scammers with illicitly obtained funds which they can then put in other schemes, contracts, or tokens of their choice.

2.3.4 Phishing

Another form of attack we have had to deal with for multiple years now, is password fishing ("phishing"). Described by its Wikipedia entry as "a type of social engineering where an attacker sends a fraudulent (e.g., spoofed, fake, or otherwise deceptive) message designed to trick a person into revealing sensitive information to the attacker".

There are many different approaches on how to fish for passwords or other sensitive information, and these certainly also tap into passwords for crypto wallets, as well as personal information related to cryptocurrency deposits. Social engineering, link manipulation, as well as phishing over various communication channels such as emails and various other messengers and online platforms, are among the most common phishing techniques. But the special properties of the Ethereum blockchain also allow for more novel phishing techniques, specifically tailored to SCs. The main goal of a phishing attack stays the same, whether it is orchestrated through an old-fashioned link manipulation or a little more sophisticated attack.

In the following two listings, we show, that by getting an entity to call a malicious function of a SCs, there is a possible quasi-phishing attack, specifically tailored to deployed SCs. The maliciously set up contract can act like a manipulated link, triggering functions of a contract in an unintended way. If entity A was the owner of the *Wallet* contract

(Listing 2.4), another entity B could trick A into calling the *transfer()* method of his own contract *Exploit* (Listing 2.5). This would then trigger the *transfer()* function of the *Wallet* contract, with the requirement, that the origin of the transaction (the original call that introduced the execution) was the owner, fulfilled. In this case the counterfeit function call uses the wallet owner´s identity to transfer all funds stored in the *Wallet* contract, to the malicious entity B. Fishing for passwords or other sensitive information of the victim is not even necessary. In this simplified scenario, it´s enough to trick the person into calling a set-up function. In newer versions of the Solidity programming language, there are better security measures in place, but this pseudo-phishing attack would still work for older contracts, and also for contracts with little or exploitable security measures.

```
pragma solidity ^0.8.13;

contract Wallet {
    address public owner;

    constructor() {
        owner = msg.sender;
    }

    function deposit() public payable {}

    function transfer(address payable _to, uint _amount) public {
        require(tx.origin == owner, "Only the owner can transfer funds.");
        _to.transfer(_amount);
    }

    function viewBalance() public view returns (uint) {
        return address(this).balance;
    }
}
```

Listing 2.4: Exploitable Wallet Contract

```
pragma solidity ^0.8.13;

abstract contract Wallet {
    function transfer(address payable to, uint amount) virtual public;
    function viewBalance() virtual public returns (uint);
}

contract Exploit {
    address payable public owner;
    Wallet wallet;

    constructor(Wallet _wallet) {
        owner = payable(msg.sender);
        wallet = Wallet(_wallet);
    }

    function transfer(address payable _to, uint _amount) public {
        // function parameters are just here to mimic the original call
        wallet.transfer(owner, wallet.viewBalance());
    }
}
```

Listing 2.5: Phishing Exploit Contract

2.4 Network analytics methods

Before we dive into our data and start investigating it, we take some time to accurately describe how our data is constructed into a graph or network, and what this means for our research.

The terms *network* and *graph* are mostly used interchangeably and describe some instances of connected entities. Whether the term network or graph is used, mostly depends on the context, and sticking to one choice comes with different terminology for the different parts that a network or graph can consist of. In this thesis, we will use both of the terms interchangeably, although some point out, that the term *network* is more frequently used in real-world settings, and the term graph is more often used to describe some abstract or mathematical representation of these networks [Bar16].

Networks come in different forms and sizes, and there are many types of networks in our everyday lives, such as transportation networks, power grids, or the internet. Even the human brain can be described as a network [New18]. A network or graph mainly consists of two different entities. The first one being *nodes* (also regarded to as *vertices*), and the second one being some type of connections either regarded to as *links*, *edges* or *relationships*. Formally the notation $G = (V, E)$ is used to describe the graph G which consists of a set of vertices V and a set of edges E. En edge e_{ij} is a connection between vertices v_i and v_j. For the context of our work we will stick with the terms of *nodes* and *relationships*, as Neo4j, the graph database we use for our research also makes use of these terms. With only these two notations an **undirected** graph can already be constructed.

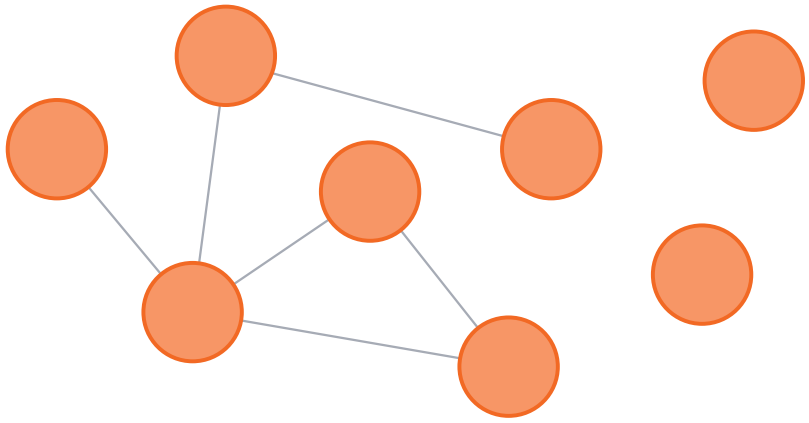


Figure 2.4: Undirected graph

The undirected graph is one of the simplest forms of graphs or networks and based on this notation we can further add restrictions as well as additions to the notation of simple undirected graphs, such that new, more complex forms, of graphs, can result.

To conduct our research, the notation of an undirected graph is enriched by *labels* and *properties*, and a notation for directed relationships is also present. Therefore the full graph database model consists of:

1. **Nodes**: Nodes describe some entity (e.g. an Ethereum address)
2. **Labels**: Labels describe a node more precisely. For example, every node could be an Ethereum address, but we could also add another label to some of the nodes, stating that these nodes are of type "ILLICIT".
3. **Relationships**: Relationships describe the connection between two nodes. In our graph database model relationships always have a relationship type, and all relationships are directed.
4. **Properties**: Properties are key-value pairs that are used to store further information inside of nodes.

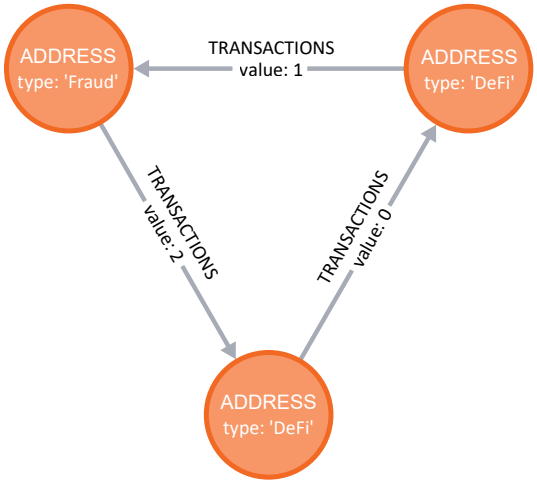


Figure 2.5: Directed graph with labels and properties

Figure 2.5 inhabits three nodes, that all have the label **ADDRESS**. Multiple labels for one node are also a possibility. All relationships are of type **TRANSACTIONS**, and are further defined by the property **value**. The nodes also have properties with key **type**. All properties can have different values, like for example "DeFi" or "Fraud".

Networks are of particular interest in many studies, as algorithms for community detection or dividing networks into distinct groups, provide helpful metrics when it comes to understanding the relationships that entities engage in certain domains.

There are a few metrics that are often used to describe networks and graphs to gain a better understanding of them:

1. **Degree**: The degree of a node describes the number of connections that a node has. In a directed graph we can differentiate between 2 types of degrees: The ingoing degree for a vertex v is formally denoted as $deg^-(v)$ and counts relationships that point *to* the node. The outgoing degree, which counts relationships that emerge *from* a vertex v is formally denoted as $deg^+(v)$.

2. **Farness:** The farness of a node is defined as the smallest sum of relationships that have to be traversed, in order to reach every other node in the network.

3. **Closeness**: Closeness describes how close a node is to all the other nodes in the network. Closeness is defined as the inverse of a node's farness.

4. **Betweenness**: The betweenness of a node is defined as the number of shortest paths of all other nodes that pass through the node.

5. **Clustering Coefficient**: The clustering coefficient of a node is defined by the number of paths between nodes that are directly connected to another node. This so-called neighbourhood for a vertex v_i is formally described as

$$N_i = \{v_j : e_{ij} \in E \vee e_{ji} \in E\}$$

The number of neighbours for the vertex v_i is defined by $k_i = |N_i|$ The number of connections between neighbours is defined by

$$c_i = |\{e_{jk} : v_i, v_k \in N_i, e_{jk} \in E\}|$$

If we divide the number of connections between neighbours of v_i by the maximum possible amount of connections between neighbours $k_i(k_i - 1)$ we get

$$C_i = \frac{|\{e_{jk} : v_i, v_k \in N_i, e_{jk} \in E\}|}{k_i(k_i - 1)} = \frac{c_i}{k_i(k_i - 1)}$$

for undirected graphs and

$$C_i = \frac{2c_i}{k_i(k_i - 1)}$$

for directed graphs.

Using networks as a base for representing data has lots of different benefits. By taking real-world data and presenting it in a graph structure, we are provided with the possibility to mathematically assess the properties of the given network. In our case, we will use the base metrics mentioned above as a foundation to take a deeper look into our network, and make certain statements about its composition, and also about how different nodes play different roles in the network.

CHAPTER 3

Data and Methods

In this section, we describe the data we used and the methods we applied in our analysis. We show how we gathered our data, and how we pre-processed it. We give a basic overview of our data collection process and the data we could gather. We also show how the metrics of our network data look like, to get a general understanding of the underlying network we investigate.

3.1 Dataset collection

To analyze illicit behaviour on the Ethereum blockchain, and the connection between DeFi protocols and illegal activities, we combine data from fraudulent activities as well as DeFi protocols with network data of the Ethereum blockchain. We conduct an unstructured online search and identify Ethereum addresses that were labeled as fraud or fraud-like in different categories. In addition to the fraudulent addresses, we also use a set of DeFi addresses, to identify different protocols. We then combine the two datasets with transaction data of the Ethereum network to obtain an observable network.

3.1.1 Fraudulent Address Data

In the following section, we list our unprocessed data sources, with links to their respective origin, as well as some additional information on how the lists were obtained, and when the sources were accessed. The data sources are ordered by their reputation and exposure from top to bottom, as we think the integrity of the data depends on how established the source is. This was mainly defined by how popular and well-known the provider (for example Etherscan) is. For reproducibility, we provide all the datafiles as well as the scripts for pre-processing and analyzing the files under our GitLab repository[1].

[1] https://gitlab.com/sebastian.luzian/defi-fraud-masterthesis/-/tree/main/data/malicious_addresses

The repository includes the following files (ordered by their assumed reputation from top to bottom):

1. **labelcloud.csv**: Accessed on 20.02.2022; Extracted Addresses from the Etherscan labelcloud[2]. We included all addresses, that were tagged with one of the following terms: Plus Token Scam, Scam, Exploit, bZx Exploit, Heist, Ponzi, Phish / Hack, Bitpoint Hack, Cryptopia Hack, EtherDelta Hack, Lendf.Me Hack and Upbit Hack. The addresses were extracted manually by calling `https://etherscan.io/accounts/label/[HYPHENATED-LABEL]?subcatid=undefined&size=10000&start=0&col=1&order=asc`, where [HYPHENATED-LABEL] should be replaced by the hyphenated versions of the terms mentioned above. In order to retrieve the full list of labels, we need to log in and set the size parameter to 10000 as there is no label with this many addresses. This guarantees we will get all accounts with the according label in one table. For example: `https://etherscan.io/accounts/label/phish-hack?subcatid=undefined&size=10000&start=0&col=1&order=asc` gives us all the Addresses labelled as "Phish / Hack". In Table 3.1 we can see the amount of found addresses and their category.

Category	Number of Addresses
Phish / Hack	5180
Upbit Hack	815
Heist	114
Exploit	86
Ponzi	50
Cryptopia Hack	6
bZx Exploit	5
Bitpoint Hack	2
Lendf.Me Hack	2
Plus Token Scam	2
Scam	2
EtherDelta Hack	1
Sum of Addresses	6265

Table 3.1: Addresses found in the Etherscan labelcloud

2. **etherscamdb_tagpack.yaml**: Provided by Complexity Science Hub Vienna (CSH) from CryptoScamDB[3]; Accessed on 20.02.2022; Last modified: 16.11.2021; Creator: INTERPOL CNTL.

   Unfortunately, this collection of addresses does not contain any information on the fraud category, and it also contains addresses belonging to other cryptocurrencies.

[2] `https://etherscan.io/labelcloud`
[3] `https://cryptoscamdb.org`

In total, the file contained 3526 Addresses, of which 3204 addresses were denoted as Ethereum addresses (see Table 3.3). As their category was not defined, all the found addresses are labelled with the category "null".

Category	Number of Addresses
null	3204
Sum of Addresses	3204

Table 3.2: Addresses found in the etherscamdb_tagpack.yaml file

3. **scams.yaml**: Accessed on 20.02.2022; A list maintained by EtherScamDB[4], an open-source database to keep track of all the current Ethereum scams. The list can be accessed via `https://github.com/MrLuit/EtherScamDB/blob/master/_data/scams.yaml`.

   Besides the address, this file also exhibits a category property. It also has some other properties we will not investigate further. In total, the file contains 6912 entries. 2696 entries have at least one address saved. From these 2696 entries, we find 3251 addresses. This is possible because one entry can save many addresses. Out of the 3251 addresses, 3015 are labeled as Ethereum addresses. The 3015 Ethereum addresses belong to the categories shown in Table 3.3.

Category	Number of Addresses
Scamming	2320
Phishing	689
Fake ICO	5
Scam	1
Sum of Addresses	3015

Table 3.3: Addresses found in the scams.yaml file

4. **addresses-darklist.json**: Accessed on 20.02.2022; A list maintained by MyEtherWallet[5], a free, client-side interface for the Ethereum blockchain. The list maintained by volunteers and can be accessed via `https://github.com/MyEtherWallet/ethereum-lists/blob/master/src/addresses/addresses-darklist.json`.

   The file does not exhibit a category property, nor does it show any property regarding the currency, therefore all the contained addresses are labeled with category null again, as seen in Table 3.4. In a later step, we will check if the addresses contained are actually Ethereum addresses, as - by looking at the properties of the file - we are not able to tell.

[4] `https://github.com/MrLuit/EtherScamDB`
[5] `https://www.myetherwallet.com/`

Category	Number of Addresses
null	715
Sum of Addresses	715

Table 3.4: Addresses found in the addresses-darklist.json file

5. **addresses.json**: Accessed on 08.03.2022; A big list maintained by CryptoScamDB[6]; an open-source database to track malicious URLs and their associated addresses. The list can be accessed via `https://api.cryptoscamdb.org/v1/addresses`.

   Besides the address, this file also exhibits a category property, and the distribution of the different categories can be seen in Table 3.5.

