CHAPTER 1

INTRODUCTION

Recent years have witnessed a rapid growth of decentralized finance application, or DeFi application, on the public blockchain ecosystem, e.g., Ethereum [1]. Unlike in traditional finance, DeFi applications leverage the transparency and openness nature of decentralized network (i.e., blockchain) to provide a diversity range of financial services, e.g., lending, borrowing, collateralizing, exchanging etc., all without trust or dependency on third-party intermediaries. While DeFi has been gaining an increasing level of market growth in terms of both popularity and liquidity, its openness nature also leaves a large space to external attacks, which may severely threaten the security of DeFi participants' funds. To elaborate on this point, consider a real-world attack (see Figure 1) on bZx project, which is a DeFi project for lending and margin trading on Ethereum.

In this case, the attacker leveraged an oracle dependency of bZx on other DeFi projects (i.e., Uniswap and Kyber) in manipulating crypto asset exchange rates, making net profit with a single atomic transaction. the attacker launched a sequence of six internal transactions, consisting of borrowing (e.g., transaction 1 and 5), exchanging (e.g., transaction 2, 3, and 4), and paying back (e.g., transaction 6) crypto assets (i.e., ETH and sUSD). Note, these transactions are packed into a single external transaction, which is then executed atomically by Ethereum. In its execution, the attacker first borrowed 7, 500 ETH from bZx (transaction 1), then used 4, 417.86 borrowed ETH in exchange of sUSD with other DeFi projects (i.e., Uniswap, Kyber, and Synthetics in transactions 2–4). Because bZx relies on Uniswap and Kyber for price feed oracles, which are instead susceptible to large amount transactions, the attacker can therefore largely skewed exchange rate of ETH/sUSD in bZx in favor of him or herself. After that, he or she triggered transaction 5, borrowing 6, 799.27 ETH with all holding sUSD (i.e., 1, 099, 841.39), followed by a last transaction 6 in paying back 7, 500 borrowed ETH at the very beginning. The outcome of transactions 1–6 is thus a net profit of 2, 381.41 ETH (minus a small amount of ETH for paying gas fee [1]), or $600K, for the attacker.

The goal of this report is to demystify DeFi. It describes the basic attributes of DeFi services, the structure of the DeFi ecosystem, and emerging developments. A forthcoming Decentralized Finance Policy-Maker Toolkit will offer guidance on risks and policy

approaches for governments navigating this new space. DeFi is a general term covering a variety of activities and business relationships. We identify six major DeFi service categories—stablecoins, exchanges, credit, derivatives, insurance, and asset management—as well as auxiliary services such as wallets and oracles. While traditional finance relies on intermediaries to manage and process financial services, DeFi operates in a decentralized environment—public, permissionless blockchains. Services are generally encoded in open-source software protocols and smart contracts. Like blockchain technology more generally, DeFi has an enthusiastic base of evangelists, who promote its potential for efficiency, transparency, innovation, and financial inclusion. It also has its critics, risks, and unknowns. There have already been significant examples of fraud, attacks, governance controversies, and other failures in the DeFi world. At this early stage, it is essential for industry and governments alike to develop a well-informed and nuanced understanding of the opportunities, risks, and challenges.

0

CHAPTER 2

WHAT IS DEFI?

DeFi is a general term for decentralized applications (Dapps) providing financial services on a blockchain settlement layer, including payments, lending, trading, investments, insurance, and asset management. DeFi services typically operate without centralized intermediaries or institutions, and use open protocols that allow services to be programmatically combined in flexible ways. Historically, intermediaries have played essential roles within financial markets, serving as agents and brokers of trust, liquidity, settlement, and security. The range and value of intermediaries has grown over time to meet the needs of an increasingly complex financial system.

Since the 2008 Global Financial Crisis, there has been increased attention on inefficiencies, structural inequalities, and hidden risks of the intermediated financial system.3 More recently, controversies such as the GameStop short squeeze, in which retail investors were blocked from trading during a period of volatility, cast a spotlight on other shortcomings of legacy financial infrastructure: slow settlement cycles, inefficient price discovery, liquidity challenges, and the lack of assurance around underlying assets.4 DeFi aims to address some of these challenges—though many still apply to the DeFi ecosystem in its current state. DeFi leverages blockchain technology to facilitate alternatives to traditional service providers and market structures. It offers the potential for innovation and creation of new services for improving efficiency of financial markets—building upon work being done in financial technology (fintech) and blockchain technology more broadly. Whether it achieves this promise remains to be seen.

2.1 DEFI Building Blocks

DeFi takes advantage of various technologies developed in the blockchain sphere. All have applications outside of DeFi, but play essential roles within the DeFi ecosystem.

- **Blockchains:** Distributed ledgers serving as the settlement layer for transactions. Currently, most DeFi services operate on the Ethereum network, due to its capabilities and developer adoption.5 DeFi activity is growing on and across other blockchains as well.

- **Digital Assets:** Tokens representing value that can be traded or transferred within a blockchain network. Bitcoin and other cryptocurrencies were the first blockchain-based digital assets. Others have a range of intended functions beyond payments.

- **Wallets:** Software interfaces for users to manage assets stored on a blockchain. With a non-custodial wallet, the user has exclusive control of funds through their private keys. With custodial wallets, private keys are managed by a service provider.

- **Smart Contracts:** Blockchain-based software code that carries out, controls, and documents relevant events and actions according to predefined terms and rules.

- **Decentralized Applications (Dapps):** Software applications built out of smart contracts, often integrated with user-facing interfaces using traditional web technology.

- **Governance Systems**: Software-based mechanisms that manage changes to smart contracts or other blockchain protocols, often based on tokens that allocate voting rights to stakeholders.

- **Decentralized Autonomous Organizations (DAOs):** Entities whose rules are defined and enforced in the form of smart contracts.

- **Stable coins:** Digital assets whose values are pegged to a fiat currency, a basket of fiat currencies or other stable-value assets.

- **Oracles:** Data feeds that allow information from sources off the blockchain, such as the current price of a stock or a fiat currency, to be integrated into DeFi services.

CHAPTER 3

PROPOSED SYSTEM

The BLOCKEYE is implemented as a web platform with front and back-end services, where the back-end architecture. There are five functional modules in this architecture. At the bottom, BLOCKEYE extends a smart contract analyzer to perform oracle analysis as introduced earlier. Z3 [7] is adopted as the SMT solver in this module. In the middle are Tx Monitor and Analysis Engine. Transactions are monitored and collected via the Eventum framework, which streams events from blockchain to BLOCKEYE. Moreover, we implemented the analysis engine to detect potential attacks based on collected transactions and events. At the top layer, BLOCKEYE provides a Configuration module to allow users to specify detection criteria, e.g., in physical time or block number. Furthermore, the Task Manager module is designed to schedule detection tasks submitted from front-end and send back notifications to users with the Twillo library.

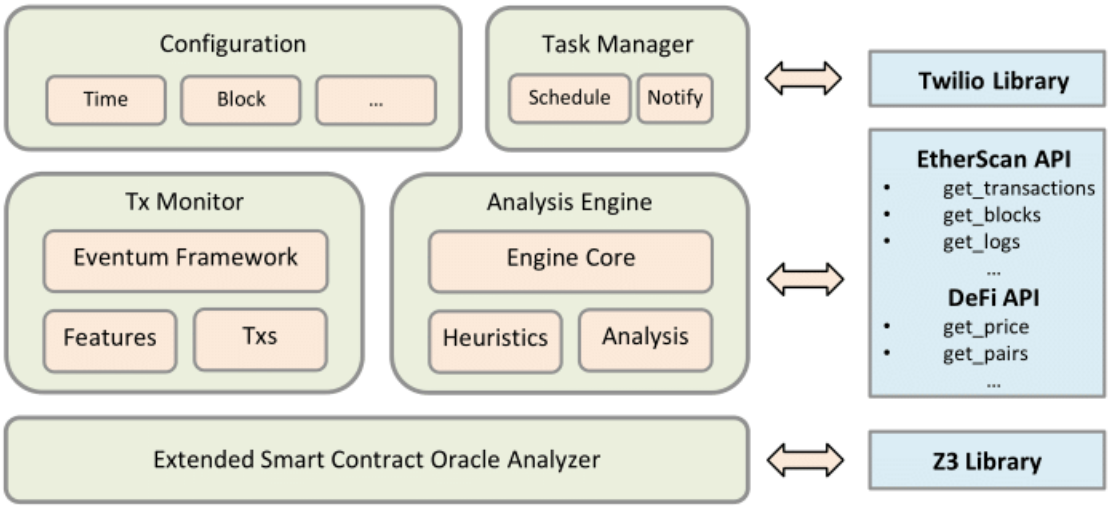

Fig. 3.1: The general architecture of BLOCKEYE.

3.1 Functionalities

BLOCKEYE expects DeFi smart contract source code as input. Users can either type in code in the code editor, or provide the address of a deployed DeFi project. BLOCKEYE then will try to load corresponding source code using Etherscan's source code retrieving API. Once smart contract code is available, users are free to click the START button to launch security analysis on the given DeFi project. An example output of BLOCKEYE is in Figure 3.2. Here,

results are divided into two parts: Oracle Analysis, which shows potential oracle dependencies found in DeFi source code, and Attack Monitoring, that provides information of real-world attack transactions which break heuristic invariants as described in Section II-C. For example, in Figure 3.2, BLOCKEYE has found an oracle dependency which spans across four smart contract functions, with oracle contract defined in code line 154 and fired by function calculate Continuous burn Return in line 168. Corresponding state access operation is in line 242 as a request to transfer DAI with dependent amount of value. Besides, in Figure 3.2, BLOCKEYE shows a list of five latest suspicious transactions, each with detailed information on its internal operation. At last, BLOCKEYE also presents a graph of top attackers along with their number of detected attack transactions, which may help users in further investigations.

