SoK: Decentralized Finance (DeFi) Attacks

Liyi Zhou [†*], Xihan Xiong [†*], Jens Ernstberger [‡], Stefanos Chaliasos [†], Zhipeng Wang [†],
Ye Wang [§**], Kaihua Qin [†], Roger Wattenhofer [¶], Dawn Song [‖], and Arthur Gervais [†‖]
[†]Imperial College London [‡]Technical University of Munich [§]University of Macau [¶]ETH Zurich [‖]UC Berkeley

***Abstract*—Within just four years, the blockchain-based Decentralized Finance (DeFi) ecosystem has accumulated a peak total value locked (TVL) of more than 253 billion USD. This surge in DeFi's popularity has, unfortunately, been accompanied by many impactful incidents. According to our data, users, liquidity providers, speculators, and protocol operators suffered a total loss of at least 3.24 billion USD from Apr 30, 2018 to Apr 30, 2022. Given the blockchain's transparency and increasing incident frequency, two questions arise: How can we systematically measure, evaluate, and compare DeFi incidents? How can we learn from past attacks to strengthen DeFi security?**

In this paper, we introduce a *common reference frame* to systematically evaluate and compare *DeFi incidents*, including both attacks and accidents. We investigate 77 academic papers, 30 audit reports, and 181 real-world incidents. Our open data reveals several gaps between academia and the practitioners' community. For example, few academic papers address "price oracle attacks" and "permissonless interactions", while our data suggests that they are the two most frequent incident types (15% and 10.5% correspondingly). We also investigate potential defenses, and find that: *(i)* 103 (56%) of the attacks are not executed atomically, granting a rescue time frame for defenders; *(ii)* SoTA bytecode similarity analysis can at least detect 31 vulnerable/23 adversarial contracts; and *(iii)* 33 (15.3%) of the adversaries leak potentially identifiable information by interacting with centralized exchanges.

I. Introduction

Blockchain-based Decentralized Finance (DeFi) ecosystem has attracted a surge in popularity since the beginning of 2020. The peak total value locked (TVL) for DeFi surpassed 253 billion USD on Dec 2, 2021, with Ethereum (145 billion, 57% TVL) and BNB Smart Chain (19.8 billion, 8% TVL) sharing the majority of DeFi's activity [1]. While DeFi certainly provides many protocols inspired by traditional finance such as cryptocurrency exchanges [2]–[4], lending platforms [5], [6], and derivatives [7], novel constructs known as flash loans [8] and atomic composable DeFi trading [9] emerged. Unfortunately, these very intertwined DeFi systems, coupled with the already well-studied vulnerability-prone smart contracts [10]–[16], broadened the threat surface of DeFi protocols. We identify that from Apr 30, 2018 to Apr 30, 2022, so-called "DeFi incidents" have accumulated to a total loss of 3.24 billion USD. Particularly exciting to interdisciplinary scholars, these harmful incidents cover a wide variety of system layers, including the network, consensus, smart contract and DeFi protocol, as well as external auxiliary services such as off-chain oracles, cross-chain bridges, centralized exchanges etc. Understanding DeFi incidents hence requires a vertical understanding of all relevant system layers and architectures.

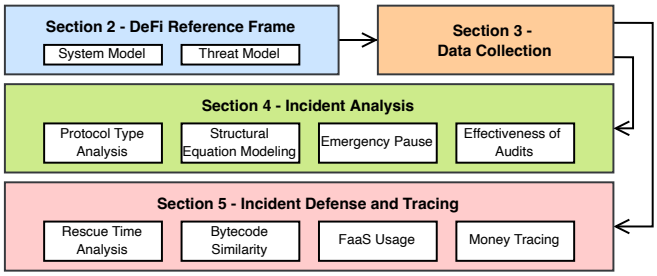


Fig. 1: Section II presents a DeFi reference frame, with a five layer system and threat model overview, allowing to categorize real-world incidents, academic works, and audit reports (cf. Section III). Section IV studies the collected DeFi incidents with statistical analysis. Section V shows how to identify adversarial and victim contracts, how to front-run adversaries, and how to trace adversarial funds. The paper concludes with a discussion in VI, related works in VII and a closure in VIII.