Category	Number of Addresses
Scamming	2978
Phishing	721
Fake ICO	5
Sum of Addresses	3704

Table 3.5: Addresses found in the addresses.json file

6. **urls.yaml**: Accessed on 20.02.2022; Another big list maintained by CryptoScamDB; The list can be accessed via `https://github.com/CryptoScamDB/blacklist/blob/master/data/urls.yaml`

   The urls.yaml file in total contains 9396 entries, of which 3382 entries contain at least one Ethereum address. From these 3382 entries, we find 3729 addresses labeled as Ethereum addresses. This is again possible because one entry can save many addresses. The 3729 Ethereum addresses and their categories are again shown in table 3.6

Category	Number of Addresses
Scamming	2980
Phishing	744
Fake ICO	5
Sum of Addresses	3729

Table 3.6: Addresses found in the urls.yaml file

[6] `https://cryptoscamdb.org`

7. **uris.yaml**: Accessed on 20.02.2022; A small list maintained by CryptoScamDB; Accessible via `https://github.com/CryptoScamDB/blacklist/tree/master/data/uris.yaml`.

This small file only contains 17 entries and is split into the following categories:

Category	Number of Addresses
Scamming	15
Phishing	1
Fake ICO	1
Sum of Addresses	17

Table 3.7: Addresses found in the uris.yaml file

In total, we were able to find 20649 Ethereum Addresses from 7 different sources.

Category	Number of addresses in source file							Total
	1	2	3	4	5	6	7	
Phish / Hack	5180	-	-	-	-	-	-	5180
Upbit Hack	815	-	-	-	-	-	-	815
null	-	3204	-	715	-	-	-	3919
Scamming	-	-	2320	-	2978	2980	15	8293
Heist	114	-	-	-	-	-	-	114
Exploit	86	-	-	-	-	-	-	86
Ponzi	50	-	-	-	-	-	-	50
Phishing	-	-	689	-	721	744	1	2155
bZx Exploit	5	-	-	-	-	-	-	5
Fake ICO	-	-	5	-	5	5	1	16
Cryptopia Hack	6	-	-	-	-	-	-	6
Scam	2	-	1	-	-	-	-	3
Bitpoint Hack	2	-	-	-	-	-	-	2
Lendf.Me Hack	2	-	-	-	-	-	-	2
Plus Token Scam	2	-	-	-	-	-	-	2
EtherDelta Hack	1	-	-	-	-	-	-	1
Sum of Addresses	6265	3204	3015	715	3704	3729	17	20649

Table 3.8: Found addresses by source file and category

Unfortunately, we cannot guarantee that all the found addresses were in fact connected to illicit behavior since most of the collections we accessed were built up by humans or organizations over time, and because of that, there will always be human errors involved. Additionally, there are inconsistencies between different providers, on what is classified as a scam, and what is not.

3.1.2 DeFi Protocols Address Data

Protocol	Type	Number of addresses
synthetix	derivatives	271
barnbridge	derivatives	40
nexus	derivatives	24
dydx	derivatives	38
hegic	derivatives	8
futureswap	derivatives	9
Sum		390
maker	lending	190
aave	lending	157
compound	lending	67
instadapp	lending	72
Sum		486
harvestfinance	assets	101
badger	assets	64
fei	assets	40
vesper	assets	44
convex	assets	22
renvm	assets	15
yearn	assets	3
Sum		289
curvefinance	dex	163
0x	dex	28
uniswap	dex	15
1inch	dex	15
sushiswap	dex	12
balancer	dex	9
Sum		242
Total		1407

Table 3.9: DeFi protocols by type

As we want to investigate the connection between illicit addresses and decentralized protocols, we use a second input file *protocols.csv*, which contains addresses of different DeFi protocols. The data of this file was already used by Kitzler et al. in the paper: "Disentangling Decentralized Finance (DeFi) Compositions"[KVSH21]. The single .csv file was obtained by aggregating all the given data into one file (the conversion script, as well as the files, can be accessed under the folder structure protocols[7]). After conversion from the given folder structure, the single file now contains 4 different properties: **type**, **protocol**, **address** and **label**. The **type** defines the type of protocol and can be 4

[7] https://gitlab.com/sebastian.luzian/defi-fraud-masterthesis/-/tree/main/data/protocols

different values: assets, derivatives, dex and lending. The **protocol** property describes the actual protocol name. The **address** is the associated Ethereum address of the protocol, and the **label** property gives some additional information on the protocol. The file contains 1407 different entries of DeFi addresses (shown in Table 3.9).

3.2 Data normalization

By aggregating the data of the different providers, we make sure that the initial data is non-redundant and consistent. As the data sources, as well as the data itself, are heavily heterogeneous, we use Python, a high-level general-purpose programming language, and some of its data-science libraries to structure and unify the found data.

To extract and aggregate metadata from the address files, we investigate which properties are included in the raw data files. We decide to include the most common properties: **category**, **reporter**, **url** and **description**, where the **category** describes what type of illicit behaviour the address was associated with. The **reporter** property describes the data's origin, stating who gathered the data. The **url** property provides a link to investigate for example phishing websites, that got the address listed, and the **description** property contains additional information on the entry. We also include the column **origin**, to denote from which file the entry originally came from.

If one of the metadata fields was not present in one of the data files, we left the field blank. In total, the different files contained 20649 Ethereum addresses, of which 20181 were valid Ethereum addresses. We denoted an address as valid if one of the following two criteria was met:

- The address matches the regex pattern ^(0x[a-fA-F0-9]{40})$
- The address matches the regex pattern ^[A-Za-z0-9+/]{27}=+$

The first regular expression matches the representation of a regular Ethereum address in the following format: **0xb794F5279a39494cE8396eA0bfffBA7413579268**

The second regular expression matches a base64 representation of Ethereum addresses. For example: **16FVN/p2nfsR+9cWRidV6ZVZf4m=**

By decoding the address, converting it to hexadecimal and prepending "0x" to the converted string, we get a regular valid Ethereum address like so:

```
>>>import base64
>>>"0x"+base64.b64decode("16FVNrp2nfsRH9cWRidV6ZVZf4m=").hex()
'0xd7a15536ba769dfb111fd716462755e995597f89'
```

After aggregating all the addresses into one file *addresses.csv*[8], We further reduced the extracted addresses to 6869 addresses using another pre-processing script which eliminates

[8] https://gitlab.com/sebastian.luzian/defi-fraud-masterthesis/-/blob/main/data/malicious_addresses/output/addresses.csv

duplicates. In the elimination process, we overrode less-reliable sources with the more reliable metadata that was associated with the addresses. The found addresses with their metadata will comprise our ground truth dataset of illicit Ethereum addresses. The file is saved as *unique_addresses.csv*[9].

Category	Number of addresses in source file							Sum		
	1	2	3	4	5	6	7	Initial	Valid	Unique
Phish / Hack	5180	-	-	-	-	-	-	5180	5180	5172
Upbit Hack	815	-	-	-	-	-	-	815	815	814
null	-	3204	-	715	-	-	-	3919	3681	393
Scamming	-	-	2320	-	2978	2980	15	8293	8136	244
Heist	114	-	-	-	-	-	-	114	114	70
Exploit	86	-	-	-	-	-	-	86	86	86
Ponzi	50	-	-	-	-	-	-	50	50	50
Phishing	-	-	689	-	721	744	1	2155	2083	26
bZx Exploit	5	-	-	-	-	-	-	5	5	5
Fake ICO	-	-	5	-	5	5	1	16	15	-
Cryptopia Hack	6	-	-	-	-	-	-	6	6	3
Scam	2	-	1	-	-	-	-	3	3	2
Bitpoint Hack	2	-	-	-	-	-	-	2	2	-
Lendf.Me Hack	2	-	-	-	-	-	-	2	2	1
Plus Token Scam	2	-	-	-	-	-	-	2	2	2
EtherDelta Hack	1	-	-	-	-	-	-	1	1	1
Sum of Addresses	6265	3204	3015	715	3704	3729	17	20649	20181	6869

Table 3.10: Found addresses by source file and category

All the steps from above were done using the files and the python script in the folder structure *data/malicious_addresses*[10]. There are 7 different commands responsible for converting the 7 files with malicious addresses into one .csv-file with the above-mentioned properties. Table 3.11 shows the different commands with which we pre-processed the found addresses data.

[9] https://gitlab.com/sebastian.luzian/defi-fraud-masterthesis/-/blob/main/data/malicious_addresses/output/unique_addresses.csv

[10] https://gitlab.com/sebastian.luzian/defi-fraud-masterthesis/-/tree/main/data/malicious_addresses

Command	Input file	Output file
convert-labelcloud-csv	labelcloud.csv	addresses.csv
convert-etherscamdb-tagpack-yaml	etherscamdb_tagpack.yaml	addresses.csv
convert-scams-yaml	scams.yaml	addresses.csv
convert-addresses-darklist-json	addresses-darklist.json	addresses.csv
convert-addresses-json	addresses.json	addresses.csv
convert-urls-yaml	urls.yaml	addresses.csv
convert-uris-yaml	uris.yaml	addresses.csv
convert-all	(all of the above)	addresses.csv
uniqueify	addresses.csv	unique_addresses.csv

Table 3.11: Pre-processing Commands

3.3 Network construction

In addition to the data describing the fraudulent addresses and DeFi protocols, we also have multiple files responsible for constructing the network in Neo4j. These files contain the aggregated transaction data of the Ethereum transaction history. We use our data of DeFi protocols and fraudulent addresses, to enrich the files describing the aggregated network data. The files, as well as the script to enrich the data, can be found under the folder structure *network/input*[11].

By merging the network's data with the data of fraudulent addresses found in our web search (see Table 3.10), we find that only 5807 out of 6869 fraudulent addresses are actually found in the data. 1062 addresses could not be found in the aggregated transaction data. While further enriching the network with the data of DeFi protocols we can only identify 1100 addresses in the network data, meaning that out of the 1407 DeFi protocols we had in our initial dataset (see Table 3.9), 307 of the DeFi protocols could not be found in the networks data.

Apparently, some of the addresses we had in our initial dataset did not send or receive any funds, and therefore some of them are not contained in the aggregated network data. This leads to an overall smaller number of fraudulent entities as well as a smaller number of DeFi protocols contained in the network. Still, the number of found fraudulent addresses is a 120% increase, compared to a similar approach published in 2021 where out of 3559 addresses in total only 2628 could be found on the blockchain[LBB+21].

Also, we have multiple files responsible for tracking message traces, which we use to construct a separate graph to closer investigate message traces sent to the DeFi protocol most used by fraudulent entities. The files responsible for the message trace data lie under the folder structure *traces/input/malicious_traces*[12].

[11] https://gitlab.com/sebastian.luzian/defi-fraud-masterthesis/-/tree/main/data/network/input

[12] https://gitlab.com/sebastian.luzian/defi-fraud-masterthesis/-/tree/main/data/traces/input/malicious_traces

In order to visualize our data, we construct a graph database with Neo4j[13], a well-known graph data platform. The data we import is split into two sets of data:

- Files describing Ethereum addresses and their metadata (see Table 3.12)
- Files describing address relations and their properties (see Table 3.13)

Node properties	Property description
address_id:ID(Address)	Internal ID that identifies a node uniquely
address	Ethereum address describing this node
first_tx_id	Internal ID of the first transaction this node made
no_incoming_txs:int	Number of transactions this node received
no_outgoing_txs:int	Number of transactions this node sent
in_degree	$deg^-(v)$: Number of relationships that point to the node
out_degree	$deg^+(v)$: Number of relationships that point away from the node
is_fraud_address	Property defining an address as fraudulent or not
is_defi_protocol	Property defining an address as DeFi protocol or not
fa_category	The fraud category of a node
fa_reporter	The reporter who labelled the node as fraudulent
fa_url	A URL associated to the fraud
fa_description	A description of the fraudulent entity
fa_origin	The file of wich the fraud address was initially imported
p_label	The label of the DeFi protocol
p_type	The type of the DeFi protocol
p_protocol	The DeFi protocols name

Table 3.12: Node properties describing the Ethereum addresses in the network

Relationship properties	Property description
:START_ID(Address)	Internal ID that identifies the relationships start
:END_ID(Address)	Internal ID that identifies the relationships end
value:float	Total value associated with the transactions between the two nodes
no_transactions:int	Number of transactions aggregated in this relationship

Table 3.13: Relationship properties describing the aggregated transaction data

The Ethereum addresses imported to the graph database are considered as nodes, and the aggregated transactions between the addresses (the address relations) are considered as directional relationships between the nodes. For this, the two properties of type *:START_ID* and *:END_ID* are used, which reference the *address_id* property of the nodes which sent or received transactions. In Table 3.12 the prepended *fa* and *p* identifiers are set for readability. The (*fa*) identifier is only set if the node is a fraudulent address. Likewise, the *p* identifier is only set if the address is considered a DeFi protocol. If values

[13] https://neo4j.com/

are assigned to the prepended properties depends on whether the protocol is a fraudulent entity (*is_fraud_address* == True) or if the address is a DeFi protocol (*is_defi_protocol* == True) respectively.

The network's data is then imported to Neo4j via the neo4j-admin tool [14]. In total, we import 148.436.284 nodes, 463.089.050 relationships and 1.816.816.820 properties for the transaction network.

Protocol	Type	Percentage (%)	Count
synthetix	derivatives	24	264
barnbridge	derivatives	3	32
nexus	derivatives	2	18
dydx	derivatives	2	17
hegic	derivatives	1	8
futureswap	derivatives	1	7
Sum		31	346
maker	lending	15	163
aave	lending	7	80
compound	lending	4	46
instadapp	lending	1	8
Sum		27	297
harvestfinance	assets	9	101
badger	assets	4	42
fei	assets	3	29
vesper	assets	3	29
convex	assets	2	18
renvm	assets	1	15
yearn	assets	0	3
Sum		22	237
curvefinance	dex	14	153
0x	dex	2	22
uniswap	dex	1	15
1inch	dex	1	12
sushiswap	dex	1	12
balancer	dex	1	6
Sum		20	220
Total			1100

Table 3.14: Protocols by type

[14] https://neo4j.com/docs/operations-manual/current/tools/neo4j-admin/

Fraud Category	Percentage (%)	Count
Phish / Hack	73.0	4249
Upbit Hack	14.02	814
null	5.77	325
Scamming	3.12	181
Exploit	1.45	84
Heist	1.19	69
Ponzi	0.84	49
Phishing	0.38	22
bZx Exploit	0.09	5
Cryptopia Hack	0.05	3
Scam	0.03	2
Plus Token Scam	0.03	2
Lendf.Me Hack	0.02	1
EtherDelta Hack	0.02	1
Sum		5807

Table 3.15: Frauds by category

Looking at the protocol data in Table 3.14, we can state that 31% of protocols in the network are labelled as derivatives, with Synthetix having the most protocol addresses, amounting to about 24% of all DeFi protocols contained in the dataset. The next biggest protocol types are lending protocols (27%) and protocols labelled as assets (22%). DEX protocols come in last amounting to 20% of all protocols. Most of the protocols labelled as DEXs actually only have very few addresses, with Curvefinance being the exception with 153 addresses. Looking at Table 3.15 we can see that out of all the illicit addresses in the network, 4239 addresses (73%) were labelled as "Phish / Hack". This is by far the biggest category of illicit activity we were able to find, as the second biggest listing "Upbit Hack" only sums up to about 14% of all findings. The next biggest category is the "null" category which contains all the uncategorized addresses. Based on these numbers we either think, that phishes and hacks are the most conducted types of illicit activities, or that other categories of fraud do not get reported as often as it could be harder to track down involved addresses or to even identify their illicit behaviour in the first place.