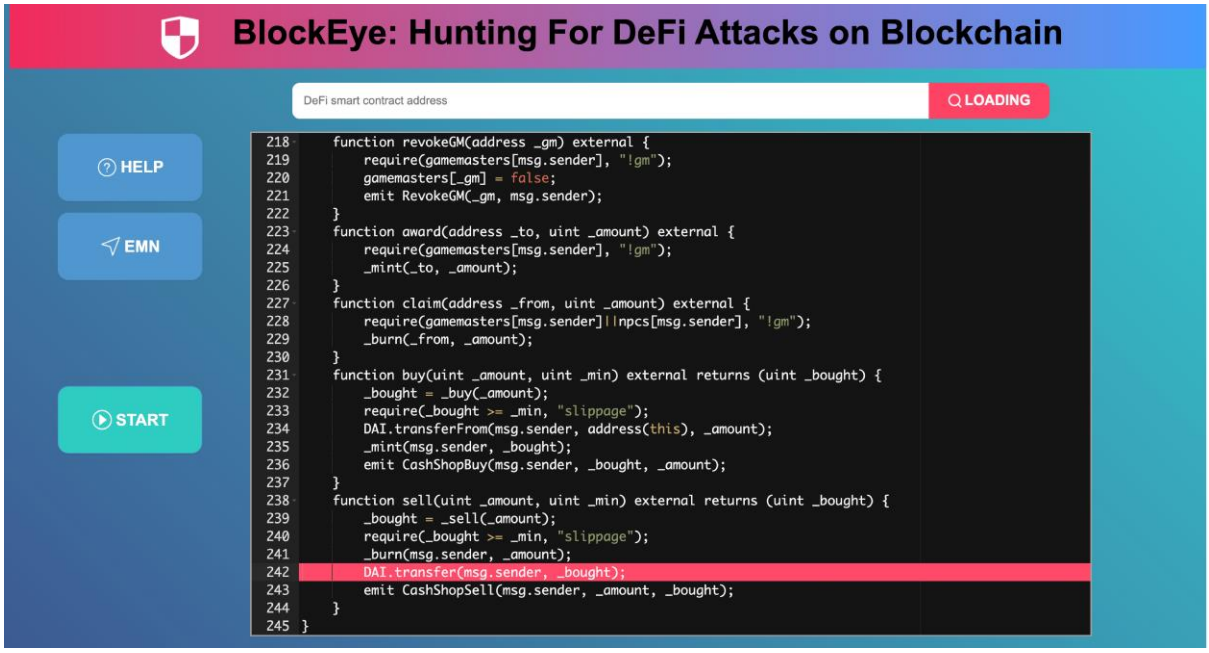

Fig. 3.2: The input interface for BLOCKEYE.

CHAPTER 4

LITERATURE SURVEY

[1]. G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," Ethereum Project Yellow Paper, vol. 151, 2014

With ubiquitous internet connections in most places of the world, global information transmission has become incredibly cheap. Technology-rooted movements like Bitcoin have demonstrated, through the power of the default, consensus mechanisms and voluntary respect of the social contract that it is possible to use the internet to make a decentralised value-transfer system, shared across the world and virtually free to use. This system can be said to be a very specialised version of a cryptographically secure, transaction-based state machine. Follow-up systems such as Namecoin adapted this original "currency application" of the technology into other applications albeit rather simplistic ones. Ethereum is a project which attempts to build the generalised technology; technology on which all transactionbased state machine concepts may be built. Moreover it aims to provide to the end-developer a tightly integrated end-to-end system for building software on a hitherto unexplored compute paradigm in the mainstream: a trustful object messaging compute framework

[2]. L. Luu, D.-H. Chu, H. Olickel, P. Saxena, and A. Hobor, "Making smart contracts smarter," in Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2016, pp. 254–269.

In this paper, we investigate the security of running smart contracts based on Ethereum in an open distributed network like those of cryptocurrencies. We introduce several new security problems in which an adversary can manipulate smart contract execution to gain profit. These bugs suggest subtle gaps in the understanding of the distributed semantics of the underlying platform. As a refinement, we propose ways to enhance the operational semantics of Ethereum to make contracts less vulnerableAmong 19, 366 existing Ethereum contracts, Oyente flags 8, 833 of them as vulnerable, including the TheDAO bug which led to a 60 million US dollar loss in June 2016. We also discuss the severity of other attacks for several case studies which have source code available and confirm the attacks (which target only our accounts) in the main Ethereum network.

[3]. C. Liu, H. Liu, Z. Cao, Z. Chen, B. Chen, and B. Roscoe, "Reguard: finding reentrancy bugs in smart contracts," in ICSE (Companion). ACM, 2018, pp. 65–68.

In this paper, we investigate the security of running smart contracts based on Ethereum in an open distributed network like those of cryptocurrencies. We introduce several new security problems in which an adversary can manipulate smart contract execution to gain profit. These bugs suggest subtle gaps in the understanding of the distributed semantics of the underlying platform. As a refinement, we propose ways to enhance the operational semantics of Ethereum to make contracts less vulnerable. For developers writing contracts for the existing Ethereum system, we build a symbolic execution tool called Oyente to find potential security bugs. Among 19, 366 existing Ethereum contracts, Oyente flags 8, 833 of them as vulnerable, including the TheDAO bug which led to a 60 million US dollar loss in June 2016. We also discuss the severity of other attacks for several case studies which have source code available and confirm the attacks (which target only our accounts) in the main Ethereum network.

[4] J. Kamps and B. Kleinberg, "To the moon: defining and detecting cryptocurrency pump-and-dumps," Crime Science, vol. 7, no. 1, p. 18, 2018.

Pump-and-dump schemes are fraudulent price manipulations through the spread of misinformation and have been around in economic settings since at least the 1700s. With new technologies around cryptocurrency trading, the problem has intensified to a shorter time scale and broader scope. The scientific literature on cryptocurrency pump-and-dump schemes is scarce, and government regulation has not yet caught up, leaving cryptocurrencies particularly vulnerable to this type of market manipulation. This paper examines existing information on pump-and-dump schemes from classical economic literature, synthesises this with cryptocurrencies, and proposes criteria that can be used to define a cryptocurrency pump-and-dump. These pump-and-dump patterns exhibit anomalous behaviour; thus, techniques from anomaly detection research are utilised to locate points of anomalous trading activity in order to flag potential pump-and-dump activity. The findings suggest that there are some signals in the trading data that might help detect pump-and-dump schemes, and we demonstrate these in our detection system by examining several real-world cases. Moreover, we found that fraudulent activity clusters on specific cryptocurrency exchanges and coins.

The approach, data, and findings of this paper might form a basis for further research into this emerging fraud problem and could ultimately inform crime prevention.

[5] L. Gudgeon, D. Perez, D. Harz, A. Gervais, and B. Livshits, "The decentralized financial crisis: Attacking defi," arXiv preprint arXiv:2002.08099, 2020.

This study aims to understand support vector machine and use it to predict lifestyle diseases that an individual might be susceptible to. In the existing system the info set is often little, for patients and diseases with specific conditions. These systems are principally designed for the additional prodigious diseases like cardiovascular disease, Cancer etc. The pre-selected characteristics could generally not satisfy the changes within the malady and its influencing factors that may lead to quality in results. The main feature will be the machine learning, in which algorithm use are such as Naïve Bayes Algorithm, K-Nearest Algorithm, Decision Tree Algorithm, Random Forest Algorithm and Support Vector Machine, which will help us in getting accurate predictions.

CHAPTER 5

ATTACK DETECTION FOR DEFI

5.1 Overview

The general workflow of BLOCKEYE works in a two-phase manner. In the first phase, BLOCKEYE performs symbolic analysis on smart contracts of a given DeFi project. This is realized by extending an underlying smart contract analyzer SERAPH [6], which is also developed by our team. Specifically, the goal of this phase is to model the inter-DeFi oracle dependency, i.e., how does the oracle data provided by one DeFi affect services of another. In cases where oracle-dependent state updates are found, we identify the DeFi as potentially vulnerable. Next, BLOCKEYE installs a runtime monitor in the second phase for vulnerable DeFi projects to detect external attacks.

Specifically, BLOCKEYE uses a transaction monitor to collect related transactions based on extracted features, e.g., address. Then, end-to-end transactions are analyzed according to predefined heuristics, e.g., a large profit is made in a short period. Potential attacks are flagged by BLOCKEYE when an abnormal sequence of transactions is detected. Moreover, BLOCKEYE generates analysis report to help blockchain service providers diagnose the found problems.

5.2 Oracle Analysis

As a forementioned, BLOCKEYE performs oracle analysis to check whether a DeFi is dependent on the oracle provided by another DeFi. Particularly, we focused on the price feed of assets shared through oracles.

```
function calculateContinuousMintReturn(uint _amount)
public view returns (uint mintAmount) {
return CURVE.calculatePurchaseReturn(totalSupply(),
reserveBalance, uint32(reserveRatio), _amount);}
function sell (uint _amount, uint _min) external
returns (uint _bought) {
_bought = _sell(_amount);
require (_bought >= _min, "slippage");
_burn (msg.sender, _amount);
DAI.transfer(msg.sender, _bought); 13 ... 14 }
```

Specifically, the function call at line 9 implicitly invokes the function from line 1–5, which receives an oracle at line 3–4. Moreover, the payment at line 12 is dependent on the oracle due to a data flow from line 9 to 12. That said, EMN has an oracle-dependent state update in its smart contracts. To enable such oracle analysis, BLOCKEYE extended the SERAPH smart contract analyser to perform symbolic reasoning on oracles. Specifically, when processing a CALL instruction to a specified oracle, BLOCKEYE starts a data flow analysis to track whether the value retrieved from the oracle is linked to a further state operation, e.g., payment, storage update etc. In cases where a feasible link is detected as in Figure 3, BLOCKEYE dumps the data flow and identifies given DeFi as potentially vulnerable.