For the first time in history, the information security community has access to a transparent, broad, timestamped, and non-repudiable dataset of million-dollar security-related incidents. In this work, we leverage the blockchain as an open dataset and systematize our findings with the following contributions:

- **DeFi Reference Frame:** We provide the first framework for reasoning about DeFi system and threat models. We outline a wide spectrum of adversarial goals, assumptions, prior knowledge, capabilities, as well as common causes for potentially harmful DeFi incidents to create a standard model for related works (cf. Section II-A and II-B).
- **Gap Between Attackers and Defenders:** We analyze 181 DeFi incidents on Ethereum and BNB Smart Chain over a time frame of four years and structure the incidents, related academic papers, and security audit reports into a comprehensive taxonomy. We discover that academia and industry practices are underdeveloped with respect to the incident cause "unsafe DeFi protocol dependencies", when compared to the practices of in-the-wild adversaries.
- **Incident Defense:** We investigate possible defense mechanisms against DeFi incidents. We show that SoTA similarity analysis can detect vulnerable and adversarial contracts. For instance, we identify 31/23 exactly matching vulnerable/adversarial contracts (i.e., bytecode similarity score of 100%) when compared to previously known incidents. We also discover that 103 (56%) of the attacks are not executed

[*]Both authors contributed equally to the paper.
[**]Work done at ETH Zurich.

atomically, granting a rescue time frame for defenders.

- **Tracing Source of Funds:** By tracking pre-incident adversarial footprints, we discover that $12(7.3\%)$ and $21(8.0\%)$ of the adversaries directly withdraw funds from exchange wallets, on Ethereum and BNB Smart Chain, respectively. Similarly, $55(21\%)$ and $12(4.6\%)$ of the attack funds stem directly from the US-sanctioned Tornado Cash mixer.

II. DeFi Reference Frame

Bitcoin is the first widely adopted permissionless system to allow users to send and receive financial value without the use of a third-party intermediary [17]. While Bitcoin also introduced the concept of smart contracts, more developer-friendly smart contract primitives [18] empowered the wide adoption of DeFi. DeFi currently provides a wide range of financial services such as lending/borrowing, market-making, stablecoins, pegged tokens, price oracles [19], mixing services, flash loans, decentralized portfolio managers, insurance, etc [5], [6], [8], [20], [21]. *Flash loans* allow traders to instantaneously request access to cryptocurrencies worth billions of USD. This is achieved through the creative use of the blockchain's transaction atomicity property, through which a loan is not granted if the loan is not paid back with the due interests. Such convenient and programmable access to substantial capital has lowered the barrier of entrance for practical DeFi traders, as well as broaden the threat surface [8]. Because permissionless blockchains such as Bitcoin and Ethereum are known to not offer anonymity, but rather pseudonymity, alternative privacy-preserving blockchains emerged. These alternative blockchains break the linkability of addresses, by shuffling assets through an anonymity set. Notable solutions include ZCash which is based on zero-knowledge proofs [22], and Monero which is based on ring signatures and confidential transactions [23], [24]. Additionally, mixers operating as applications on existing blockchains emerged, such as Tornado Cash [25], Typhoon Network and AMR [26]. An extended background on DeFi, as well as a comparison to centralized finance (CeFi), is provided by related work [27]. In the following, we present a five-layer system model which is applicable to all DeFi incidents, as well as a threat model taxonomy based on various adversarial utilities, goals, knowledge, and capabilities.

A. System Model

As Figure 2 shows, our system model consists of five layers. The network layer enables data transmission between and among system layers. The blockchain consensus and smart contract layers enable financial services such as cryptocurrency trades to be performed without the use of trusted intermediaries. The protocol layer is a collection of DeFi protocols that are deployed and built on the smart contract layer. Note that on a permissionless blockchain, any DeFi user can create or deploy financial service protocols. Furthermore, DeFi protocols may rely on auxiliary services to increase the entire financial ecosystem's efficiency, stability, and usability. We proceed to introduce the key components in each layer:

(i) Network Layer (NET):

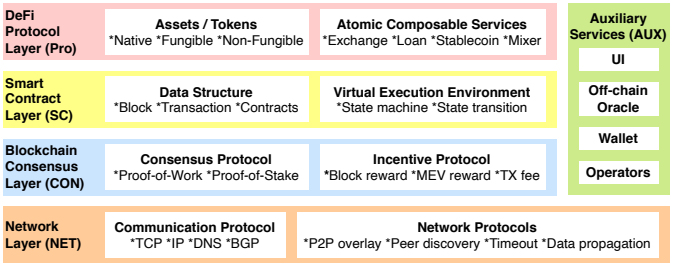


Fig. 2: High-level systematization of Decentralized Finance. DeFi is built on smart contract enabled blockchains, where auxiliary services help to ensure the overall efficiency, stability, and usability of the ecosystem. The network layer enables data transmission between and among system layers.