3.4 Topology analysis

With the properties of the DeFi addresses, as well as the fraudulent addresses imported to the graph database, we provide some insight into the network's topology. We are specifically interested in how the fraudulent addresses are connected to each other, as well as how they are connected to DeFi protocols. For this we look at a few different topology metrics, using the Cypher Query Language (CQL) suited for extracting data from the graph database:

1. In and Out Degrees of different fraud categories
2. In and Out Degrees of different DeFi protocols
3. Amount of fraudulent addresses directly connected to DeFi protocols
4. Percentage of fraudulent neighbours of fraudulent addresses
5. Local clustering coefficients for fraud addresses and DeFi addresses

As our first step, we show the different ingoing degrees of nodes grouped by different fraud categories, in order to get a macro perspective of how fraudulent addresses are connected and how they receive funds. We do the same for DeFi protocols to get a better understanding of DeFi usage and how often DeFi protocols and different protocol types are used. We then interpret the data and try to give an explanation for differences between certain fraud categories as well as DeFi protocols.

In the next step, we expand our focus to see how many fraud addresses are connected to DeFi protocols and to what extent. We also show the distribution of fraudulent neighbours per fraud category and explain key differences between different fraud categories.

For the last step in our network study, we calculate local clustering coefficients for all the fraudulent addresses, to not only see how many neighbours of fraudulent addresses are also fraudulent, but also to extract how many of the addresses neighbours are also connected to each other. The local clustering coefficient is calculated by dividing the number of connections between the neighbours of a node by the overall possible number of connections between neighbours. This is done by extracting the count of neighbours for every node we want to investigate and then checking the number of connections between those neighbours in a separate step. As this basically involves triangle-counting, it can take some time until the calculation is complete for a network our size, so we calculated this step in Python to get some feedback on the progress, as this was not possible in Neo4j.

CHAPTER 4

Analysis & Results

In this section, we show the results we found in our research. In a topology analysis, we will give an overview of the network's structure and the connectedness of fraudulent nodes and DeFi nodes. In a transaction analysis, we show how fraudulent entities move illicitly obtained funds in the Ethereum network. In a more detailed trace analysis, we then look at the DEX Uniswap, which receives illicit funds particularly often. By examining message traces, we then show which ERC-20 tokens are preferentially used by fraudulent addresses and how transactions within the protocol take place.

4.1 Fraudulent activities and their connection to DeFi

To give an overview of how fraudulent addresses are connected to DeFi protocols, Figure 4.1 shows all DeFi nodes, as well as all fraudulent nodes which have direct connections to DeFi nodes. The label sizes depend on the amount of value that was transferred to the protocols. The graph shows 734 nodes, of which 123 (16,76%) are DeFi protocols and 613 (83,51%) are fraudulent addresses[1]. The edge colour depends on the color assigned to a specific fraud category, shown in the legend of the plot. We can see that big parts of the graph are colored in pink, as the category "Phish / Hack" makes up 76.18% of all nodes in the network, and nodes of this category apparently also have quite some connections to DeFi protocols. We can as well see that the node labelled "UniswapV2Router02" receives the largest amount of funds. As the network only contains nodes with direct connections to DeFi, percentages of frauds shown in the graph deviate slightly from the before-mentioned total distributions, as fraudulent nodes with no connection to DeFi nodes are excluded.

In Figures 4.2 and 4.3 we show how fraudulent entities are connected to other nodes. Here we do not specifically look at the connection to DeFi nodes, but show which types

[1]Note: Two adresses were labelled as DeFi and as fraud

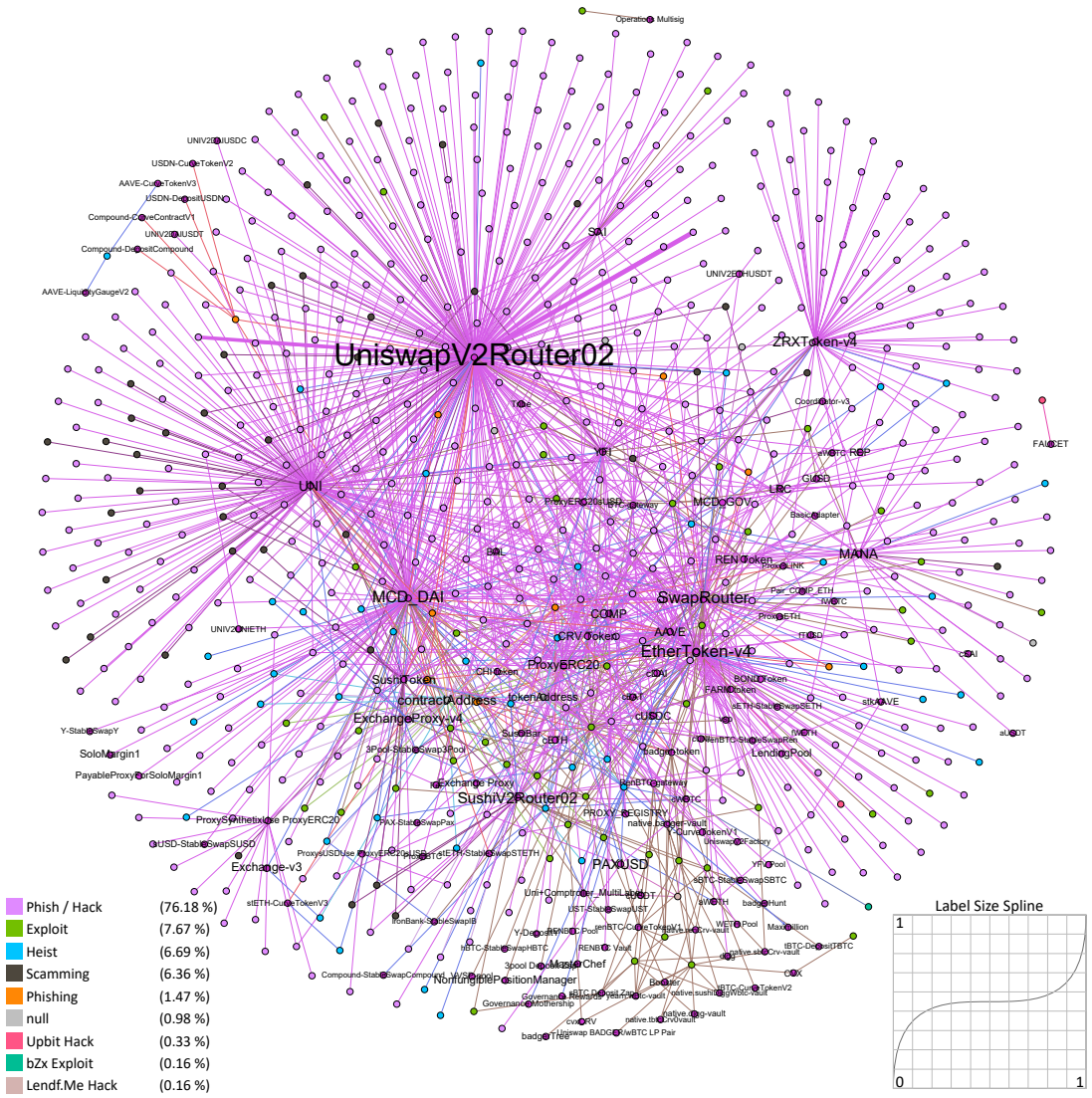


Figure 4.1: Connections from fraudulent addresses to DeFi protocols

of fraud are well-connected to other nodes. We therefore show the average of ingoing degrees per fraud category by calculating $\sum_v deg^-(v)$ where v describes all fraudulent nodes of a specific category. we can see, that although we had a lot of addresses labelled as "Phish / Hack", the addresses in this category do not have a very high amount of ingoing transactions, with an average in-degree of 31.2. The biggest average of ingoing transactions was measured on the two addresses labelled as "Plus Token Scam". As the category shrank all the other entries significantly, due to the small number of addresses it contains, in Figure 4.3 we excluded the "Plus Token Scam" category to get a clearer picture.

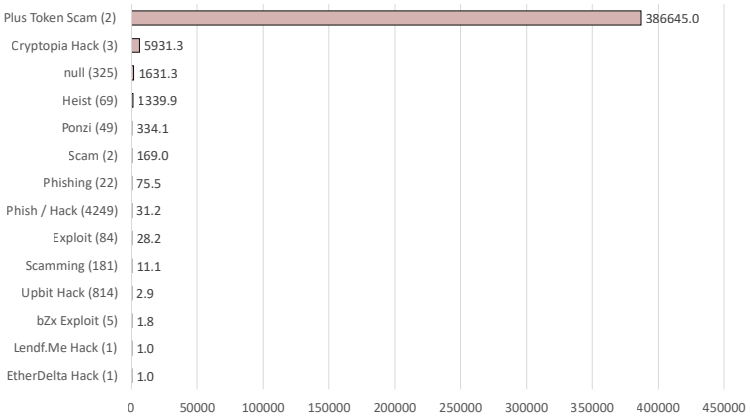


Figure 4.2: Average of in_degree per fraud category

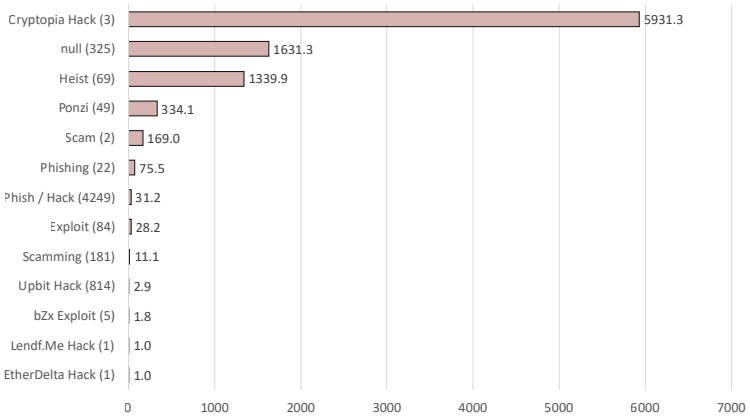


Figure 4.3: Average of ingoing degrees per fraud category (Plus Token Scam excluded)

To understand which protocol types and protocols are regularly used, in Figure 4.4 and 4.5 we investigate the protocols connections. We find that DEXs have the highest amount of ingoing transactions, followed by lending protocols, assets and derivatives (shown in Figure 4.4). Looking at the different protocols, we find that Uniswap addresses are particularly often used. On average Uniswap addresses have 3 times as many ingoing transactions as any other DeFi protocol we investigated (see Figure 4.5). The second and third places are also taken by DEXs: 0x and Sushiswap.

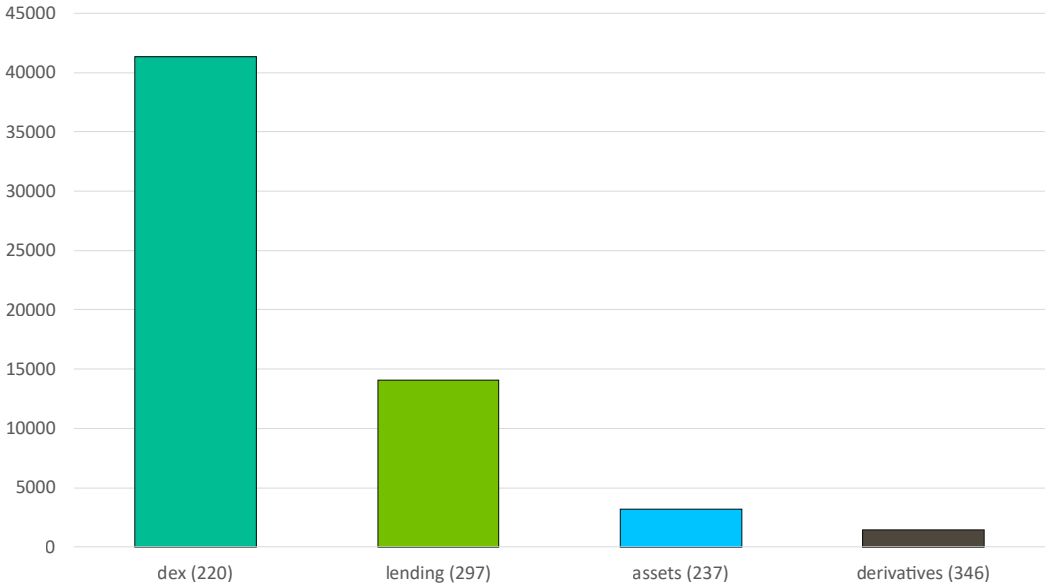


Figure 4.4: Average of in_degree per protocol type

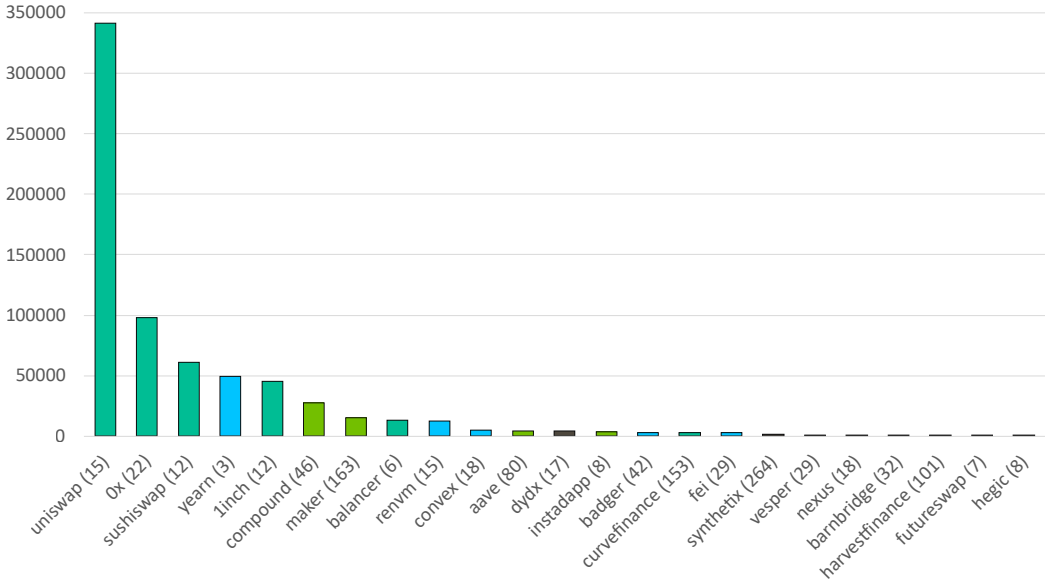


Figure 4.5: Average of in_degree per protocol

We can already see a tendency of usage here, pointing towards protocols which are DEXs. In the following, it will be interesting to see if this connectedness also means that illicitly obtained funds are also more likely to flow to DEXs, or if fraudulent entities stand out from the crowd and are more likely to choose other DeFi protocols or less-used protocols.

To find out if this tendency towards DEXs also holds for fraudulent entities, we filter the connections to now only include fraudulent nodes as well as DeFi nodes. We look at the connections between fraudulent addresses and DeFi protocols and see that in fact, DEXs are the most-common destination for fraudulent funds as well.

In Figure 4.6 we show the number of fraudulent addresses directly connected to DeFi protocols. Out of the 5807 fraudulent addresses we were able to find, 1479 (more than 25%) were directly connected to DeFi protocols. Out of these 1479 addresses, most addresses are connected to DEXs (almost 60%). Fraudulent addresses are particularly often connected to the DEX Uniswap (483 fraudulent addresses have a direct connection to Uniswap). 355 addresses are directly connected to lending protocols which amounts to about 23%. Addresses belonging to asset- or derivative-related DeFi protocols are connected to 171 and 77 fraudulent addresses (about 11% and 5% respectively).