5.3 Automatic Attack Monitoring

For vulnerable DeFi projects, BLOCKEYE launches a runtime transaction monitoring to detect external attacks. To this end, BLOCKEYE allows users to specify targeted projects and further performs end-to-end analysis on relevant transactions. In general, the analysis aims at finding violations on invariant as predefined heuristic rules. Specifically, as in Figure 1 with associated DeFi projects Uniswap, Synthetic and Kyber, BLOCKEYE first marks a random transaction t0 in these platforms as a target. Moreover, we search for other related transactions t1 · · ·tk based on the sender address x in t0. Additionally, we filter transactions which are not in the same block as t0 in order to find frequent transactions, which are more likely to be involved in attacks. With the collection of t0 · · ·tk transactions, BLOCKEYE runs a process to calculate the benefits received by x and its cost as well. With both numbers, BLOCKEYE is then able to determine whether x is attacking the target DeFi by comparing the profit made by x and a threshold value as configured. BLOCKEYE will also dumps the malicious sequence of transactions to facilitate an in-depth analysis of the potential attack.

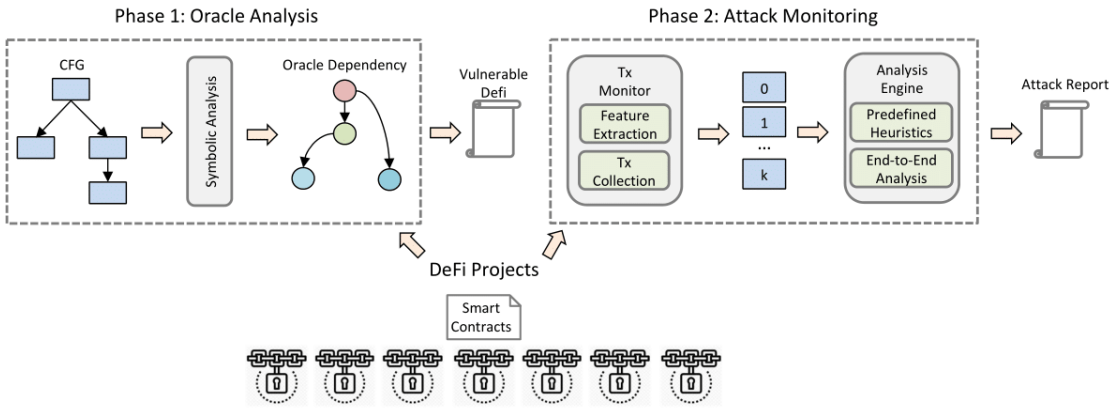

Fig. 5.1: The general workflow of BLOCKEYE.

CHAPTER 6

INCENTIVE STRUCTURES AND GOVERNANCE

While strictly speaking not required, most DeFi services incorporate token-based incentive structures for important objectives such as liquidity and governance. As shown in Figure 2, these typically involve digital asset holders locking up assets in order to receive payments (similar to earning interest on a certificate of deposit) and DeFi users paying in fees (analogous to interest rates) to access assets from the resulting pool. Unlike traditional finance, this arrangement applies to virtually every category of DeFi. The assets obtained might be stablecoins, a different token than the one the user provides (in the case of an exchange), loans, or insurance contracts, for example.

Common mechanisms include lock-up yields that pay interest for immobilizing digital assets in pools, where they serve as liquidity or collateral for a DeFi service; liquidation fees that pay market-makers a percentage of the value of under-collateralized, liquidated loans; and liquidity mining that pays the interest in the form of tokens issued by the DeFi service itself. Because of DeFi's composable, programmatic architecture, these mechanisms can be further integrated into structures such as yield farming, which optimizes returns from liquidity mining and lock-up yields by automatically moving funds across DeFi services. The earnings rate may be determined in several ways, including a pro-rata share of transaction fees, parameters set through the protocol's governance process, or a bonding curve that rewards earlier participation.

The idea of using tokens to incentivize decentralized network growth is not new. It is the essential to the consensus system of Bitcoin and other cryptocurrencies. Coinbase co-founder Fred Ehrsam outlined in 2016 how tokens could be used to help solve the proverbial chicken and egg problem of bootstrapping new networks and marketplaces.7 In the 2017 initial coin offering (ICO) bubble, however, tokens were often sold to speculators who flipped them for a quick profit, providing little functional benefit to the networks.8 DeFi provides a new opportunity for token models that reward long-term focused participants. The provision of capital is not just a payment to fund future protocol development and reward insiders; it is a direct contribution to DeFi activities such as trading, lending, stablecoin collateralization, and insurance. More liquidity increases the value of the network, and some of that value flows

back to the liquidity providers. However, well-designed incentive structures and careful attention to leverage and volatility are needed to address risks.

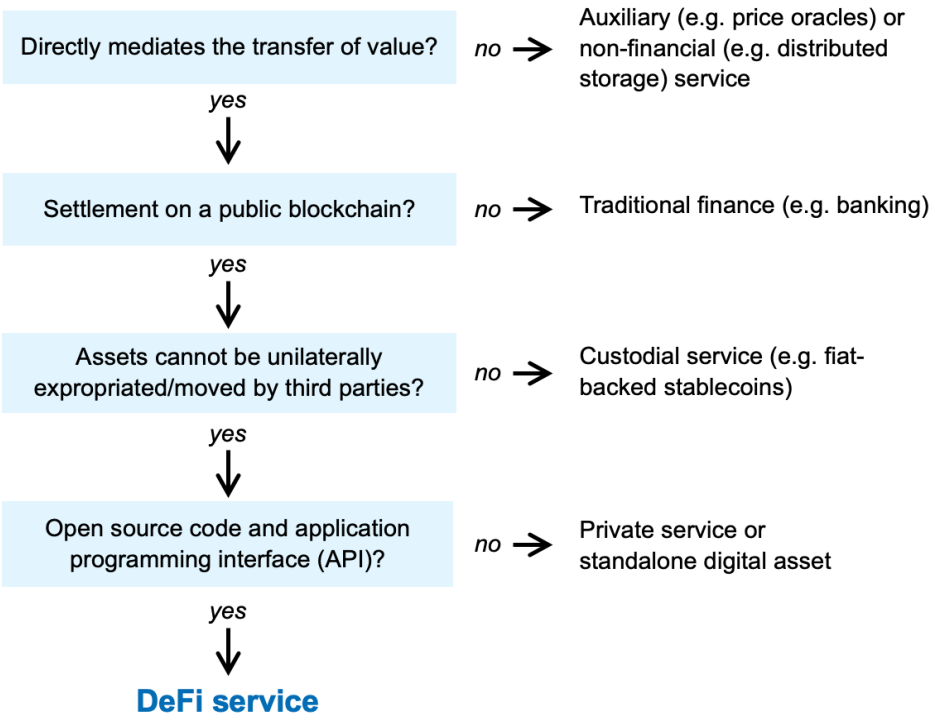

Figure 6.1 – DeFi classification flowchart.

One of the major applications of DeFi incentive structures is governance. Tokens issued in connection with liquidity mining or related mechanisms often provide governance rights for the DeFi service. Token-holders can vote on proposed changes to protocols, or on defined parameters such as interest rates or collateralization ratios. The scope of decisions that can be made by token holders other than the developers varies. These tokens are tradeable on certain exchanges, with their value in theory tied to the activity level of the issuing DeFi service.

CHAPTER 7

Security Problems in DEFI Projects

Detection of these attacks requires a deep understanding on both the business nature of a DeFi project as well as the market it is involved in, which are missing in existing solutions to finding low-level bugs of smart contracts. We summarized challenges in terms of addressing security problems in DeFi projects as below.

- **Challenge 1: Model DeFi Dependency:** Attacks on DeFi often involve multiple projects rather than a single one. Therefore, detection of such attacks requires an effective modelling of critical dependencies among DeFi projects, e.g., information flow between two DeFi. While a full analysis would introduce too much complexity, an abstract analysis might not be able to capture important high-level business semantics.

- **Challenge 2: Understand End-To-End Transactions**: Furthermore, whether a sequence of transactions is considered as malicious is largely determined by end-to-end analysis, i.e., comparing benefits and costs in the transaction sequence. However, such insights are hard to configure and generate based on existing analysis infrastructures for blockchains.

7.1 The BLOCKEYE Solution

To overcome above challenges, we have designed and developed BLOCKEYE, the very first automatic attack detection platform for blockchain DeFi projects. The key insights behind BLOCKEYE are twofold. First, a symbolic analysis is performed in BLOCKEYE to reason on important data flow (e.g., asset price) among associated DeFi projects. Potentially vulnerable projects are identified in this process. Then, BLOCKEYE installs a runtime monitor on vulnerable DeFi projects to detect potential attacks on the fly. Specifically, an end-to-end economic analysis is executed to report malicious transactions based on given heuristics, e.g., a large number of profits are made in a very short time. We further applied BLOCKEYE in several popular DeFi projects on Ethereum and managed to uncover potential attacks which were previously unreported.

CHAPTER 8

DEFI SERVICE CATEGORIES

DeFi embodies a variety of activities meeting the criteria of trust-minimized, non-custodial, open, composable, and programmable financial services. We identify six major DeFi categories, in addition to auxiliary services such as oracles and wallets. The lines between them are not always clear. However, this typology generally reflects participant perceptions of the DeFi market.

1. **Stablecoins:** seek to maintain a constant value of a token relative to some asset, most commonly the U.S. dollar or other major fiat currency. Non-custodial stablecoins function as DeFi services themselves. Custodial stablecoins are centralized but may be incorporated into DeFi services.

2. Exchanges: allow users to trade one digital asset for another. DeFi exchanges avoid taking custody of user assets, either through a decentralized order book or by matching orders and setting prices algorithmically.

3. Credit: involves the creation of time-limited interest-bearing instruments, which must be repaid at maturity, and the matching of lenders and borrowers to issue those instruments.

4. Derivatives: are synthetic financial instruments whose value is based on a function of an underlying asset or group of assets. Common examples are futures and options, which reference the value of an asset at some time in the future.

5. Insurance: provides protection against risks by trading the payment of a guaranteed small premium for the possibility of collecting a large payout in the event of a covered scenario.

6. Asset: management seeks to maximize the value of an asset portfolio based on risk preferences, time horizons, diversification, or other conditions.

While this report includes examples throughout, it is important to keep in mind that DeFi is developing quickly. These categorizations—and the projects that fit within them—may change over time. Because DeFi services are programmable and composable, aggregators have emerged that mediate activity across services in these base categories. Yield farming services such as Yearn Finance, which optimize returns from liquidity and collateral provision, are one example. Exchange aggregators such as 1inch and Matcha send trade orders to the exchange offering the best price. Zapper and Zerion provide integrated

interfaces for order routing, yield optimization, and other DeFi activities across different protocols, reducing complexity for users. Tally aggregates DeFi governance token activity to facilitate participation in governance decisions. Because of their potential for better usability, aggregators may originate a significant share of DeFi activity in the future.

8.1 Uniswap and Sushiswap:

Uniswap is a decentralized AMM protocol built on Ethereum. It uses an algorithm in which the product of the liquidity pool token count in the two digital assets being traded always equals a constant (x*y=k). Buying tokens removes them from the liquidity pool, causing their price to increase to maintain the constant. Uniswap also charges a trading fee, part of which is distributed to liquidity providers. Among other things, this model provides liquidity on very thinly traded assets and reduces high spreads for illiquid assets. From launch in November 2018 to the end of 2020, Uniswap facilitated over $100 billion in trading volume. As with other DeFi platforms, Uniswap is rapidly developing and adding new features.

In mid-2020, an anonymous developer forked the Uniswap software to create SushiSwap. It added a governance token, SUSHI, whereby the community votes on major changes to the protocol. A portion of trading fees across the platform are paid out to SUSHI token holders. The potential for value accrual in SUSHI tokens, in addition to liquidity provision, attracted substantial interest in SushiSwap. Uniswap subsequently created a UNI governance token. Uniswap’s cash flows do not yet accrue to the UNI token holders, but Uniswap has announced plans to do so.

CHAPTER 9

CHARACTERISTICS OF DEFI

Not every application of blockchain technology—even those involving financial transactions—is a form of DeFi. Nor is every element contributing to the DeFi ecosystem appropriately considered a DeFi service, business or software protocol. While the space is evolving, there are certain distinguishing characteristics:

1. Financial services: DeFi directly mediates the transfer and exchange of value. Auxiliary services such as oracles, query systems, and decentralized storage may be important enablers of DeFi activity, but they should be distinguished from DeFi services themselves.

2. Trust-minimized operation and settlement: DeFi projects generally build on public, permissionless blockchains offering smart contract functionality, such as Ethereum. Transactions are executed and recorded according to the rules of the DeFi protocols. Trust minimization is often extended to the governance structures that establish the conditions for protocol changes.

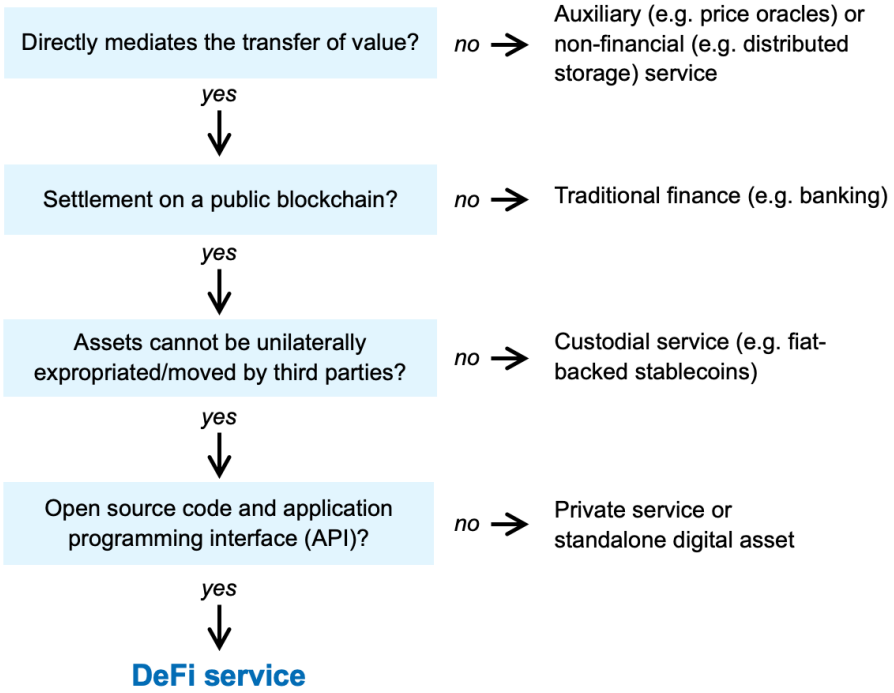

Fig 9.1 – DeFi classification flowchart.

3. Non-custodial design: The assets issued or managed by DeFi services cannot in theory be unilaterally expropriated or modified by third parties, even by those providing intermediation and other services. Users retain full control. Thus, centralized cryptocurrency exchanges that have custody over digital assets are not DeFi businesses, though many are developing DeFi offerings.

4. Open, programmable, and composable architecture: There is broad availability of the underlying source code and a public application programming interface (API). Components can be composed together and programmed to create new financial instruments and services dynamically. For example, a stablecoin may be used as the foundation for a derivative which is used as collateral on a loan and subject to an insurance contract.

5. Fixed Rate Products: offer a stable interest rate despite fluctuations in the value of underlying assets. Yield Protocol creates tokens that may be redeemed one-for-one for a target asset after a predetermined maturity date, similar to zero-coupon bonds. The implicit yield curve is beginning to serve as the foundation for additional products.

6. Credit Delegation: allows users to deposit collateral assets into a DeFi lending service such as Aave, and then authorize trusted users to draw loans against that collateral. The two parties specify details of the loan through an arrangement called a Ricardian contract, in which a legal agreement is cryptographically linked to an associated smart contract on the blockchain. This system allows a depositor with unused borrowing power to delegate a credit line to someone whom they trust to earn additional interest. Borrowers can also refinance existing loans at much more favorable terms.

7. Institutional Corporate Credit: services such as Maple Finance allow institutions to borrow from liquidity pools managed by experienced investors. These Pool Delegates are responsible for assessing borrowers and negotiating loan terms before lending from their managed pool.

CHAPTER 10

DEFI PROTOCOLS

We now present DeFi protocols categorized by the type of operation they provide. An overview is shown in Figure 1, while a classification of a selection of existing DeFi protocols.

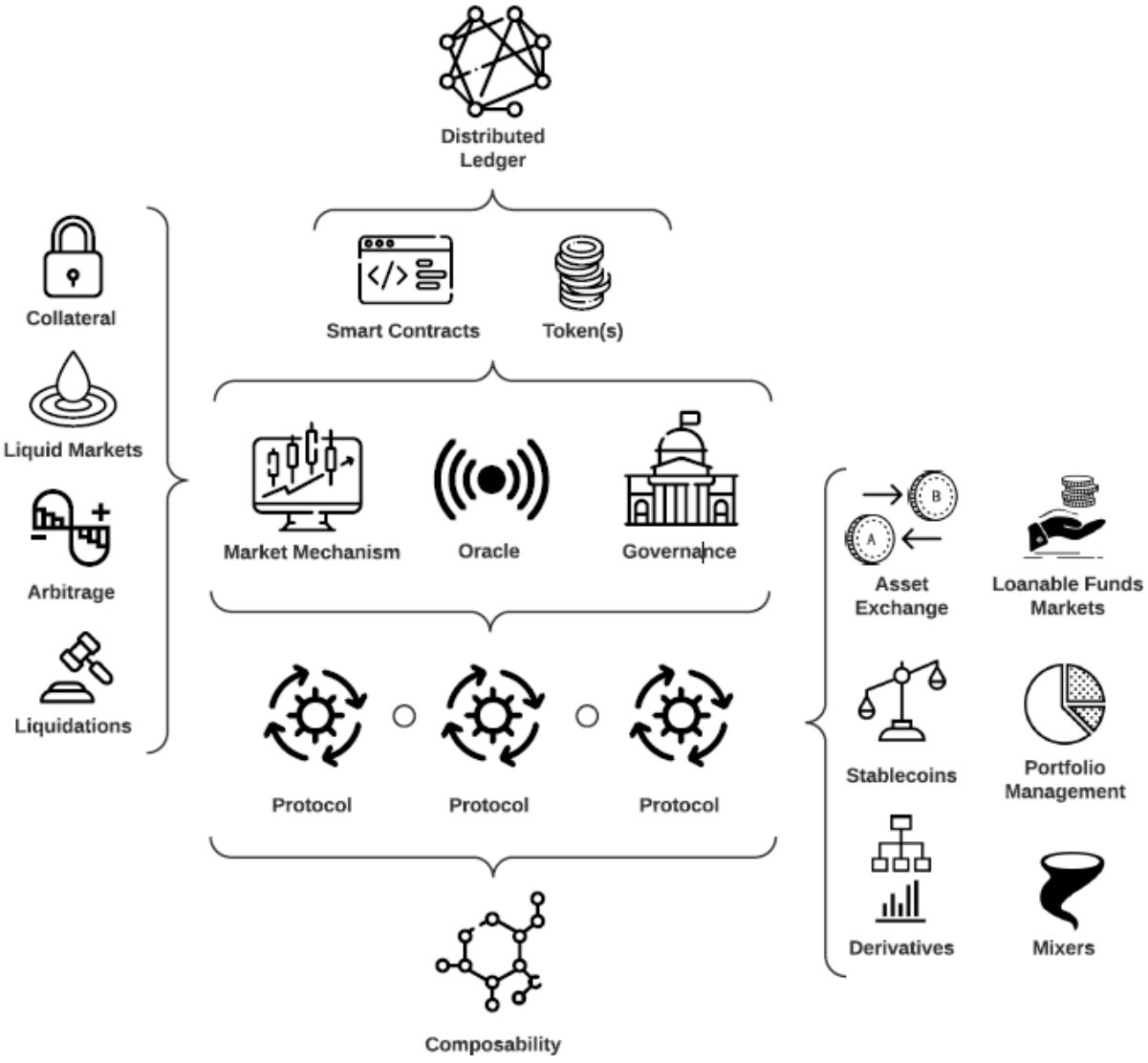

Fig. 10.1: A conceptual overview of the different constructs within the DeFi ecosystem.