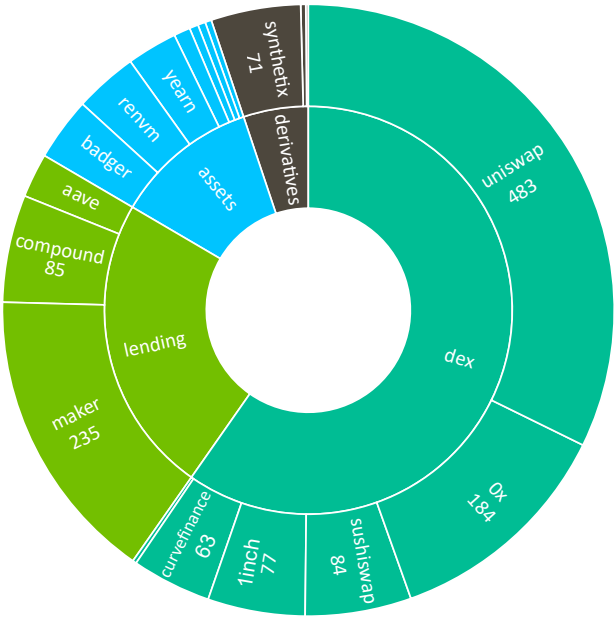


Figure 4.6: Amount of fraudulent addresses directly connected to DeFi protocols

In Figure 4.7 we show the distribution of fraudulent neighbours per fraud category, to see if certain fraudulent activities show high interaction between the nodes conducting them. We find, that most of the addresses in the different fraud categories do not have any fraudulent neighbours or only a small percentage of all neighbours are labelled as fraudulent (most addresses have $< 10\%$ fraudulent neighbours). For most fraudulent activities, this leads us to believe that there are many separate entities conducting these and that most of the entities act on their own or in rather small groups. Addresses that were related to the Upbit Hack show a different picture. A lot of addresses related to this hack have a very high percentage of neighbours, which are also labelled as fraudulent.

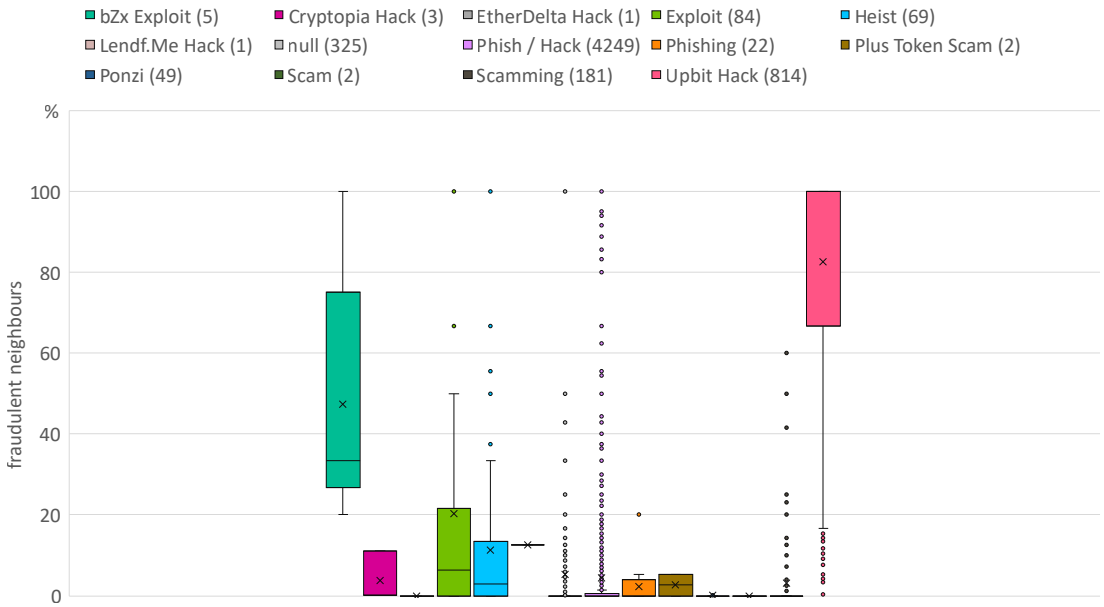


Figure 4.7: Percentage of fraudulent neighbours of fraudulent addresses

We find that 534 of the addresses tagged as *Upbit Hack* have only neighbours that are also labelled fraudulent. This presumably shows the efforts of the hackers, to obfuscate the stolen funds in an untraceable transaction chain, which possibly enabled the attackers to cash out - at least parts of - the stolen funds. Some sources although suggest that exchanges are working proactively to freeze the associated funds [Blo20]. Therefore, even if parts of the funds reached exchanges, we cannot certainly state if they were also successfully converted to fiat. Looking at the data, we also observe that addresses labelled as *Phish / Hack* have a very high number of addresses (about 3142 of 4249 = 73%), that do not have any fraudulent addresses in their neighbourhood. This could either indicate, that the received funds are instantly obfuscated in a way, such that most of the reporting sites lose track of the transactions immediately, or that these activities often get reported and uncovered before the fraudulent address can actually obtain any funds. As different sources state that phishing has interaction rates of about 20% to 30% [Cor20] [BGL16] [JJJM07], we think this corresponds nicely with the number of addresses labelled as Phish / Hack, that do not have any fraudulent neighbours (although the interaction rates of phishing can differ significantly depending on the means by which the attack is delivered [BCAZ20] [JJJM07]). The logical conclusion we want to state here is that addresses in the *Phish / Hack* category may not receive funds very regularly, due to the low success rate of phishing scams. Transfers in this category could therefore be rare altogether because most of the addresses will never receive funds from victims, and therefore will never have any funds to move. But it could also indicate the use of dedicated collector addresses (addresses which are used to collect the obtained funds). In this case, other addresses participating in a fraudulent scheme would only need to

transfer funds to a single collector address, therefore keeping the percentage of fraudulent neighbours low. All together we can state, that for most fraudulent activities, we do not find a high percentage of fraudulent neighbours, except when a single entity uses multiple addresses in an attack like in the Upbit Hack.

In the next step we calculate local clustering coefficients for all fraud addresses, to not only see how many neighbours of fraudulent addresses are also fraudulent but also to extract how many of the addresses neighbours are also connected to each other. In Figure 4.8 we show the distributions of clustering coefficients per fraud category. Overall, most of the clustering coefficients for the different fraud categories are very low and do not exceed the 10% mark, meaning that only every fifth neighbour would be connected to another neighbour. Exceptions are the unlabelled category (*null*) and the *Upbit Hack* category. Addresses labelled as *Upbit Hack* do not only have the highest percentage of fraudulent neighbours as we have seen before but also the neighbours have a rather large amount of connection between them, although they are still far from being fully connected. This is partly due to the fact, that most nodes are only connected in one way, thus resulting in overall lower clustering coefficients. As we found the addresses in the *null* category to have a low percentage of fraudulent neighbours, it is somewhat surprising, that the neighbours of these addresses are apparently connected to each other in at least some cases. This could show an indication that fraudulent addresses in the set of uncategorized activities are at least partly related to each other.

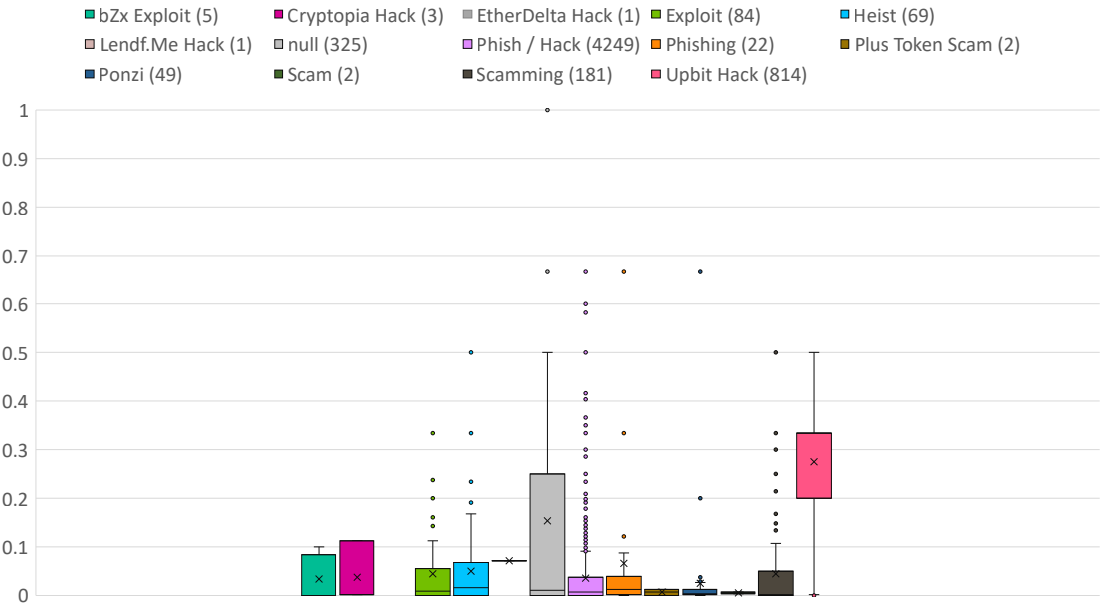


Figure 4.8: Clustering coefficient per fraud category

We have now seen a general overview of the network's topology, as well as how addresses are connected to DeFi protocols, and how illicit addresses, as well as their neighbors, stand

in relationship to each other. Summing up our findings we can state that decentralized exchanges are popular targets for fraudulent funds. Uniswap in particular receives large amounts of funds coming from illicit activities. We have also seen that most of the fraudulent activities do not involve many connected entities, and most of the activities are probably conducted by only a few addresses.

In the next section, we will further present not only the structural aspects of how addresses are connected, but also highlight how much value from different illicit activities actually flows to DeFi protocols, and what happens with these funds in more detail.

4.2 Analysis of transactions and transaction activity

In this section of the results, we look at the macro picture of the network, and check where funds are flowing, in one hop, from fraudulent entities to DeFi addresses. For this, the same network data that was used to conduct the network study, which is shown in tables 3.12 and 3.13 is used. The transaction analysis is conducted by extracting all the transactions of the network, that happened between addresses labelled as fraudulent, and addresses which were labelled as DeFi protocols. We first investigate the overall transaction value that went from fraudulent addresses to addresses marked as DeFi, and group the values by fraud categories as well as protocol types and protocols. In a further step, we analyse the transaction activity over time and give an insight into how fraudulent activities have developed over time.

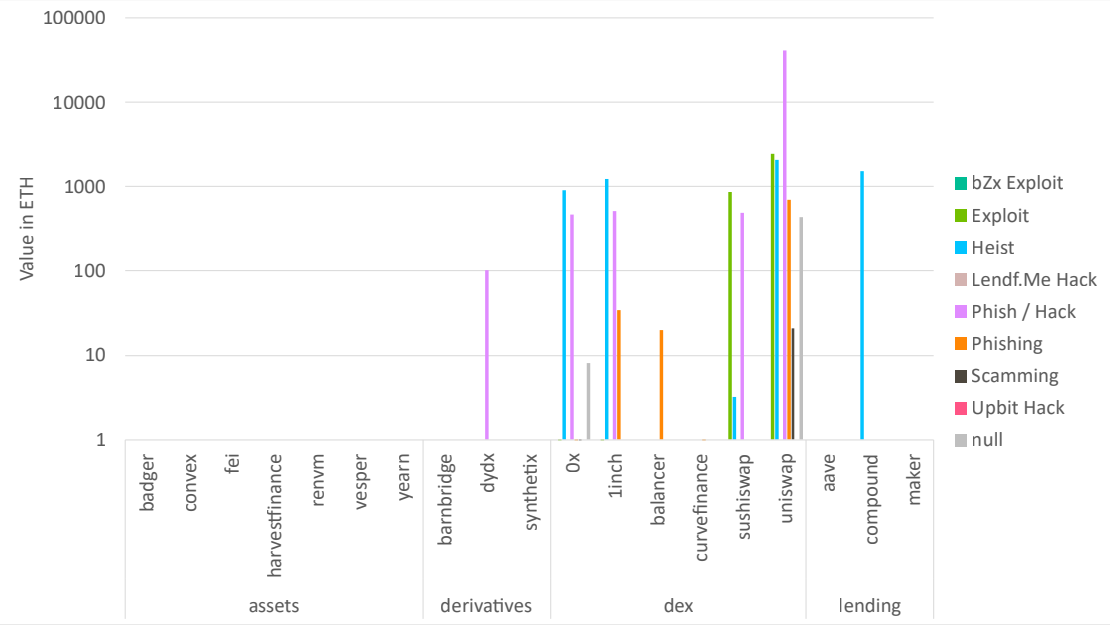


Figure 4.9: Flows from fraud addresses to DeFi per category on logarithmic scale

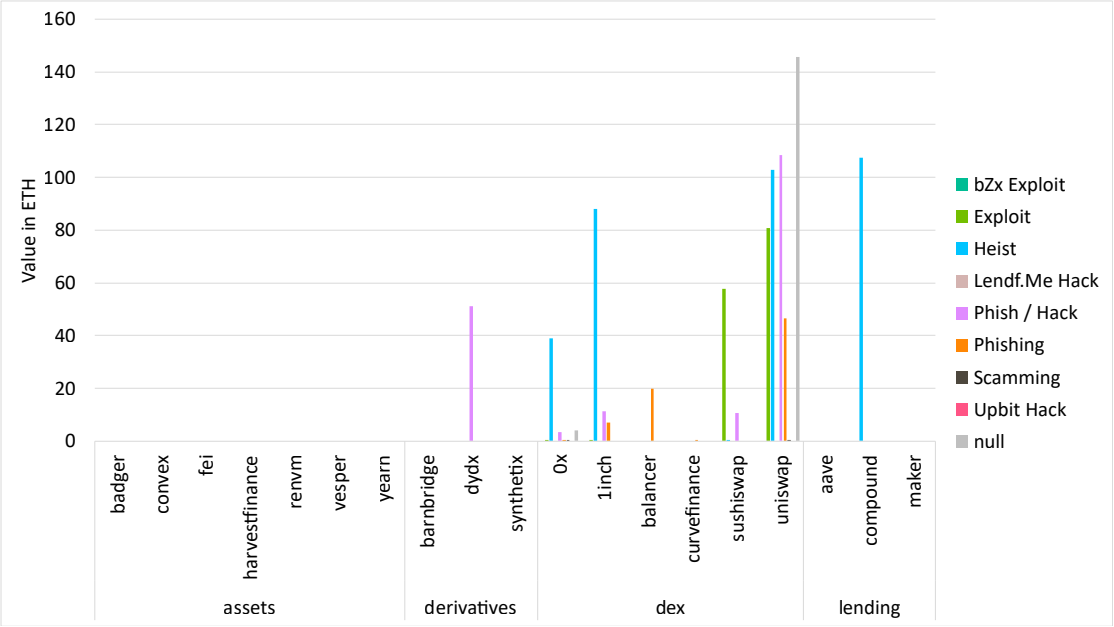


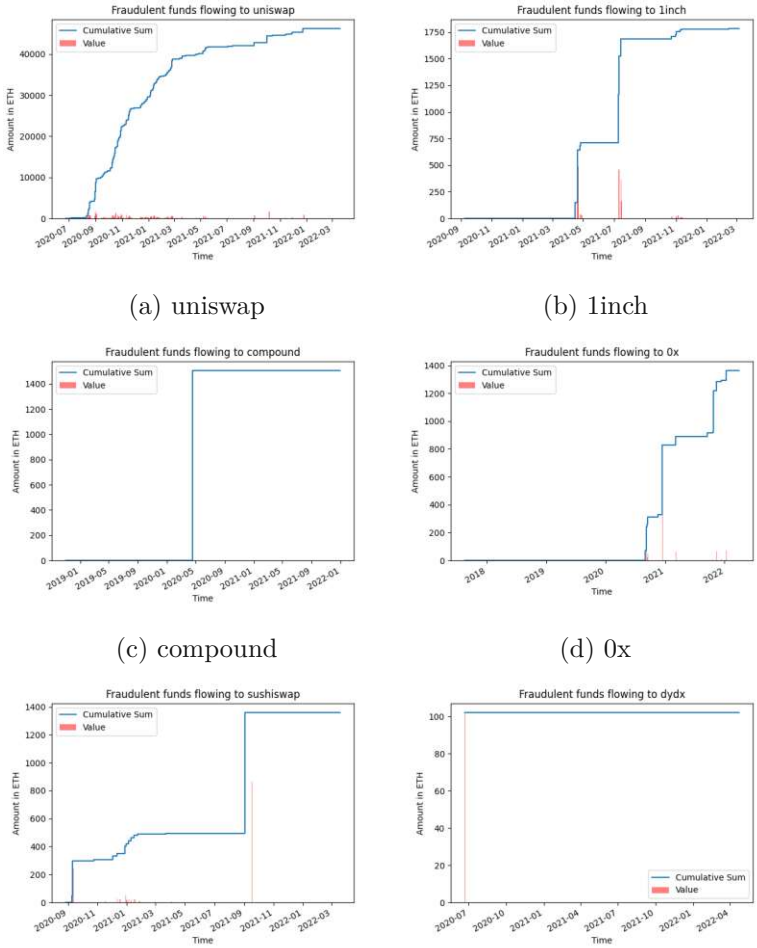
Figure 4.10: Average value per transaction flowing to DeFi protocols by fraud category

In Figure 4.9 we show the cumulative sums of funds flowing from different fraudulent activities to DeFi protocols. From the figure, we see that the DEX Uniswap received the majority of all funds which were sent to DeFi protocols. In total Uniswap received 46215.0 ETH from fraudulent activities. 2427.3 ETH from fraudulent activities labelled as exploits, 2058.9 from heists, 40574.3 ETH from phishes or hacks, 696.3 ETH from phishing, 21.0 from scamming and 437.3 ETH in the aftermath of the Upbit Hack. It comes as no surprise, that the biggest category "Phish / Hack" also accumulated the most value flowing to DeFi protocols, as it just includes so many fraudulent addresses. All other protocols are not targeted this heavily, 1inch with 1779.7 ETH being the next biggest destination of fraudulent funds. After 1inch, the lending protocol compound makes the third spot with a received amount of 1505.3 ETH. Places 4 and 5 are DEXs 0x and Sushiswap with 1362.6 ETH and 1358.7 ETH. In place 6 is derivatives protocol dYdX with a received amount of 102 ETH. Place 7 and 8 are also 2 DEXs, balancer and curvefinance, although they only received 20 and 0.01 ETH respectively.

In conclusion, we can state that all of the DEXs we had in our dataset received funds coming from fraudulent addresses. Protocols, that were labelled as assets did not receive any funds that came from fraudulent entities. Apparently, DeFi protocols labelled as assets are not used in a way that would need illicit entities to transfer funds. As we have seen before, 10% of all fraudulent addresses have a direct connection to the assets sector, so we assume that there are ways in which these entities can still make use of these protocols, but they do so without transferring any value in the transactions. In the sector of derivatives, only protocol dYdX received some funds, and in the lending sector, only the compound protocol received funds. We can clearly see a tendency that

DEXs (probably due to their ability to exchange tokens for fiat and other currencies) are heavily addressed by fraudulent entities.

For our next investigation, we show how this activity evolved over time. In Figure 4.11 we can see the different DeFi protocols, that received significant funds in the past years. We can see that most of the activity started to trend up around the end of 2020. With Uniswap (see figure 4.11a) and 1inch (see figure 4.11b) being the two DeFi protocols that received the most funds, we see that Uniswap was addressed quite regularly, but then in 2021, the fraudulent funds started declining. This could be partly caused by the fact, that Uniswap has been updated to a newer version and the addresses changed in the meantime. Nevertheless, we can see, that Uniswap is the most regularly addressed destination of fraudulent funds.



(a) uniswap (b) 1inch

(c) compound (d) 0x

(e) sushiswap (f) dydx

Figure 4.11: Transaction activity over time for different DeFi protocols.

4.3 Analysis of transaction schemes

After the investigation of transactions between DeFi protocols and fraudulent addresses, we we now want to answer our last research question "In which ways are illicitly obtained funds being transferred to DeFi protocols?". For this matter, we want to focus on the most targeted DeFi protocol, Uniswap, and check what exactly happens to the tokens that were transferred to it.

On the Ethereum blockchain, the initial transaction and its data are enough to specify how the state will change after executing it. But if we want to investigate what happens in the course of a transaction, and give a semantic meaning to it, we need another dataset, a dataset of message traces, that helps us accomplish this. As message traces are not stored on the blockchain, they need to be extracted by recording all the EVM calls that were made in the course of a transaction. For this, a modified EVM is used to re-execute all transactions that were stored on the blockchain, with their associated data. On this modified EVM all calls that are made during execution are stored. By looking at the message traces that occurred between addresses that are marked as fraudulent or DeFi, we can then check what functions were called on transaction execution. This is in contrast to our first dataset, where we could only tell where transacted value was flowing to, but make no statement about what exactly happened with the value that was transferred. In Table 4.1 we can see all the properties that our trace data contains.

Trace properties	Property description
transaction_index	Internal ID to uniquely identify a transaction
from_address	The address from which the message call originated
to_address	The address to which the message call was made
value	The value associated with this message call
input	The input data associated with this message call
trace_type	the type of trace either call or create
call_type	The call type of the trace either call, staticcall or delegatecall
subtraces	How many subtraces the trace consists of
trace_address	Unique identifier for a trace inside of a transaction
error	Information if the call failed and in which way
status	1 if the transaction was executed successful, 0 otherwise
transaction_hash	Unique hash identifying the transaction

Table 4.1: Trace properties

As the trace data does not exhibit any human-readable information on what the different message calls do, we use the Ethereum Signature Database[2] to extract function signatures in a human-readable form. After we extracted all the possible signatures that might potentially match the made calls, we further enrich the data with descriptions for the from- and to-addresses, in order to better understand the trace. Table 4.2 shows the

[2] https://www.4byte.directory/

properties that are additionally added to the trace data in order to make meaningful statements about the exact execution of a trace.

Additional trace properties	Property description
from_desc	A text description for the from_address property
to_desc	A text description for the to_address property
hex_signature	The hexadecimal function signature extracted from the input data
text_signature	The textual representation of the function signature

Table 4.2: Additional trace properties

The full trace analysis consists of the below steps, again first giving a general overview and then going into more detail:

1. We check which addresses of the Uniswap protocol are targeted, and how much value the different addresses receive.
2. By checking the lengths of all traces, we investigate different patterns and see how the behaviour of fraudulent entities changes, depending on how much value is involved in a transaction.
3. By doing a comparison of the string representation of all the traces, we identify the most common actions that fraudulent entities take when interacting with Uniswap. We also show which trace patterns are responsible for the biggest value transfers.
4. We check which ERC-20 compatible token addresses occur inside of the traces to make statements about how value is swapped into tokens, and which tokens are particularly popular among fraudulent entities.
5. In the last step we look at ERC-20 compatible tokens and check which tokens receive the most funds, to make a statement about how illicitly obtained money is utilized.

Before we start analysing the actual traces, we take a look at which addresses of the Uniswap protocol actually received any funds. We therefore take the traces and look at the initial message call. As Uniswap isn't only a single Smart Contract, but a number of different contracts, we wanted to see where value was initially going, and find, that the two router contracts of Uniswap are the primary addresses where funds are sent to. Uniswap's version 2 router received 7543 message calls coming from addresses contained in our fraudulent set, delivering over 41493.2 ETH over the last years. The newer version of the router received 244 message calls and about 2979.0 ETH from fraudulent addresses respectively, meaning that the average sent value of fraudulent entities has increased from about 5.96 ETH per transaction to about 13.00 ETH per transaction, during the changeover from version 2 to version 3. We also note that of the 7543 message calls going to the version 2 router of Uniswap, 583 message calls were erroneous, and therefore

a value of 1480.7 ETH was not able to successfully reach its destination. Of the 244 transactions that went to the version 3 router, 15 were erroneous and 262.2 ETH was not sent successfully. In total 1742.9 ETH that was meant to be transferred did not end up on the Uniswap protocol, giving us a total received value of 44472.2 ETH (see Figures 4.12, 4.13 and Table 4.3.

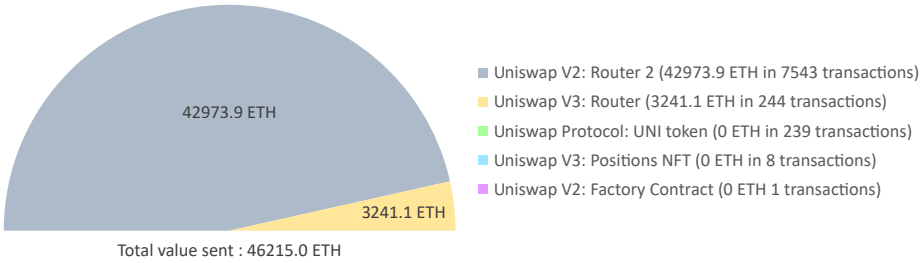


Figure 4.12: Receiving entities of initial traces and associated value

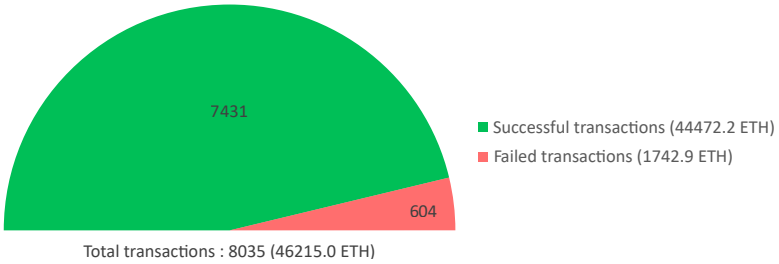


Figure 4.13: Amount of successful and failed transactions to the Uniswap protocol

Receiver	Success (value)	Fail (value)	Sum (value)
Uniswap V2: Router 2	6960 (41493.2)	583 (1480.7)	7543 (42973.9)
Uniswap V3: Router	229 (2979.0)	15 (262.2)	244 (3241.1)
Uniswap Protocol: UNI token	233	6 (0)	239 (0)
Uniswap V3: Positions NFT	8	0 (0)	8 (0)
Uniswap V2: Factory Contract	1 (0)	0 (0)	1 (0)
Sum	7431 (44472.2)	604 (1742.9)	8035 (46215.0)

Table 4.3: Receiving entities of initial traces and associated value

Now that we know where value is being transferred to, in Figure 4.14 we show the most common trace lengths. Inspecting the chart we can see that there are a lot of traces, that probably have the same structure, or at least very similar internal calls, regarding the length of the trace. A trace length of 8 is the most common occurring 2912 times. The next most common trace length is 10 occurring 1992 times. In 4.15 we get an even clearer picture of how often these trace lengths occur when different amounts of ETH are sent to the Uniswap contracts.

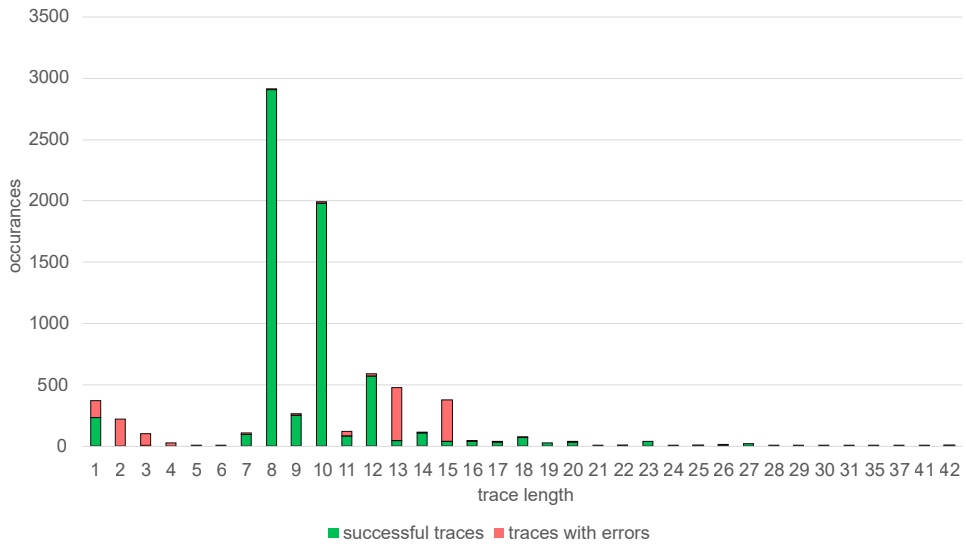


Figure 4.14: Most common trace lengths

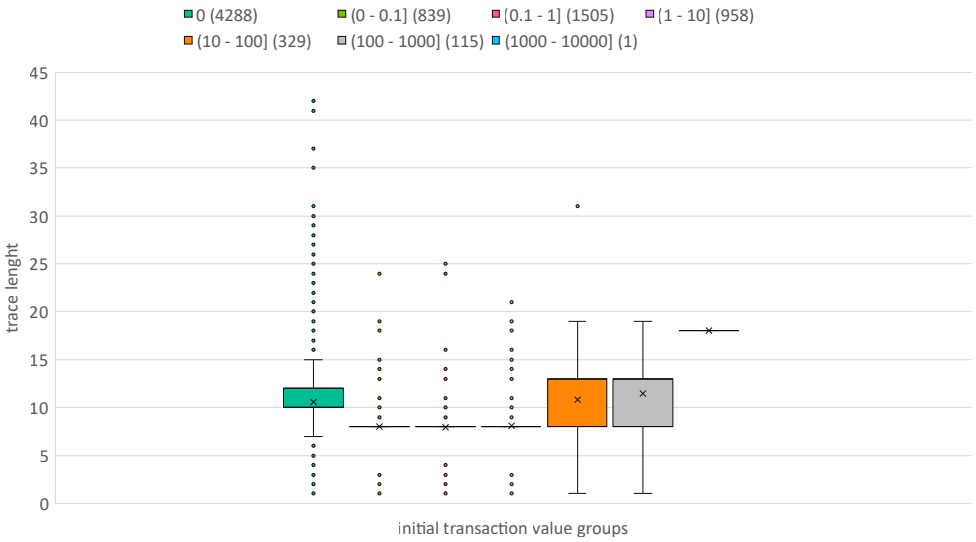


Figure 4.15: Lengths of traces grouped by initial transaction value

We can see that with a value greater than 0 and up to 10 ETH, almost all traces are comprised of 8 message calls, which result in a trace length of 8. Therefore, the initial transaction value groups from 0 to 0.1 ETH, 0.1 - 1 ETH and 1 - 10 ETH all have the most traces being of length 8. As the value increases to over 10 ETH we can clearly see that traces of length 8 do not occur as often, and traces have the tendency to become more complex, as the value increases.