- **On-chain Asset Exchange:** Decentralized exchanges (DEXs) [51], [52] are a class of DeFi protocol that facilitate the non-custodial exchange of on-chain digital assets. Apart from being non-custodial, i.e., the exchange does not have ownership over a user's funds at any point in time, a DEX settles all trades on-chain, thereby ensuring public verifiability for all transactions to network participants. While DEXs initially only supported assets native to the chain on which they operate, wrapped tokens, such as wBTC [24] (wrapped Bitcoin), and novel cross chain solutions [53], [15], [54], [55] have enabled DEXs to overcome this limitation. Today, based on the mechanism for price discovery, DEXs come in different variants, such as order book DEXs (including individual [56], [57] and batch settlement [58], [59]) and automated market makers (AMMs) (e.g., [60], [61], [62]).

  Due to their widespread adoption and novelty in DeFi, we specifically focus on AMMs. In traditional finance, market makers are liquidity providers that both quote a bid and ask price, selling from their own book, while making a profit from the bid-ask spread. Optimal market making strategies quickly become sophisticated optimization problems. In contrast, AMMs provide liquidity algorithmically through simple pricing rules with on-chain liquidity pools in place of order books. AMMs have been studied in algorithmic game theory, e.g., logarithmic market scoring rule (LMSR) [63] in prediction markets. While they have largely remained unimplemented in traditional finance, they have become popular in DeFi for a several reasons: (1) they allow easy provision of liquidity on minor assets, (2) they allow anyone to become a market maker, even if the market making returns are suboptimal, (3) AMM pools can be separately useful as automatically rebalancing portfolios
- **Loanable Funds Markets for On-chain Assets:** Lending and borrowing of on-chain assets is facilitated through protocols for loanable funds (PLFs) [71], which refer to DeFi lending protocols that establish distributed ledgerbased markets for loanable funds of cryptoassets by pooling deposited funds in a smart contract. In the context of a PLF, a market refers to the total supplied and total borrowed amounts of a token, where the available (i.e., non-borrowed) deposits make up a market's liquidity. An agent may directly borrow against the smart contract reserves, assuming the market for the token is sufficiently liquid. The cost of borrowing is given by an

interest rate charged to the borrower, which is determined by a market's underlying interest rate model [72].

An alternative to over-collateralized loans are flash loans. These are uncollateralized loans for the duration of a single transaction, requiring the borrower to repay the full borrowed amount plus interest by the end of the transaction. Flash loans leverage a blockchain's atomicity (i.e., the transaction fails if the loan is not repaid in the same transaction) and offer several use cases, such as decentralized exchange arbitrage and collateral swaps. However, they can also be used in attacks [74]. For a more detailed discussion and formal analysis of PLFs, we direct the reader to [71], [75].

- **Portfolio Management:** For liquidity providers seeking to maximize their returns, liquidity allocation can be an onerous task given the complex and expansive space of yield-generating options. The management of on-chain assets can thus be automated through DeFi protocols which serve as decentralized investment funds, where tokens are deposited into a smart contract and an investment strategy that entails transacting with other DeFi protocols (e.g., PLFs) is encoded in the contract. Yield in DeFi is generated through interest (including accrued fees earned) and token rewards. For the latter, a protocol (e.g., PLF or AMM) distributes native tokens to its liquidity providers and/or users as rewards for the provision of deposits and/or protocol adoption.

  These protocol-native token rewards are similar to equity in the sense that they serve as a right to participate in the protocol's governance, as well as often represent a claim on protocol-generated earnings. The distribution model for token rewards in exchange for supplied liquidity may vary across protocols, yet is commonly proportional to how much liquidity an agent has supplied on a protocol. Therefore, smart contractencoded investment strategies of on-chain assets are tailored around yield generating mechanisms of different protocols with the sole aim of yield aggregation and maximization. In practice, on-chain management of assets may range from automatic rebalancing of a token portfolio [79] to complex yield aggregating strategies [80].
- **Privacy-preserving Mixers:** Mixers are methods to prevent the tracing of cryptocurrency transactions. These are important to preserve user privacy, as the transaction ledger is otherwise public information; however, this also means they could be used to obscure the source of illicit funds. Mixers work by developing a

"shielded pool" of assets that are difficult to trace back before entering the pool. They typically take one of two forms: (1) mixing funds from a number of sources so that individual coins can't easily be traced back to address individually (also called a "coinjoin", e.g., [90]), or (2) directly shielding the contents of transactions using zero knowledge proofs of transaction validity (e.g., [91], [92]). Mixers serve as a DeFi-like application itself and additionally as a piece that could be included within other DeFi protocols.

CHAPTER 11

DEFI PRIMITIVES

DeFi protocols require an underlying distributed ledger such as a blockchain, a peer-to-peer distributed append-only record of transactions. We take the underlying distributed ledger layer solely as an input into DeFi and refer the reader to existing work (notably [12], [13], [14], [15]) for a fuller exposition of the blockchain layer itself. We assume that the ledger has the basic security properties of consistency, integrity and availability [16]. Without these security properties, DeFi protocols built on top of such a ledger would themselves become inherently insecure. In this section, we draw attention to and outline the essential features of the underlying blockchain layer which have particular relevance to the security of DeFi protocols.

A. Smart Contracts: The most important provision is that the underlying ledger offers the ability to use smart contracts. These are programs that encode a set of rules for processing transactions which are enforced by a blockchain's consensus rules, allowing for trustless economic interactions between parties. Smart contracts rely on blockchains that are transaction-based state machines, whereby an agent can interact with smart contracts via transactions. Once a transaction is confirmed, the contract code is run by all nodes in the network and the state is updated. The underlying cost to state updates comes in the form of transaction fees charged to the sender. For instance, the Ethereum Virtual Machine (EVM) [17] on the Ethereum [18] blockchain is a stack machine which uses a specific set of instructions for task execution. The EVM maintains a fixed mapping of how much gas, an Ethereum-specific unit that denominates computational cost, is consumed per instruction. The total amount of gas consumed by a transaction is then paid for by the sender [19], [20].

In order for DeFi protocols to function on top of them, smart contracts must:

• be expressive enough to encode protocol rules

• allow conditional execution and bounded iteration

• be able to communicate with one-another within the same execution context (typically a transaction)

• support atomicity, i.e., a transaction either succeeds fully (state update) or fails entirely (state remains unaltered), such that no execution can result in an invalid state.

In relation to DeFi, the most notable property of smart contracts is that they are able to call each other via message calls. This makes possible composability: smart contracts can be

snapped together like Lego bricks ("Money Lego" [21]), with the possibility of building complex financial architectures. This is similar to as was envisaged in [22]. While promising, the side-effects of smart contracts interactions and the space of all possible interactions is vast. In a setting focused on financial applications, such complexity brings with it a great burden to understand the emergent security properties of composed smart contracts. We discuss this in more detail in Sections IV and V.

B. Tokens:

A common use of smart contracts is to implement tokens, which can be used to represent assets, ranging from Ether [23] and other cryptoassets [24] to synthetic assets or derivatives [25], as well as provide some utility, such as the right to participate in an election. Tokens are implemented by contracts adhering to a standard token interface, allowing protocols to easily handle different tokens without having to know about their implementation in advance. In Ethereum, tokens are usually implemented via the standardized ERC-20 [26] and ERC-721 [27] interfaces for fungible and non-fungible tokens, respectively [28], although other token standards exist [29], [30], [31]. A common distinction is between fungible tokens, which are interchangeable [26], and non-fungible tokens which are distinct [27].

C. Transaction Execution

When a blockchain network participant wishes to make a transaction, the details of the unconfirmed transaction (e.g., transaction cost, sender, recipient, data input) are first broadcast to a network of peers, validated, and then stored in a waiting area (the mempool of a node). This mempool is then propagated among the network nodes. Participants of the underlying ledger responsible for ensuring consensus, miners, then choose which transactions to include in a given block, based in part on the transaction fee attached to each transaction. Transactions in a block are executed sequentially in the order in which the miner of the respective block included them. For a detailed treatment of how this process works, we refer the reader to [1], [17], [32], [33]. Miners' ability to control the sequence in which transactions are executed means that miners can order transactions in ways that will earn them revenues. In addition, they can insert their own transactions to extract further revenues. Miners can even be bribed to undertake such transaction re-ordering [34], [35]. The value that miners can extract is known as Miner Extractable Value (MEV) [36]. We consider these issues in detail in Sec. V-B.

D. Keepers: Protocols may rely on their on-chain state being continually updated for their security. In transaction-based systems, updating the on-chain state requires transactions that are triggered externally. Since smart contracts are not able to create transactions programmatically, protocols must rely on external entities to trigger state updates. These entities, keepers, are generally financially incentivized to trigger such state updates. For instance, if for whatever reason a protocol requires a user's collateral to be automatically liquidated under certain conditions, the protocol will incentivize keepers to initiate transactions to trigger such liquidation.