ID	**pattern occurrence**	sum of initial transaction value	unique sender addresses	trace description
A	983	205.4	2	ETH → VSHIBA swap
B	237	166.1	1	ETH → STARK swap
C	233	0	145	UNI token approval
D	188	97.6	1	ETH → VBEE swap
E	146	849.2	1	ETH → MetaDAO swap
F	107	0	1	PAMP → ETH swap
G	66	55.7	1	ETH → STARK swap
H	64	1235.9	22	ETH → USDT swap
I	49	44.3	1	ETH → DTX swap
J	46	0	1	SNF → ETH swap

Table 4.4: Top 10 trace patterns by occurrence

ID	pattern occurrence	**sum of initial transaction value**	unique sender addresses	trace description
K	1	1700	1	ETH → USDC → DAI swap
L	27	1369.2	18	ETH → DAI
H	64	1235.9	22	ETH → USDT swap
M	4	1036	2	WBTC & ETH liquidity supply
E	146	849.2	1	ETH → MetaDAO swap
N	1	710	1	DMN & ETH liquidity supply
O	17	640	1	ETH → RAMP swap
P	19	503	6	ETH → renBTC swap
Q	2	500	2	ETH → USDC → DAI swap
R	1	500	1	DMN & ETH liquidity supply

Table 4.5: Top 10 trace patterns by sum of transferred value

ID	pattern occurrence	sum of initial transaction value	**unique sender addresses**	trace description
C	233	0	145	UNI token approval
H	64	1235.9	22	ETH → USDT swap
L	27	1369.2	18	ETH → DAI
S	13	373.6	10	ETH → USDC swap
T	21	0	10	UNI → ETH → USDT swap
U	17	410.4	9	ETH → WBTC swap
V	16	0	8	UNI → Ether → DAI swap
P	19	503	6	ETH → renBTC swap
W	12	0	8	SUSHI → ETH → USDT swap
X	6	0	5	YFI → ETH → USDT swap

Table 4.6: Top 10 trace patterns by unique sender addresses

For now, we see that most of the executed traces have a fairly simple structure based on their length of either 8 or 10 message calls, but by only looking at the length we cannot tell anything about the semantic meaning of the trace, and we cannot state how fraudulent entities use DeFi protocols to move funds. Tables 4.4, 4.5 and 4.6 therefore show the top 10 trace patterns by occurrence, transferred value and unique sender addresses, to make statements about typical behaviour of fraudulent entities.

By looking at the data, we come to the conclusion, that the trace in Figure 4.16 is the most common trace in our dataset. The trace describes a swap from ETH to the ERC-20 compatible Wrapped Ether (wETH) token and after that wETH is swapped for the Vitalik Shiba Buterin (VSHIBA) token. In the data, we see that only two different addresses were responsible for the 983 times this exact message call sequence was executed (only the value is differing). Although two different addresses are used, it can be assumed, that all of the swaps to this token are probably from the same entity, as it is unlikely that two unrelated scammers used the same token almost 1000 times. This would not be uncommon if we would inspect some well-known ERC-20 token like wETH or Tether Stablecoin (USDT), but as this token can clearly be identified as a scam token (one of the entities responsible for the 983 calls holds 22% of the whole token supply), it can be assumed that these two addresses are from the same entity. In total, the two addresses swapped about 205.4 ETH for this token. This could have been an effort to raise the price of the token in a pump and dump scheme, but we do not know for sure why these two entities choose to operate this way. As this swap seems like a special occurrence, we cannot say that this is typical illicit behaviour, and we therefore continue to investigate more of the found patterns.

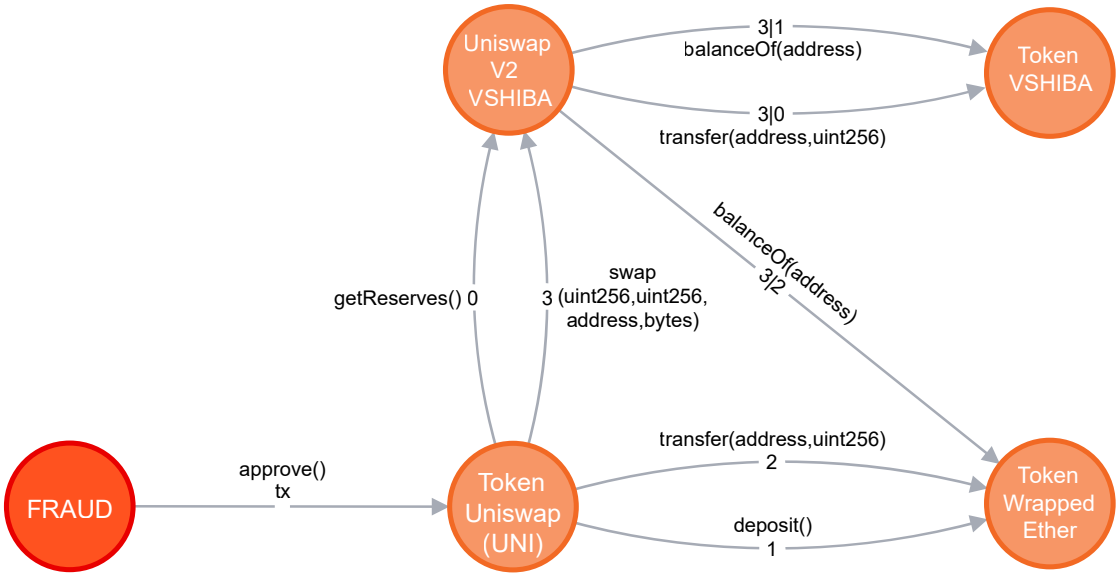


Figure 4.16: Trace A: ETH (Ethereum) → VSHIBA (Vitalik Shiba Buterin) swap

The second-most found trace pattern follows the exact same trace structure, except that the token involved in the swap is a different one. In this case, StarkWare (STARK) is the involved token, and again there are not a lot of addresses that behave this way, in fact, it is only one address, responsible for 237 transactions resulting in a total value of around 166.1 swapped ETH.

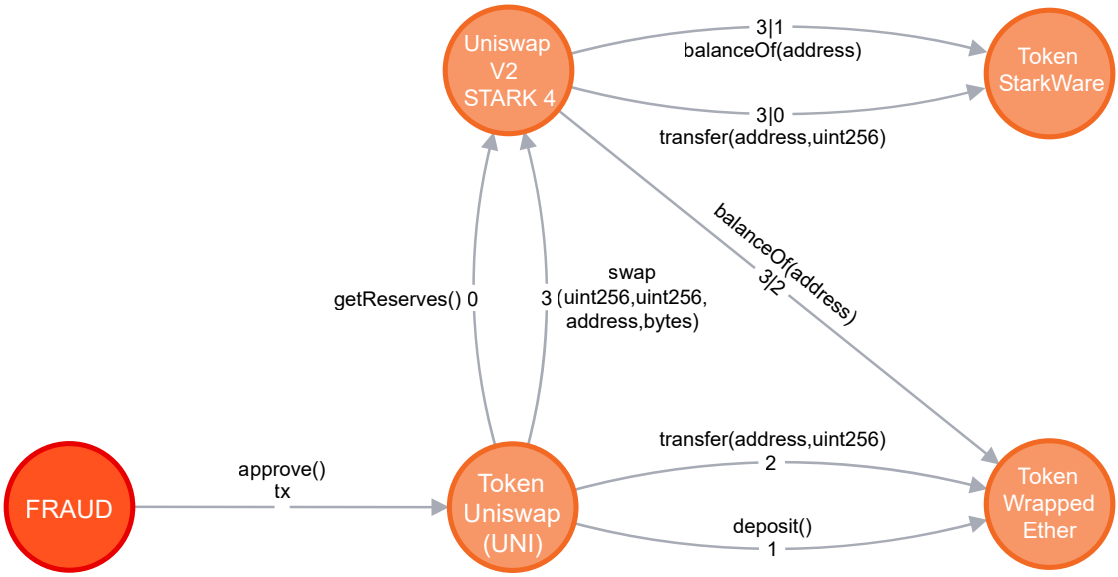


Figure 4.17: Trace B: ETH (Ethereum) → STARK (StarkWare) swap

The third-most executed trace pattern is more interesting to investigate as it was executed 233 times by 145 different addresses. As this trace is executed by many different entities, it shows a typical use case that fraudulent entities are aware of and use regularly. The trace itself describes an approval to spend Uniswap (UNI) tokens by some other address. Basically, all this trace pattern does, is call the standard ERC-20 *approve()* function on the UNI tokens contract, to allow another address to access and transfer a specified amount of tokens with the permission of the transaction initiator.

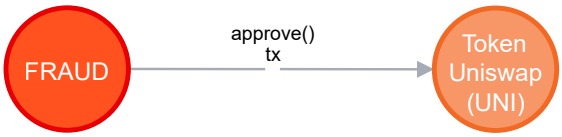


Figure 4.18: Trace C: UNI (Uniswap) token approval

As there is no associated value for this kind of message - the approved amount is specified inside the message's call data - this behaviour could be used to obfuscate token transfers in a not very sophisticated way. For explorers only looking at transaction-value-flows between different Ethereum accounts, the amount of approved tokens remains hidden, as

they are only investigating the value that was associated with the message call, which in this case is zero. As the amount of approved tokens can still be accessed by processing the message's input, this is not a very sophisticated form of obfuscation and it remains debatable if obfuscation is really the intention here.

As illicit entities often use multiple addresses to distribute their holdings and activities, this is another assumption on why this approval is used so often. In a research paper about email spams in the Bitcoin ecosystem [PCRHC19], Paquet-Clouston et al. hint that multiple addresses involved in spamming schemes are likely connected. Just like email spammers often use a range of different addresses, crypto scammers often distribute their accounts as well. Approving one or more addresses for token transfers could then be used to collect funds from a pool of other fraudulent addresses. Additionally, holding (and therefore being able to swap) UNI tokens has a range of benefits for fraudulent addresses. Probably the biggest advantage comes from the fact that holding UNI enables the holder to swap the tokens for basically any other available token at any point in time. An attacker can therefore quickly swap tokens and specify who can access them, making it a little bit harder for investigators to track down the illicit activity.

In Table 4.4 we have seen the patterns which occurred the most often, but these traces are only responsible for a small portion of funds which were sent to the Uniswap protocol. In Table 4.5 we not only look at the occurrence of traces but sort the traces by the sum of their initial transaction values. This gives us a better indication of which transaction schemes are used to swap big amounts of ETH. The first entry again is not very interesting to look at (besides its high value of 1700 ETH), as it is only a single transaction, and therefor again a special occurrence. In 4.19 we see the structure of the trace.

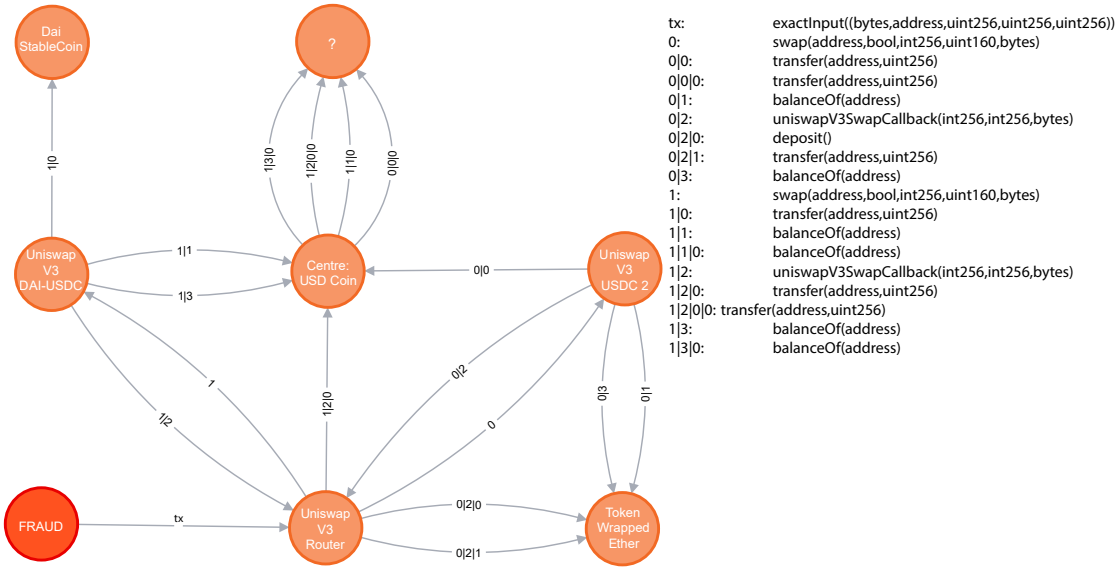


Figure 4.19: Trace L: ETH (Ethereum) → USDC(USD Coin) → DAI (Dai Stablecoin) swap

The second and third entries however are more interesting and show, that fraudulent entities particularly often swap their obtained ETH for Stablecoins of MakerDao (DAI) and USDT. In total 1369.2 ETH was swapped for DAI in 27 transactions coming from 18 different addresses, leading to an average transaction volume of about 50.7 ETH. The USDT Stablecoin occurred in 64 transactions coming from 22 unique senders summing up to a total value of 1235.9 ETH and an average transaction value of 19.3 ETH per transaction. The trace structure of both swaps can be seen in 4.20 and 4.21.

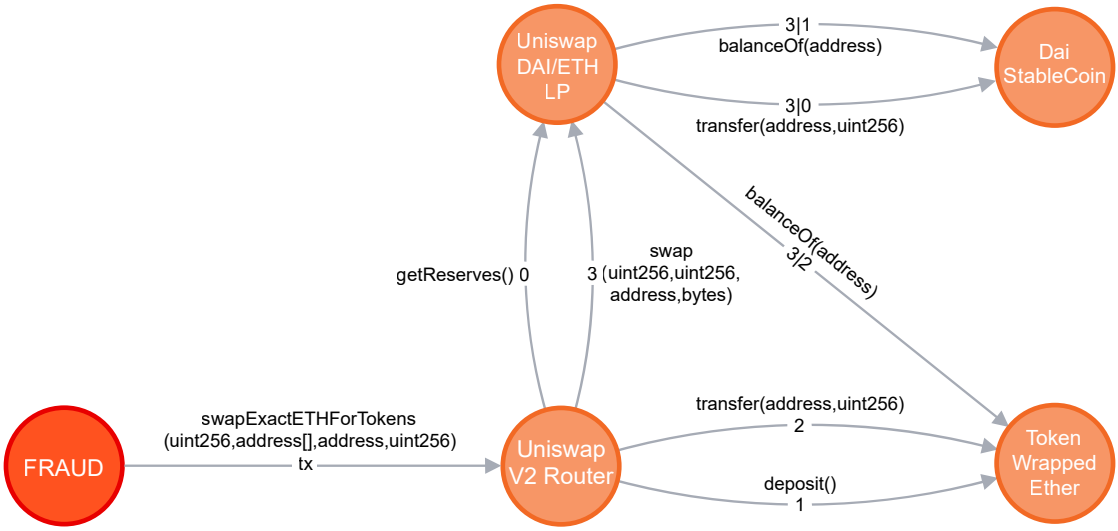


Figure 4.20: Trace L: ETH (Ethereum) → DAI (Dai Stablecoin) swap

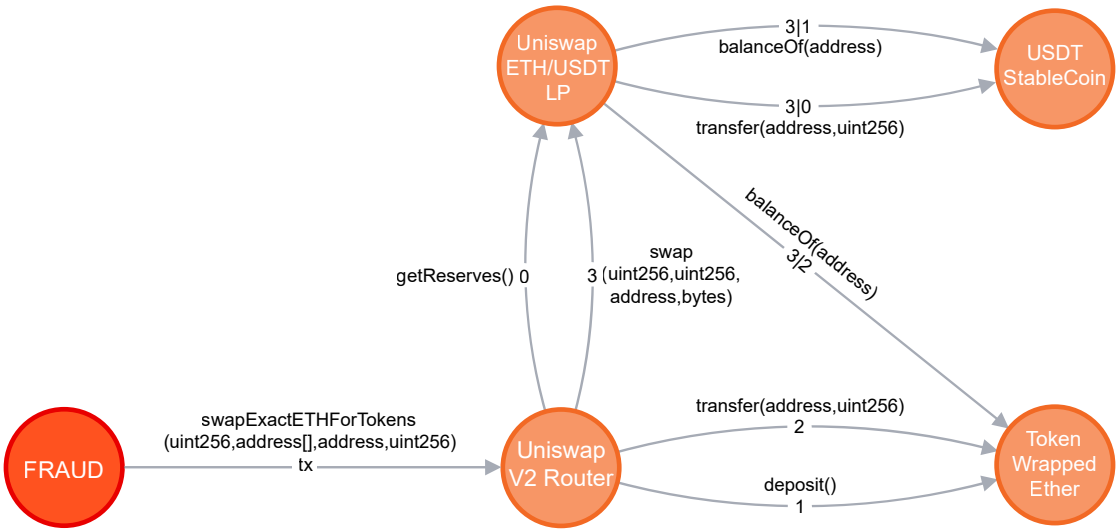


Figure 4.21: Trace H: ETH (Ethereum) → USDT (US Dollar Tether) swap

We can also see that these two traces, as well as the Uniswap token approval (shown in Figure 4.21) are also among the top traces when we order by the number of different sender addresses, meaning that these are common transactions that fraudulent addresses execute (see Table 4.6).