E. Oracles: An oracle is a mechanism for importing off-chain data into the blockchain virtual machine so that it is readable by smart contracts. This includes off-chain asset prices, such as ETH/USD, as well as off-chain information needed to verify outcomes of prediction markets. Oracles are relied upon by various DeFi protocols (e.g. [37], [38], [39], [40], [41]). Oracle mechanisms differ by design and their risks, as discussed in [42], [43]. A centralized oracle requires trust in the data provider and bears the risk that the provider behaves dishonestly should the reward from supplying manipulated data be more profitable than from behaving honestly. Decentralized oracles offer an alternative. As the correctness of off-chain data is not verifiable on-chain, decentralized oracles tend to rely on incentives for accurate and honest reporting of off-chain data. However, they come with their own shortcomings. We provide a detailed overview of oracle manipulation risks and on the shortcomings of on and off-chain oracles in Sections IV-B and V-D.

F. Governance:

Governance refers to the process through which a system is able to effect change to the parameters which establish the terms on which interactions between participants within the system take place [42]. Such changes can be performed either algorithmically or by agents. While there is existing work on governance in relation to blockchains more broadly (e.g. [44], [45], [46]), there is still a limited understanding of the properties of different mechanisms that can be used both for blockchains and DeFi. Presently, a common design pattern for governance schemes is for a DeFi protocol to be instantiated with a benevolent dictator who has control over governance parameters, with a promise made by the protocol to eventually decentralize its governance process. Such decentralization of the governance process is most commonly pursued through the issuance of a governance token (e.g. [47], [48], [49], [50]), an ERC-20 token which entitles token holders to participate in protocol governance via voting on and possibly proposing protocol updates. We return to governance in Sec. V.

CHAPTER 12

WHAT ARE FLASH LOANS?

Flash loans enable users to borrow instantly from decentralized credit protocols such as Aave and dYdX with no collateral required, provided that the liquidity is returned at the end of the same transaction (which may contain a series of actions). If this does not happen, the whole transaction is automatically reversed. Flash loans can be used to swap the collateral supporting a loan, to liquidate a liquidity pool on a decentralized exchange in order to refinance a loan, or to conduct arbitrage by exploiting the priced differences of an asset across various exchanges. They can also be used to exploit protocols for market manipulation, essentially creating artificial leverage. For example, in October 2020, an attacker siphoned off $24 million of tokens through Harvest Finance, a DeFi robo-advisor, by using large flash loans repeatedly to manipulate stablecoin oracle prices.

Borrowing and lending are central to finance, because they facilitate risk-taking and expand the supply of capital through leverage. The classic form of centralized credit provision is banking. The bank manages the spread between the interest rates it pays to depositors, whose assets are liquid, and the rates it receives from borrowers on longer-term loans. It must assess credit-worthiness of borrowers and set interest rates appropriately to account for defaults. By contrast, DeFi credit protocols such as Compound and Aave pool together tokens, subject to an interest rate determined by the ratio of supply to borrowing. When lenders commit capital to DeFi credit services, they receive platform-native tokens representing their tokens plus the specified interest rate. Net of transaction fees—which accrue to service providers—lenders receive and borrowers pay the same interest rate, which is typically variable (some platforms now purport to offer fixed rates as well). Both sides maintain full custody over their assets and the ability to liquidate at any time. Credit terms can be quite complex, and these instruments can themselves be securitized and traded

CHAPTER 13

TECHNICAL SECURITY

We define a DeFi security risk to be technical if an agent can generate a risk-free profit by exploiting the technical structure of a blockchain system, for instance, the sequential and atomic execution of transactions. In current blockchain implementations, this coincides with (1) manipulating an onchain system within a single transaction, which is risk-free for anyone, and (2) manipulating transactions within the same block, which is risk-free for the miner generating that block. By exploiting technical structure, the underlying blockchain system allows no opportunity for markets or other agents to act in the course of such exploits. We identify three categories of attacks that fall within technical security risks of DeFi protocols: attacks exploiting smart contract vulnerabilities, attacks relying on the execution order of transactions in a block, as well as attacks which are executed within a single transaction.

- **Smart Contract Vulnerabilities:** Smart contracts being at the center of any DeFi protocol, any vulnerabilities in their implementation can cause them to be at loss. Smart contract vulnerabilities have been extensively discussed in the literature [93], [94], [95] and we will therefore not give an extensive list of all the known vulnerabilities but rather focus on the one which have already been exploited in the DeFi context.

  - **Reentrancy:** A contract is potentially vulnerable to a reentrancy attack if it delegates control to an untrusby calling it with a large enough gas limit, while its state is partially modified [96]. A trivial example is a contract with a withdraw function that checks for the internal balance of a user, sends them money and updates the balance. If the receiver is a contract, it can then repeatedly re-enter the victim's contract to drain the funds. Although this attack is already very well-known, it has been successfully used several times against DeFi protocols. We briefly present two of these attacks in more detail.
  - **dForce:** One of the most prominent examples of this exploit was against the dForce protocol [97], which features a PLF, in April 2020 to drain around 25 million USD worth of funds [98]. The attacker used imBTC [99], which is an ERC777 token [29], to perform the attack. A particularity of ERC-777 tokens, as opposed to ERC-20 tokens, is that they have a hook calling the receiver when the receiver receives funds. This means that any ERC-777 tokens will indirectly result in the receiver having

control of the execution. In the dForce attack, the attacker used this reentrancy pattern to repeatedly increase their ability to borrow without enough collateral to back up their borrow position, effectively draining the protocol's funds.

- **imBTC Uniswap Pool:** Despite the fact that Uniswap does not support ERC-777 tokens [61], an imBTC Uniswap [3] pool worth roughly 300 000 USD was drained using the reentrancy attack. Both of these attacks show a common attack pattern in DeFi applications: identifying and exploiting attack vectors which exploit protocols' interconnectedness, where the composability risks therein are often under-examined. In practice, reentrancy vulnerabilities are generally simple to detected and fix by using static analysis tools [95], [100]. There are two main ways to prevent this vulnerability: (1) using a reentrancy guard that prevents any call to a given function until the end of its execution or (2) finalizing all the state updates before passing execution control to an untrusted contract.

➢ **Integer Manipulation.** Almost every DeFi application manipulates monetary amounts in some way or another. This often involves not only adding and subtracting to balances but also converting into different units or to different currencies. We present the two most common types of integer manipulation issues. The first issue, which has been extensively studied in the literature [101], [102], is integer over- and underflow. The EVM does not raise any exception in case of over- or underflow and without correct checks, such overflows could stay undetected until the value is used in some sort of action such as, for example, a transaction sending a token amount. This will often result in failed transactions and cause the smart contract to misbehave [94]. The second issue is unit error during integer manipulation. While unit manipulation should in principle be a trivial task, limitations in the expressivity of both the programming language and the virtual machine, as well as poor development practices have caused issues related to this type of arithmetic operations. The main language used to develop DeFi applications at the time of writing is Solidity [103], which has a limited type system and no support for operator overloading. In addition, the EVM only supports a single type, 32-byte integers, and has no built-in support for fixed-point numbers. To work around this limitation, each protocol decides on an arbitrary power of 10 to use as its base unit, often 1018, and all the computations are performed in terms of this unit. However, given the limitations of the type-system, most programs elect to use exclusively 32-byte integers. Arithmetic on two units accidentally on different scales would not be caught by the compiler.

CHAPTER 14

PRELIMINARY EVALUATION

We conducted a preliminary evaluation on BLOCKEYE to validate its effectiveness in finding oracle-dependent state updates. Specifically, we considered eight DeFi projects on Ethereum, i.e., bZx, DDEX, Aave, dYdX, Compound, Nuo, Oasis, and Eminence. In Table I, we present a comparison of BLOCKEYE with Codefi Inspect [8] in oracle-dependent state update detection. The results show that BLOCKEYE successively identifies all vulnerable DeFi's with no false positive or false negative alert. Whereas, Codefi Inspect falsely ignores vulnerabilities in DDEX, leading to a false negative (FN) result.