In Table 4.7 we show matching transaction hashes, by which anybody can identify the found pattern. The value associated with the transaction can vary from the value associated to the shown transaction hashes, but the trace structure for all the patterns mentioned in tables 4.4 and 4.5 are identical to the structure of the listed transaction hashes.

ID	transaction_hash
A	0xbb483ef1a994b22de626ddc50ee704930c610fcc4f8a9a6d157215c39507c51c
B	0x59d3dc1ec09b2057e29f24e90ce678b60d2aff2a8dd2b3f4bec54d5b0f15460d
C	0xf18e97814b2d6332dd221247265d432e3a6f7c09097bae36efd33998b2a16627
D	0xd032c3f7c54e9a7450990021708a063d3e18e19238c3c6811d9aedcb4c4310f0
E	0xd0270ed2a7f0349e748ae7c40c811974fc3e968392d4ccf4425757e45335e40c
F	0x6b34b92bf30ddb4d308cf45f09e64af5a6a753f51e1820170e064a0445b5e39a
G	0x6331309b1af2566e792a6146b0311937de65763a4c036a325bc68be57dd4e866
H	0x8dde2c222968afcb443dcdf30c6135eb7b89757f3d57d7409ef4ef5b067c8853
I	0x377873e8acbd254f62b250c89b8f9861347cf7bd24184f432e925f60a1d06bcd
J	0x3f5698e5eae12dcdadcce8bbbc727286ab2d1c7e70bc5459050b860e24084ffb
K	0x2207b5c4abd07f699ec5a5289f06d6f9b746d9d87a120fed2d8935d1eee23ce9
L	0xacc3f46aede9dbc71ad8f52ef1765a2d7d3e9364ff541c6c373cc28f6fb538e0
M	0xfc15a9152acc9ff9dc78bcb5b483c62971f7bfd988257578e753175a2ee6ffa5
N	0x6c9eccf0f3b4f3ec21337fab65e0da2c9a4d6aeab7fb4a3c3787069d90908f79
O	0xda2aa7f39cd7dc39922da38247b525556382bb3c6377a5abe8079aaeabf72af8
P	0xe0b326c67015902de03e35ab43e802393e4f3064e6dce831c510bfa8cb5edad0
Q	0x8c02254d5c27e9e07c58069bbc91ac8d7c1bbb10020ab806efb30de7f79deded
R	0x6c267fd75b2ad5a3fe59d001387b7b8d0ead3b880d853941414bfcc6bb6bc822
S	0xc33c718223fb0a89cc75aeb11d77ff3ac9e85a6facab65de0182eacb887603c5
T	0xf53f8b0844f4ad4f1c2053556e7468e4d417dc7279761a3bdb7f0ee195cad033
U	0xf3b1846f63cf7990bbd0183c85625e3b0e10c68f9f9800fdfea64ae569909ace
V	0xee469c25728e9bb240b4810014e6b1dd5348d73619c842059daf1b31404a9246
W	0x4e384d25a1e89e669654913557a2476ef9aefe0cdac696176411ee9ec43cf751
X	0xbba02ac329cd003b7183650e110776fa9fcc65a011c7efadeba8faa3047c2302

Table 4.7: List of transaction hashes to identify trace structures

As we find that most of the traces can be identified to be token swaps, we also wanted to find out which tokens are widely used by fraudulent entities. For this purpose, we check inside the trace data which token addresses appear most often. We then show how much value was sent directly to the token contracts. We also show the sum of initial values, in case a specific token was involved. In table 4.8 we show that ETH is always first swapped to wETH. This presumably is an internal Uniswap specification and the conversion happens internally without the transactor specifically stating that value should

first be swapped to wETH. Because of this behavior, we can see that besides wETH no other token in the top 25 occurring tokens directly receives any ETH in column v_d. Column v_s shows the sum of initial values of the traces if the token occurred inside of them. Column v_s therefor gives a better indication of how much ETH fraudulent entities actually use in swaps with other tokens. Besides the wETH token contract, the Uniswap V2 Factory Contract, DAI, a special Token with no Nametag by Centre[3] and USDT are in the top 5 of the most occurring tokens.

Concluding the trace analysis we can state, that fraudulent addresses make use of DeFi protocols in many different ways, although a lot of them also follow simple, more common, approaches. They use DEXs to store their funds and approve other addresses to move and manage them regularly. They also regularly swap ETH for a variety of different ERC-20 tokens. Stablecoins are among the top tokens fraudulent addresses swap, possibly showing that they regularly try to cash out their stolen funds. We also saw that likely single entities are responsible for lots of transactions, and that there are many different schemes and unique trace patterns. 2465 of the 3145 found traces are unique regarding their structure, showing that fraudulent entities use the Uniswap protocol in many different ways. One of the most used functionalities they use is Uniswaps token approval method, which allows for an easy "transfer" of funds between addresses. For these approvals, conducting further research could be interesting, to find out which addresses got approved by fraudulent entities.

[3] https://www.centre.io/

ERC-20 token address	token name	occurance	v_d	v_s	v_a
0xc02aaa39b223fe8d0a0e5c4f27ead9083c756cc2	Wrapped Ether (WETH)	21639	44420.7	44472.2	2.1
0x5c69bee701ef814a2b6a3edd4b1652cb9cc5aa6f	Uniswap V2: Factory Contract	1434	0	28768.3	20.1
0x6b175474e89094c44da98b954eedeac495271d0f	Dai Stablecoin	611	0	4121.3	6.7
0xa2327a938febf5fec13bacfb16ae10ecbc4cbdcf	FiatTokenV2_1*	294	0	2466.8	8.4
0xdac17f958d2ee523a2206206994597c13d831ec7	Thether: USDT Stablecoin	1314	0	1866.3	1.4
0x2260fac5e5542a773aa44fbcfedf7c193bc2c599	Wrapped BTC: WBTC Token	196	0	1682.1	8.6
0xbb2b8038a1640196fbe3e38816f3e67cba72d940	Uniswap V2: WBTC	141	0	1654.1	11.7
0xa478c2975ab1ea89e8196811f51a7b7ade33eb11	Uniswap V2: DAI	341	0	1411.4	4.1
0x0d4a11d5eeaac28ec3f61d100daf4d40471f1852	Uniswap V2: USDT	938	0	1372.4	1.5
0x68bcfbddccb1650cbe584eb80cdd0a6e46ad134c	Uniswap V2: META 57	327	0	942.3	2.9
0x6795a9603e9992417560479b07ba04c5f0dd79e4	Meta Dao (META)	328	0	942.3	2.9
0x1b7de8867c202a95fb192e3f969d7db8fb3a9882	RAMP DEFI (RAMP)	74	0	905	12.2
0x77e0e87b3f1a1336419b2c39a2ee11271083413b	Uniswap V2: RAMP 11	75	0	905	12.1
0x0374a144f799bc64ec00c03dbb9065ad19e84c4e	Dracula Network (DMN)	10	0	730	73
0x964ab334f74aed1604b806ed299b422e41e99d35	Uniswap V2: DMN 2	11	0	730	66.4
0x99224c18424b6c23a21de62ebbb226d6a0062c45	Uniswap V2: NIX	68	0	615	9.0
0xff7c3b7f4e4260097a33c9dcc291b9d1baf2edb5	Fake_Phishing4526	76	0	615	8.1
0xca5397a1611959e6103f19fe89195b2e94ab6231	Reef Finance (REEF)	12	0	540	45
0x2b465a15e67215fb167f497d324cdc79bab31e11	Uniswap V2: REEF 1	12	0	540	45
0x81fbef4704776cc5bba0a5df3a90056d2c6900b3	Uniswap V2: renBTC	48	0	503.0	10.5
0xe2d6ccac3ee3a21abf7bedbe2e107ffc0c037e80	RenERC20LogicV1*	56	0	503.0	9.0
0x1b3c27ef7248541e90b0cb49092571e56dee3e93	dracula.network (DMN)	6	0	500	83.3
0xc6bde036a9be176895510c65ebba8416c19f72e4	Uniswap V2: DMN 3	7	0	500	71.4
0x1ccc4638228a3f574a309197934f3e5f532df000	yearnX.finance (YFF)	8	0	500	62.5
0x47378fa8f5622875efc494b06f22c7d7a6560546	Uniswap V2: YFF 5	8	0	500	62.5

Table 4.8: List of token addresses, description
v_d: sum of value directly sent to the token address
v_s: sum of initial values if token occurs in the trace
v_a: average value per transaction (v_s/occurance)
*Tokens did not have a name tag on `https://etherscan.io/` therefor the contract name was used

CHAPTER 5

Discussion

We now summarize the conducted work in the thesis, show limitations to our approach and propose some fields of study that could be interesting for future research. First, we reiterate and summarize our work and findings. We then talk about limitations that occurred during our research. In the last part of this chapter, we further discuss potential refinements and future extensions for this research.

5.1 Summary of insights

DeFi protocols, which are non-custodial, permissionless financial services that run on top of the blockchain have gained significant amounts of interest in the past years. Assets, DEXs, lending and borrowing services as well as derivatives are among the top applications which the DeFi sector provides. With all these application fields, criminal activities related to this sector have also been on the rise. In 2020, illicit activity in the crypto space represented 0.34% of all transaction volume amounting to $10.0 billion. A near 20-fold increase of illicit funds flowing to DeFi during 2021 shows why this work is interesting.

In our web search, we found 20.181 fraudulent addresses, of which 6869 unique addresses remained, showing that a lot of providers have the same information accessible on the web. A variety of criminal activities like scams, exploits and hacks, Ponzi schemes and phishing attempts were identified.

In our topology analysis, we show that different fraud categories have different ingoing degrees indicating that certain categories attract more attention from presumably unknowing users. We also find that most of the fraudulent entities have only little fraudulent neighbours and low clustering coefficients. The biggest chunk of found addresses in our dataset (over 70% of them) are labelled as "Phish / Hack". Most of the addresses related to the "Phish / Hack" category have low ingoing degrees, and no or only little fraudulent

neighbours. Nevertheless, this category is in charge of the biggest sums flowing to DeFi. While this category was responsible for the biggest total sums flowing to DeFi, addresses with no categorization have the highest average transaction value flowing to DeFi.

In our analysis of transactions we find that, on the other end of fraudulent transactions, DEXs receive the biggest parts of funds. DEX Uniswap in particular is *the* main destination of illicit funds, although the cumulative sum of funds flowing to the Uniswap protocols in our dataset is not growing as fast as it did a few years ago.

To show fraudulent activities related to the most used protocol in more detail, we moved from the network perspective to a more detailed analysis of transaction schemes. By investigating trace data of an additional dataset of transactions which were sent to the Uniswap protocol we find that Uniswap token approvals are among the most common traces, probably used to manage and transfer funds without the need for an actual value transfer from one address to another. Swaps with high amounts of associated transaction values to USDT, DAI and other Stablecoins, are used by a number of unique illicit nodes, leading us to think that swaps to Stablecoins are quite popular among fraudulent entities. On the other hand, we also show that fraudulent addresses make use of DeFi protocols in many different ways and that we cannot speak of common fraudulent behavior when DeFi protocols like Uniswap are involved.

5.2 Limitations

During our work, we encountered some limitations and difficulties, which we want to mention in the below section, in hope that these remarks will be beneficial for future researchers investigating similar topics.

First, we want to mention that we cannot guarantee that all the addresses found by our web search were in fact connected to illicit behaviour, as errors such as wrong categorization or even completely wrong listings in the found datasets cannot be ruled out. We want to conclude the three main issues we encountered looking for data:

1. **Lack of reliable data sources**: Although public data sources provide APIs and full sets of addresses can be found, there are a lot of alleged providers, that have old data states, or have stopped collecting data altogether. We often encountered blank web pages or very small lists of addresses with no way to find out where the addresses came from. We can assume those addresses were tagged as scams or have some connection to illicit activities, but we will never know for sure.

2. **Wrong categorization**: Although some data providers like Etherscan[1] have a good reputation when it comes to collecting and tagging scam addresses, we don't really know where these addresses came from, and what the criteria was that got them listed or tagged as possible scam address. We therefore have no knowledge if

[1] https://etherscan.io/

the addresses are only included in the dataset because somebody at some point in time decided to include them, or if there is evidence-based data that lead to the entry. The same issue must be considered for the actual tagging of the data. We cannot be sure, that everything that is tagged as "scam" can also be viewed as a scam, or if different providers maybe have different views on what actually comprises a scam.

3. **User-generated lists**: In a user-generated list, users of crypto wallets or services have the possibility to report fraudulent activities and get addresses blacklisted. Of course, this also leaves the possibilities for wrong listings, as well as stale entries, that were listed once, and not removed in case of wrong accusation.

In conclusion, we cannot state with certainty, that every single entry in our dataset is actually a fraudulent entity, and investigating further in this matter would be suitable for future work. Nevertheless, we still think that most of the addresses listed are correct and well suited for our purpose, as we made sure to only include trustful sources, that have some reputation when it comes to providing data.

An intentional limitation of this thesis was our decision to only focus on DeFi protocols. As the DeFi sector is rapidly growing, we wanted to focus our efforts on the field of decentralized financial protocols. But as the effects of fraudulent activities are likely also affecting other areas, widening the scope of the investigation would pose an opportunity to further understand how fraudulent entities on the Ethereum blockchain operate.

In the course of writing this thesis, we also noticed that the data on we based our results is changing rapidly, and therefore reevaluations for this matter will always be a necessity to stay up to date with the latest data.

Last but not least, the manual investigation of traces we conducted in our thesis presented us with some difficulties. As there was no possibility to concisely check if a trace was a swap, an approval or anything else, we manually checked the traces' structure and applied semantic meaning to it. As there is no categorization for trace data, for now this was our only option but further efforts in this direction could be beneficial for future researchers.

5.3 Future work

In this thesis, we presented how illicit transaction behaviour is connected to DeFi protocols, and which protocols are destinations of illicit funds. Our transaction and trace analyses give a good insight into how funds move on the Ethereum blockchain, but there is still a lot of room for improvement that would be suitable for future work. The following ideas present some of the possibilities of how this work could be extended in the future.

Manual data classification As we show, the unreliable, user-generated data in our thesis is a limitation to how exact our analysis is. Doing a manual data classification on

potentially fraudulent addresses and dividing categories like "Phish / Hack" into more fine-grained sub-categories would be beneficial for further investigations.

Updating set of DeFi protocols As in this thesis we use the same set of DeFi protocols already used in the paper of Kitzler et al.[KVSH21] (Nov. 2021), we think that updating the set of DeFi protocols could also lead to new findings. We mainly think that more decentralized protocols in various spaces could have been introduced in the meantime, so updating the set and keeping it up to date could refine future results.

Expanding dataset of malicious addresses As Li et al. [LBB+21] have shown, the set of malicious addresses can be expanded by using different network clustering mechanisms. Their approach has shown to expand the set by a big amount from 3559 to 23,638 addresses, although the total amount of value sent in transactions was only marginally increased. Using different heuristics to expand the dataset, and comparing the deviation of the resulting addresses, could therefore hold great potential for future work.

Investigating token approvals In our thesis we show that many fraudulent entities use Uniswap's internal UNI token and its *approve()* function in order to allow other addresses to manage and transfer funds. Investigating these function calls to see for which address an approval was invoked would give an even better insight and maybe this approach could even be used to further expand the dataset of fraudulent addresses For this case, the messages call data would need more investigation in order to make clear statements about fraudulent addresses and their relation via approvals. This method is particularly interesting as any approach based on networks connected by simple Ethereum transactions can easily miss relationships via approval, as the nodes are seemingly separated by a reliable intermediary node, for example a DEX like Uniswap.

Expanding to other blockchains Expanding the focus of this work to other blockchains with native support for SCs, one could show how fraudulent activities are conducted in a different ecosystem. As Ethereum is not the only blockchain that uses SCs, this would be an interesting field to investigate. Making out differences or similarities between different blockchains would therefore definitely be another interesting field of research.

CHAPTER 6

Conclusion

In our work, we provided insight into how fraudulent transaction activities on the Ethereum blockchain are connected to the Decentralized Finance (DeFi) ecosystem. By web search, we were able to find 6869 unique Ethereum addresses which were labelled fraudulent. The found addresses as well as aggregated transaction data of the whole Ethereum blockchain were used to populate a graph database. The graph database was then used to conduct a topology analysis, in order to understand how fraudulent entities are connected to the DeFi sector. Based on our network data we showed that fraudulent entities heavily use DeFi applications to manage and transfer funds. We show that Decentralized Exchanges (DEXs) receive the biggest parts of illicit funds, as they can be used to swap the blockchains native currency to a variety of other ERC-20 tokens. DEX Uniswap in particular is the main destination for fraudulent transactions. To gain a better understanding of this matter, we used another dataset of message traces showing which transactions exactly were executed on the Uniswap protocol. By analysing the message traces sent to the DEX we found that UNI token approvals are among the most common transactions. Likely, these transactions are used by fraudulent entities to manage and transfer funds without the need for an actual value transfer from one entity to another. We also show that fraudulent entities make use of a variety of tokens, indicating that fraudulent actors use wildly different approaches when it comes to hiding, investing or managing their funds. In summary, our thesis provides a macro perspective into fraudulent activities on the network level as well as a more detailed perspective on a transaction-based level to show which DeFi protocols are used by fraudulent entities in which ways. With the DeFi sector becoming more and more popular, understanding how fraudulent entities make use of certain DeFi protocols is a first step towards a broader general analysis of illicit activities and their connection to decentralized protocols.

List of Listings

List of Figures

List of Tables

Acronyms

API Application Programming Interface. 16, 68

CA Code Account. 10, 12, 13

CQL Cypher Query Language. 42

DAI MakerDao. 63, 65, 68

DAO Decentralized Autonomous Organization. 22

DeFi Decentralized Finance. ix, xi, xiii, 1–5, 7, 13, 15–20, 22, 23, 31, 36, 37, 39–43, 45, 47–49, 51–55, 60, 65, 67–71, 75, 76

DEX Decentralized Exchange. ix, xi, 19, 42, 45, 47–49, 53, 54, 65, 67, 68, 70, 71

DLT Distributed Ledger Technology. 1, 2

EOA Externally Owned Account. 10, 13, 14

ERC-20 Ethereum Request for Comments. ix, xi, 16, 17, 45, 56, 60, 61, 65, 73

ETH Ethereum. 7, 9, 14, 23, 53, 56–58, 60–65

EVM Ethereum Virtual Machine. 10, 11, 13, 55

NFT Non-Fungible token. 15, 16, 23

P2P Peer-to-peer. 21

SC Smart Contract. ix, xi, 1, 4, 7, 8, 10, 12–19, 23–25, 70

STARK StarkWare. 61

UNI Uniswap. 61, 62, 70

USDT Tether Stablecoin. 60, 63, 65, 68

VSHIBA Vitalik Shiba Buterin. 60

wETH Wrapped Ether. 60, 64, 65

Bibliography

[B+18] Paul Barnes et al. Crypto currency and its susceptibility to speculative bubbles, manipulation, scams and fraud. *Journal of Advanced Studies in Finance (JASF)*, 9(2):16, 2018.

[Bar16] Albert-László Barabási. *Network Science*, volume 27, page 6. Cambridge University Press, 2016.

[BCAZ20] Pavlo Burda, Tzouliano Chotza, Luca Allodi, and Nicola Zannone. Testing the effectiveness of tailored phishing techniques in industry and academia: A field experiment. In *Proceedings of the 15th International Conference on Availability, Reliability and Security*, ARES '20, New York, NY, USA, 2020. Association for Computing Machinery.

[BCCS20] Massimo Bartoletti, Salvatore Carta, Tiziana Cimoli, and Roberto Saia. Dissecting ponzi schemes on ethereum: Identification, analysis, and impact. *Future Generation Computer Systems*, 102:271, 2020.

[BGL16] Zinaida Benenson, Freya Gassmann, and Robert Landwirth. Exploiting curiosity and context: How to make people click on a dangerous link despite their security awareness. 2016.

[BLL+21] Massimo Bartoletti, Stefano Lande, Andrea Loddo, Livio Pompianu, and Sergio Serusi. Cryptocurrency scams: Analysis and perspectives. *IEEE Access*, 9:Abstract, 2021.

[Blo20] Cylynx Blog. Tracing the trail of the upbit hack. `https://www.cylynx.io/blog/tracing-the-trail-of-the-upbit-hack/`, 2020. Accessed: 2022-05-11.

[Blo22] Coinbase Blog. What is a token? `https://www.coinbase.com/learn/crypto-basics/what-is-a-token`, 2022. Accessed: 2022-06-13.

[But14] Vitalik Buterin. A next generation smart contract & decentralized application platform - ethereum white paper. `https://ethereum.org/669c9e2e2027310b6b3cdce6e1c52962/Ethereum_Whitepaper_-_Buterin_2014.pdf`, 2014.

[ByB22] ByBit. How to avoid p2p crypto scams and fraud. https://learn.bybit.com/bybit-p2p-guide/how-to-avoid-p2p-crypto-scams-fraud/#2, 2022. Accessed: 2022-04-24.

[CB20] Yan Chen and Cristiano Bellavitis. Blockchain disruption and decentralized finance: The rise of decentralized business models. *Journal of Business Venturing Insights*, 13:1,3, 2020.

[CGC+20] Weili Chen, Xiongfeng Guo, Zhiguang Chen, Zibin Zheng, and Yutong Lu. Phishing scam detection on ethereum: Towards financial security for blockchain ecosystem. In *IJCAI*, pages 4506–4512, 2020.

[Cha21] Chanalysis. Chainalysis crypto crime report. https://go.chainalysis.com/rs/503-FAP-074/images/Chainalysis-Crypto-Crime-2021.pdf, 2021. Accessed: 2022-03-08.

[Cha22] Chanalysis. Defi takes on bigger role in money laundering but small group of centralized services still dominate. https://blog.chainalysis.com/reports/2022-crypto-crime-report-preview-cryptocurrency-money-laundering/, 2022. Accessed: 2022-03-16.

[Cip21] CipherTrace. Cryptocurrency crime and anti-money laundering report. https://ciphertrace.com/cryptocurrency-crime-and-anti-money-laundering-report-august-2021/, 2021. Accessed: 2022-03-28.

[Cor20] Terranova Worldwide Corporation. Gone phishing tournament global benchmark report 2020. page 6, 2020.

[CZC+18] Weili Chen, Zibin Zheng, Jiahui Cui, Edith Ngai, Peilin Zheng, and Yuren Zhou. Detecting ponzi schemes on ethereum: Towards healthier blockchain technology. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, page 1409–1418, Republic and Canton of Geneva, CHE, 2018. International World Wide Web Conferences Steering Committee.

[ESES18] William Entriken, Dieter Shirley, Jacob Evans, and Nastassia Sachs. Eip-721: Non-fungible token standard. In *Ethereum Improvement Proposals, no. 721, January 2018*. 2018.

[Eur21a] Europol. Cryptocurrencies - tracing the evolution of criminal finances. https://www.europol.europa.eu/cms/sites/default/files/documents/Europol%20Spotlight%20-%20Cryptocurrencies%20-%20Tracing%20the%20evolution%20of%20criminal%20finances.pdf, 2021. Accessed: 2022-03-26.

[Eur21b] Europol. Cryptocurrency related searches - europol. https://www.europol.europa.eu/search?q=cryptocurrency, 2021. Accessed: 2022-01-17.

[FP21] Allan Fowler and Johanna Pirker. Tokenfication - the potential of non-fungible tokens (nft) for game development. In *Extended Abstracts of the 2021 Annual Symposium on Computer-Human Interaction in Play*, CHI PLAY '21, page 152–157, New York, NY, USA, 2021. Association for Computing Machinery.

[Fra22] Jake Frankenfield. Crypto tokens. `https://www.investopedia.com/terms/c/crypto-token.asp`, 2022. Accessed: 2022-06-13.

[GSD04] P. Grabosky, R. Smith, and G. Dempsey. Electronic theft: Unlawful acquisition in cyberspace. *International Journal of Law and Information Technology*, 06 2004.

[Har19] Christopher G. Harris. The risks and challenges of implementing ethereum smart contracts. In *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 104–107, 2019.

[HRM+21] J.T. Hamrick, Farhang Rouhi, Arghya Mukherjee, Amir Feder, Neil Gandal, Tyler Moore, and Marie Vasek. An examination of the cryptocurrency pump-and-dump ecosystem. *Information Processing & Management*, 58(4):Abstract, 2021.

[JJJM07] Tom N. Jagatic, Nathaniel A. Johnson, Markus Jakobsson, and Filippo Menczer. Social phishing. *Commun. ACM*, 50(10):94–100, oct 2007.

[KC20] Sesha Kethineni and Ying Cao. The rise in popularity of cryptocurrency and associated criminal activity. *International Criminal Justice Review*, 30(3):325–344, Abstract, 2020.

[KVSH21] Stefan Kitzler, Friedhelm Victor, Pietro Saggese, and Bernhard Haslhofer. Disentangling decentralized finance (defi) compositions. *CoRR*, abs/2111.11933, 2021.

[LBB+21] Jiasun Li, Foteini Baldimtsi, Joao P. Brandao, Maurice Kugler, Rafeh Hulays, Eric Showers, Zain Ali, and Joseph Chang. Measuring illicit activity in defi: The case of ethereum. In Matthew Bernhard, Andrea Bracciali, Lewis Gudgeon, Thomas Haines, Ariah Klages-Mundt, Shin'ichiro Matsuo, Daniel Perez, Massimiliano Sala, and Sam Werner, editors, *Financial Cryptography and Data Security. FC 2021 International Workshops*, pages 197–203, Berlin, Heidelberg, 2021. Springer Berlin Heidelberg.

[Mac22] Simon Mackenzie. Criminology towards the metaverse: Cryptocurrency scams, grey economy and the technosocial. *The British Journal of Criminology*, 02 2022. azab118.

[MHC12] Tyler Moore, Jie Han, and Richard Clayton. The postmodern ponzi scheme: Empirical analysis of high-yield investment programs. In Angelos D.

Keromytis, editor, *Financial Cryptography and Data Security*, pages 41–56, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[Nak08] Satoshi Nakamoto. Bitcoin whitepaper. 2008.

[New18] Mark Newman. *Networks*, pages 89–92. Oxford University Press, 2018.

[OEC20] OECD. Digital disruption in banking and its impact on competition. `http://www.oecd.org/daf/competition/digital-disruption-in-financial-markets.htm`, 2020. Accessed: 2022-01-17.

[PCRHC19] Masarah Paquet-Clouston, Matteo Romiti, Bernhard Haslhofer, and Thomas Charvat. Spams meet cryptocurrencies: Sextortion in the bitcoin ecosystem, 2019.

[PKPD22] Andrew Park, Jan Kietzmann, Leyland Pitt, and Amir Dabirian. The evolution of nonfungible tokens: Complexity and novelty of nft use-cases. *IT Professional*, 24(1):9–14, 2022.

[RCC+18] Witek Radomski, Andrew Cooke, Philippe Castonguay, James Therien, Eric Binet, and Ronan Sandford. Eip-1155: Multi token standard. In *Ethereum Improvement Proposals, no. 1155, June 2018*. 2018.

[SS21] Sean Stein Smith. Decentralized finance & accounting – implications, considerations, and opportunities for development. *The International Journal of Digital Accounting Research*, pages 129–153, 07 2021.

[Sza96] Nick Szabo. Smart contracts: Building blocks for digital markets, 1996.

[TM20] David Towmey and Andrew Mann. Fraud and manipulation within cryptocurrency markets. In Carol Alexander and Douglas Cumming, editors, *Corruption and Fraud in financial markets: Malpractice, Misconduct and Manipulation*, chapter 8, pages 205–249. John Wiley & Sons, 2020.

[VCS03] Vivek Vishnumurthy, Sangeeth Ch, and Emin Sirer. Karma : A secure economic framework for peer-to-peer resource sharing, 06 2003.

[WAH12] Anne M Wilkins, William W Acuff, and Dana R Hermanson. Understanding a ponzi scheme: Victims' perspectives. *Journal of Forensic & Investigative Accounting*, 4(1):1–19, 2012.

[WLZZ21] Jiajing Wu, Jieli Liu, Yijing Zhao, and Zibin Zheng. Analysis of cryptocurrency transactions from a network perspective: An overview. *Journal of Network and Computer Applications*, 190:103139, 2021.

[Woo14] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger, berlin version b8ffc51 – 2022-02-21. pages 1–2, 2014.

[WPG+21] Sam M. Werner, Daniel Perez, Lewis Gudgeon, Ariah Klages-Mundt, Dominik Harz, and William J. Knottenbelt. Sok: Decentralized finance (defi). *CoRR*, abs/2101.08778:2, 2021.

[Wro21] Christoph Wronka. Financial crime in the decentralized finance ecosystem: new challenges for compliance. *Journal of Financial Crime*, ahead-of-print(ahead-of-print), Jan 2021.

[XWL+20] Pengcheng Xia, Haoyu Wang, Xiapu Luo, Lei Wu, Yajin Zhou, Guangdong Bai, Guoai Xu, Gang Huang, and Xuanzhe Liu. Don't fish in troubled waters! characterizing coronavirus-themed cryptocurrency scams. In *2020 APWG Symposium on Electronic Crime Research (eCrime)*, pages 1–14, 2020.

[YJX+21] Shanqing Yu, Jie Jin, Yunyi Xie, Jie Shen, and Qi Xuan. Ponzi scheme detection in ethereum transaction network. In Hong-Ning Dai, Xuanzhe Liu, Daniel Xiapu Luo, Jiang Xiao, and Xiangping Chen, editors, *Blockchain and Trustworthy Systems*, pages 175–186, Singapore, 2021. Springer Singapore.

[ZYL+22] Yanmei Zhang, Wenqiang Yu, Ziyu Li, Salman Raza, and Huaihu Cao. Detecting ethereum ponzi schemes based on improved lightgbm algorithm. *IEEE Transactions on Computational Social Systems*, 9(2):624–637, 2022.