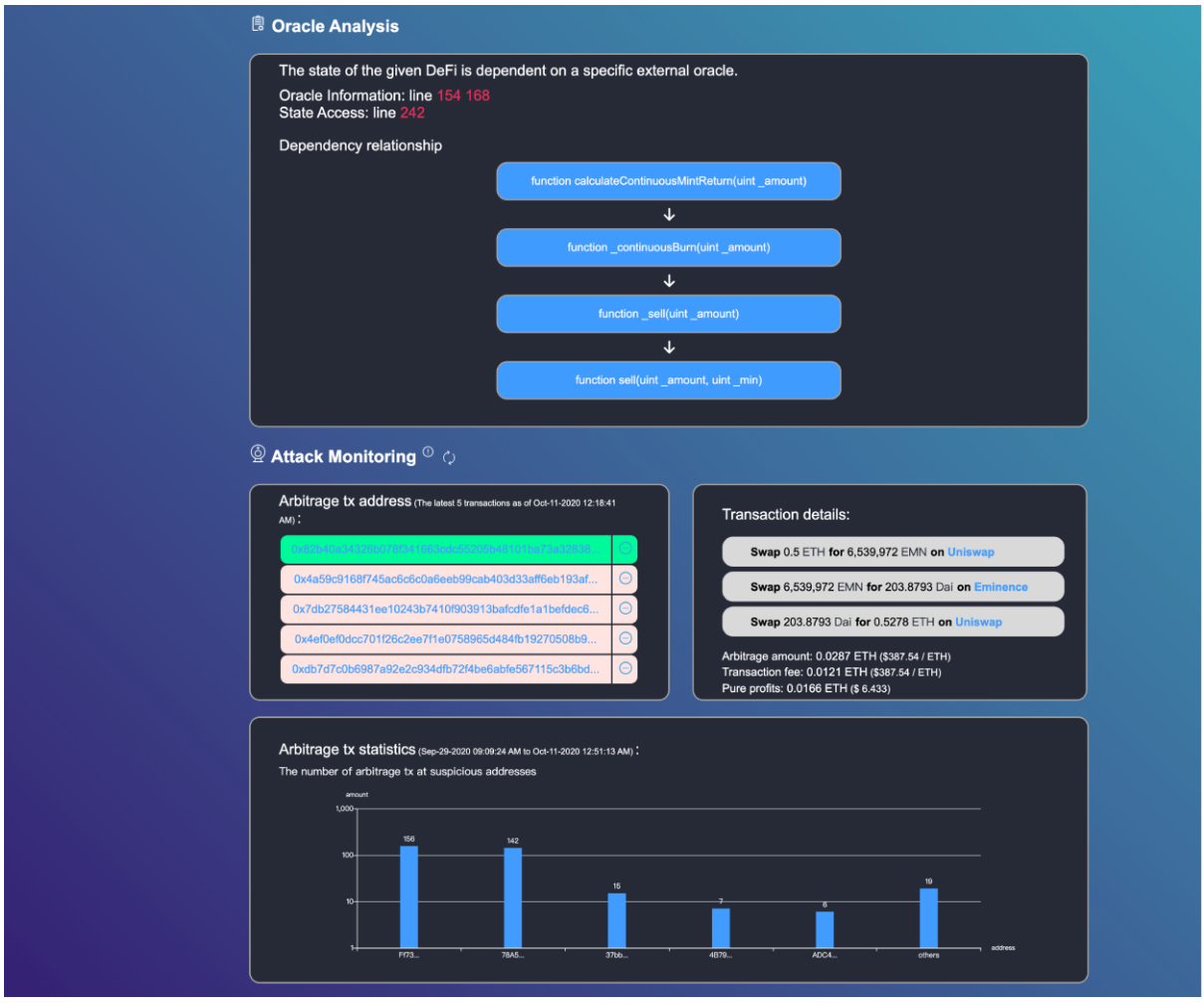

Fig. 14.1: The output interface for BLOCKEYE.

- **TABLE 1: A comparison of BLOCKEYE and Codefi Inspect**

A comparison of BLOCKEYE and Codefi Inspect in oracle-dependent state update detection. TP: True Positive; TN: True Negative; FN: False Negative; N/A: Not Available.

DeFi	Codefi Inspect	BLOCKEYE
bZx	TP	TP
DDEX	FN	TP
Aave	TN	TN
dYdX	TN	TN
Compound	TN	TN
Nuo	N/A	TN
Oasis	N/A	TN
Eminence	N/A	TP

We further evaluated BLOCKEYE with real-world transactions on the Ethereum main net. In Table II, we show detailed results of detected arbitrage transactions in two DeFis, i.e., ETH/sUSD token pair on bZx and DAI/EMN on Eminence

- **TABLE 2**: **Detailed results of BLOCKEYE with two DeFis.**

DeFi	bZx(ETH/sUSD)	Eminence(DAI/EMN)
Block	10799704 ~10950575	10956504 ~11031087
# Tx	_ 7138	_ 1486
# Suspicious Tx	slippage > 0.05 25 slippage > 0.07 23	slippage > 0.05 124 slippage > 0.057 107

For example, as for bZx (ETH/sUSD), there were around 7, 138 valid transactions detected between block 10, 799, 704 and 10, 950, 575. By enforcing different slippage thresholds, BLOCKEYE found 19 to 25 suspicious arbitrage transactions, e.g., 19 for slippage threshold 0.1 and 25 for slippage threshold 0.05. Besides, for Eminence(DAI/EMN), the number of suspicious transactions found ranged from 37 to 124 with different slippage thresholds, where overall valid transactions detected were 1, 486 between block 10, 956, 504 and 11, 031, 087.

CHAPTER 15

DEFI: USE CASE AAVE

Aave is an open source and non-custodial liquidity protocol for earning interest on deposits and borrowing assets [17]. Aave is a decentralized application that runs on Ethereum blockchain.

In essence, Aave is a liquidity protocol that operates solely via smart contracts on Ethereum. Loans are not individually matched in Aave, instead loans rely on pooled funds, as well as the amounts borrowed and collateral posted [3]. This enables instant loans on Aave [3]. Aave can be considered as an open lending protocol, providing a market (protocol) for loanable funds, where the role that an intermediary would play in traditional finance has been replaced by a set of smart contracts [34].

Borrowers using Aave must lock-up collateral that is a greater than the value of the loan they are taking out from ‘reserves’ in a lending pool. All borrowing positions in Aave are backed by collateral. Lending pool ’reserves’ accept deposits from lenders, who are using Aave to earn interest on their deposits. When a lender deposits into Aave, they receive an interest earning representation of their deposit, known as an ’aToken’. The amount of interest a lender earns on their deposit (represented as an aToken) is determined algorithmically in a smart contract and is based on supply and demand for that asset. For example, for borrowers in Aave, the cost of money for an asset at any particular time is dependent upon the amount of funds that have been borrowed from a ‘reserve’ for that asset. If the ‘reserves’ of an asset are low from high borrowing, a high interest rate will ensue for that asset

DeFi is a new, fast-growing area. Yet it remains immature, with a variety of unresolved economic, technical, operational, and public policy issues that will be important to address. Although some protocols have attracted significant capital and the associated network effects in a short period of time, the DeFi sector remains volatile. DeFi has the potential to transform global finance, but activity to date has concentrated on speculation, leverage, and yield generation among the existing community of digital asset holders. In addition, the very flexibility, programmability, and composability that make DeFi services so novel also expose new risks, from hacks to unexpected feedback loops among protocols. Retail investors, professional traders, institutional actors, regulators, and policy-makers will need to temper

enthusiasm for the innovative potential of DeFi with a clear understanding of its challenges. Developers are actively working to address vulnerabilities and introduce new mechanisms to manage risks efficiently, but the process is ongoing. DeFi will ultimately succeed or fail based on whether it can fulfill its promise of financial services that are open, trust-minimized, and non-custodial, yet still trustworthy.

The biggest difference between Aave and a traditional bank now is the maturity of unsecured lending. It is uncertain if DeFi protocols are actually in direct competition with banks at all. Gudgeon et al. noted that protocols for loanable funds are not directly acting as a fully-fledged replacement for banks [34], because traditional banks are not intermediaries of loanable funds: rather, they provide financing through money creation [37]. In Aave, loans are predominantly secured (over-collaterised), there is no money creation as the borrower must post collateral that is greater than the loan they are taking out. Aave requires an over-collaterisation of a loan to mitigate borrower default risk for lenders who have deposited reserves into a lending pool. Lenders are the ones who have provided liquidity into the reserves that a borrower utilises and are directly exposed to the risk of borrower default. To mitigate risks for lenders, Aave has public risk framework documentation, which analyses the fundamental risks of the protocol and describes the processes in place to mitigate them [12]. Aave's risk framework documentation focuses on the risk assessment for currencies supported by Aave (e.g. value/risk trade-offs of adding assets, market/counterparty/smart contract risks of existing assets, risk assessments to quantify risks per factor of assets and a risk quantification criterion). Aave uses risk parameters to mitigate market risks of the currencies supported by the protocol [12]. Each asset in the Aave protocol has specific values related to their risk, which influences how they are loaned and borrowed [12]. Aave's risk assessment methodology uses historical data to quantify market, counterparty and smart contract risks of an asset. The historical data is then computed through a risk quantification algorithm, created by Aave, resulting in risk ratings for sub-factors of assets ranging from A+ to D-. After retrieving sub-factor risks of an asset, Aave then aggregates the average of sub-factor risks to find one overall risk rating of an asset.

CHAPTER 16

OPEN RESEARCH CHALLENGES

There are many open research challenges in DeFi stemming from the technical and economic security issues presented in Sections IV and V.

A. Composability Risks Cryptoassets can be easily and repeatedly tokenized and interchanged between DeFi protocols in a manner akin to rehypothecation. This offers the potential to construct complex, inter-connected financial systems, yet bears the danger of exposing agents to composability risks, which are as of yet mostly unquantified. An example of composability risk is the use of flash loans for manipulating instantaneous AMMs and financially exploiting protocols that use those AMMs as price feeds. This has repeatedly been exploited in past attacks (e.g. [10], [158], [130]). Many protocols still struggle to implement sufficient protective measures for addressing this risk. The breadth of composability risks spans far beyond the negative externalities stemming from instantaneous AMM

manipulations. For instance, there remain open questions about the consequences of the following types of exploitations on connecting systems: the accumulation of governance tokens to execute malicious protocol updates, the failure of non-custodial stablecoin incentives to ensure price stability, and failure of PLF systems to remain solvent. Note, however, that this list is far from exhaustive. These become increasingly important issues as more complex token wrapping structures [159] stimulate higher degrees of protocol interconnectedness. For example, the use of PLF deposit tokens (as opposed to the tokens in their original forms) within AMM pools and strategies to earn yield on underlying assets through leverage by borrowing non-custodial stablecoins and depositing into PLFs or AMMs. Recent works [71], [160] begin to explore protocol interdependence, while [161] propose a process-algebraic technique that allows for property verification by modeling DeFi protocols in a compositional manner. Nonetheless, a critical gap in DeFi research toward taxonomizing and formalizing models to quantify composability risks remains. This problem is elevated as a holistic view on the integrated protocols is necessary: failures might arise from both technical and economic risks. Ensuring safety of protocol

composition will be close to impossible for any protocol designer and forms a major challenge for DeFi going forward.

B. Governance: We identify important research directions in governance and GEV. A general direction is modeling incentive compatibility of governance in various systems with GEV. For instance, setting up models, finding equilibria, and understanding how other agents in the system respond. The models in [42] get this started in the context of stablecoins and additionally discuss how to extend to other DeFi protocols. There is moreover a range of discussions around simulating and formalizing governance incentives through tools like cadCAD [162]. There remain unanswered questions with regards to the general design of governance incentives. For instance, how to structure governance incentives to reward good stewardship: e.g., intrinsic vs. monetary reward, reward per vote vs. reward per token holder, and measures of good stewardship. Furthermore, there lies potential in formally evaluating protection of minority agents in systems with flexible governance and large GEV. For systems utilizing governance tokens, we identify research gaps rooted in security risks of the ability to borrow governance tokens via flash loans and PLFs. Specifically, we identify opportunities to formally explore how (1) technical security can be compromised and (2) from an economic security point of view, incentive compatibility is further complicated by the borrowing of governance tokens.

C. Oracles We highlight a few open challenges about oracle design and security. Note that, in many cases, the oracle problem can also be directly related to the governance problem, as typically governors are tasked with choosing the oracles that are used. A more general open challenge lies in how to structure oracle incentives to maintain incentive compatibility to report correct prices. This is similar to governance design in some ways and needs to take into account the possible game theoretic manipulations that could be profitable.

CHAPTER 17

DATA AVAILABILITY STATEMENT

For ethical considerations, experimental data used in our work will be publicly available after discussions with the relevant DeFi development team. We point out the crux of this kind of arbitrage (i.e., making profits by buying and selling goods at different prices) is for the attacker successfully controlling exchange rates of cryptoasset pairs, ETH/sUSD here in Figure 1, by exploiting data dependencies of bZx on Uniswap and Kyber.

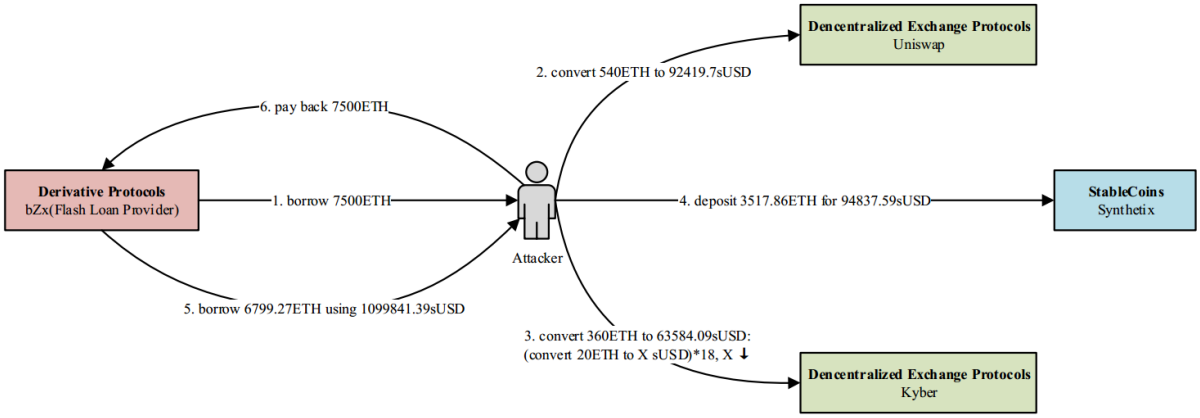

Fig. 17.1: Attack on the bZx project.

While many previous research works and tools have focused on the security of smart contracts [2], [3], [4], [5], there is relatively little study on the security of DeFi projects as aforementioned. In general, detection of these attacks requires a deep understanding on both the business nature of a DeFi project as well as the market it is involved in, which are missing in existing solutions to finding low-level bugs of smart contracts.

CHAPTER 18

THE FUTURE OF DEFI

Within and beyond the categories described here, DeFi is evolving rapidly. Developers are experimenting with new services, business models, and combinations of DeFi protocols Technologies are maturing. Services are moving to decentralized management and governance of protocols. Tools are emerging to simplify the user experience on and across DeFi services. A significant aspect of ongoing DeFi development will involve composition of Dapps and financial primitives as "Money Legos." Already, aggregator services are emerging. DeFi composability might create new financial instruments and services, as well as new risks due to unanticipated interaction effects. The investor population will also change as both less-sophisticated participants with more-limited cryptocurrency experience and more-sophisticated institutional traders enter in greater numbers. Speculative demand bubbles that produced spectacular returns over short periods will not be sustainable. Regulators will engage more actively in the DeFi area, especially as financial institutions and centralized finance providers look to become involved. While we cannot offer a crystal ball to predict the future of DeFi, we highlight some significant recent developments.

- **RISK MANAGEMENT INNOVATIONS:** There is growing demand for better tools to repackage and redistribute risk associated with DeFi activities, allowing more efficient capital allocation and more complex derivatives.

• **Options:** form the basis for a wide range of hedging strategies in finance. Opyn is a DeFi service for creating tokenized options, which can be used to hedge against risks or take speculative positions. It also supports flash mints, analogous to flash loans: options without collateral that are burned before the end of the transaction. Ribbon Finance further supports Structured Products that combine options with other instruments for even more complex strategies.

• **Tranched Lending:** under development by DeFi service BarnBridge, separates debt pools into tranches of assets with different risk/reward characteristics, which investors can access separately.

• **Credit Default Swaps**: allow investors to purchase insurance against default risk of credit arrangements. Saffron Finance is working on enabling this mechanism in DeFi, where users are able to trade swaps on the underlying lending platform.

• **Reinsurance:** is traditionally how insurance companies themselves diversify risks. NexusMutual and other DeFi insurance players have begun extending smart contract insurance to each other, mimicking these arrangements.

- **SCALING INNOVATIONS:** Ethereum in its current form is slow and suffers from high transaction fees, known as gas prices. Other blockchains such as Algorand, Avalanche, Binance Smart Chain, Cosmos, EOS, NEAR, Polkadot, and Solana are trying to attract DeFi-focused developers and users with promises of higher throughput and lower fees. However, better scalability at the base layer may come at a cost in the degree of decentralization or other attributes. The Ethereum Foundation promises significant scalability improvements in the upcoming Eth2, while Ethereum developers have been building a variety of Layer 2 solutions, such as sharding or "rollups," that offload computation execution, but keep some transaction data on-chain. Eth2 is also scheduled to replace proof-of-work mining, which has been criticized for intensive energy usage, with a proof-of-stake system.
- Security problems of smart contracts have been widely discussed in recent years [4], [5], [3], [2], [6]. Luu et al. highlighted four types of vulnerabilities for smart contracts [2]. Tsankov et al. proposed a verification technique [3], which transforms Ethereum smart contracts into Data log logics [9]. Permenev et al. further presented their solution to verify smart contracts in an inductive manner [10]. In addition to security problems, Liu et al. proposed a statistical approach to identify potential code smells [5]. Security of DeFi projects is relatively less discussed in previous works. Several mathematical and economic models were proposed to help understand risks of DeFi in a theoretical manner [11], [12], [13], [14].

CONCLUSION

In this paper, we highlighted BLOCKEYE as an open platform to detect DeFi attacks on blockchain. Compared to existing analyzers for smart contracts, BLOCKEYE provides important capabilities to model dependency among DeFi projects and flag potential end-to-end attacks at real-time. The key insights behind BLOCKEYE are symbolic oracle analysis and pattern-based runtime transaction validation. We applied BLOCKEYE in several popular DeFi projects on Ethereum and managed to find potential attacks previously unreported.

In this SoK we have considered DeFi from two points of view, the DeFi Optimist and the DeFi Pessimist, and examined the workings of DeFi systematically and at length. First, we laid out the primitives for DeFi before categorizing DeFi protocols by the type of operation they provide. We examined the security challenges protocols are exposed to by making a distinction between technical and economic security risks. In so doing, we were able to systematize attacks that have been proposed in theory and/or occurred in practice into categories of attacks that either rely on an agent's ability to generate risk-free profits by exploiting the technical structure of a blockchain or to game the incentive structure of a protocol to obtain a profit at the expense of the protocol. Finally, we drew the attention to open research challenges that require a holistic understanding of both the technical and economic risks.

REFERENCE

[1] G. Wood, “Ethereum: A secure decentralised generalised transaction ledger,” Ethereum Project Yellow Paper, vol. 151, 2014.

[2] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, and A. Hobor, “Making smart contracts smarter,” in Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2016, pp. 254–269.

[3] P. Tsankov, A. Dan, D. D. Cohen, A. Gervais, F. Buenzli, and M. Vechev, “Securify: Practical security analysis of smart contracts,” arXiv preprint arXiv:1806.01143, 2018.

[4] C. Liu, H. Liu, Z. Cao, Z. Chen, B. Chen, and B. Roscoe, “Reguard: finding reentrancy bugs in smart contracts,” in ICSE (Companion). ACM, 2018, pp. 65–68.

[5] H. Liu, C. Liu, W. Zhao, Y. Jiang, and J. Sun, “S-gram: towards semantic-aware security auditing for ethereum smart contracts,” in ASE. ACM, 2018, pp. 814–819.

[6] Z. Yang, H. Liu, Y. Li, H. Zheng, L. Wang, and B. Chen, “Seraph: enabling cross-platform security analysis for evm and wasm smart contracts,” in Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Companion Proceedings, 2020, pp. 21–24.

[7] “Microsoft z3 smt solver,” https://z3.codeplex.com/, 2019.

[8] “Codefi inspect,” https://inspect.codefi.network/, 2020.

[9] T. Eiter, G. Gottlob, and H. Mannila, “Disjunctive datalog,” ACM Transactions on Database Systems (TODS), vol. 22, no. 3, pp. 364–418, 1997.

[10] A. Permenev, D. Dimitrov, P. Tsankov, D. Drachsler-Cohen, and M. Vechev, “Verx: Safety verification of smart contracts,” Security and Privacy, vol. 2020, 2019.

[11] K. Qin, L. Zhou, B. Livshits, and A. Gervais, “Attacking the defi ecosystem with flash loans for fun and profit,” arXiv preprint arXiv:2003.03810, 2020.

[12] J. Kamps and B. Kleinberg, “To the moon: defining and detecting cryptocurrency pump-and-dumps,” Crime Science, vol. 7, no. 1, p. 18, 2018.

[13] B. Liu and P. Szalachowski, “A first look into defi oracles,” arXiv preprint arXiv:2005.04377, 2020.

[14] L. Gudgeon, D. Perez, D. Harz, A. Gervais, and B. Livshits, “The decentralized financial crisis: Attacking defi,” arXiv preprint arXiv:2002.08099, 2020.