- **Network Communication Infrastructure:** A communication protocol is a set of rules that allows two or more nodes in a system to communicate over a physical medium [28]. Users must rely on communication protocols such as TCP/IP, DNS, and BGP to interact with DeFi, whether directly through their own blockchain nodes or indirectly through third-party auxiliary services.
- **Blockchain and Peer-to-Peer (P2P) Network:** Blockchain network protocols instruct nodes on how to join, exit, and discover other nodes in the P2P network. A blockchain node may become unresponsive at any point in time, and related works observed frequent node churn [29]. Blockchain networks typically instruct each node to connect with many peers while also configuring a timeout to disconnect from non-responsive peers to ensure the network's connectivity.
- **Front-running as a Service (FaaS):** Independent of the public blockchain P2P network, emerging centralized transaction propagation services offer an alternative option for traders to communicate to miners (e.g., Flashbots[1], Eden network[2], Bloxroute[3], and Ethermine[4]). FaaS services allow DeFi traders to submit a bundle that consists of one or more transactions directly to FaaS miners without a broadcast on the P2P network. FaaS services may in addition provide bundle-level atomic state transition[5], where the entire bundle is either executed successfully in the exact order that the transactions are provided, or fails collectively. Furthermore, FaaS traders are required to place a single sealed bid for the priority inclusion of the entire bundle, without observing the bid from other DeFi traders (i.e., sealed-bid auction). FaaS miners prioritize transaction bundles with the highest average bid at the top of the next mined block.

***(ii)* Consensus Layer (CON):**

- **Consensus Mechanism:** A consensus mechanism is a fault-tolerant mechanism in blockchain systems, which assist

[1]Flashbots: https://blocks.flashbots.net/
[2]Eden network: https://www.edennetwork.io/
[3]Bloxroute: https://bloxroute.com/
[4]Ethermine private RPC: https://ethermine.org/private-rpc
[5]This is different from transaction-level state transition in SC layer

blockchain nodes to achieve the required agreement on a single data value or network state. The blockchain consensus mechanism typically consists of the following components:
  - **A Sybil attack-resistant leader election protocol**, such as Proof-of-Work (PoW) for Ethereum or Proof-of-Stake (PoS) for BNB Smart Chain;
  - **A consensus protocol** to synchronize the latest chain state (e.g., the longest chain with most difficulty); and
  - **A CON incentive mechanism**, which aims to encourage benign consensus activity. The *block reward* for instance, compensates every successful block appended to the main chain. *Transaction fees* are paid by transaction issuers to sequencers for inclusion in specific blocks and positions, and, lastly *blockchain extractable value (BEV)* and *miner extractable value (MEV)*, is potential extractable revenue left untouched [9], [30]–[34]. Transaction fees are typically enforced to be paid in the native blockchain coin.
- **Nodes and Their Operation Protocol:** A blockchain node may be responsible for one or several tasks: *(i)* transaction sequencing, specifying the order of transactions within a block; *(ii)* block generation; *(iii)* data verification; and *(iv)* data propagation. The two common types are:
  - **Sequencer nodes**, also known as miners in PoW blockchains, or validators in PoS blockchains, capture all four of the above responsibilities. Sequencers can insert, omit and reorder transactions in blocks they generate within the scope allowed by the protocol;
  - **Ordinary nodes** only perform blockchain data propagation and may perform data verification.

(iii) Smart Contract Layer (SC):

Despite the existence of different data storage structures (e.g., directed acyclic graph [35], sharding [36]–[39], etc.), SoTA smart contract enabled blockchains order their transactions as a linear sequence in order to achieve deterministic state transition [40]. In the following, we denote non-generic SC components with the asterisk mark (*). The remaining SC components are applicable to any DeFi systems.

- **Transactions:** A user specifies financial operations within a transaction to request blockchain state transitions. SC layer typically supports transaction-level *atomic state transition*, where all financial operations within the same transaction either execute in their entirety, or fail collectively.
- **State:** DeFi system state S specifies: *(i)* the cryptocurrency asset balances of users, *(ii)* the blockchain information, such as timestamps, coinbase addresses, block numbers, block gas limits (maximum computation unit per block), as well as *(iii)* the DeFi application state.
- **State Transition:** $\mathcal{T}(s \in S, tx \in TX) \rightarrow S$ is the state transition function returning a new state after executing tx, where TX denotes the set of all valid DeFi transactions.
- **Smart Contract:** A smart contract is code that is translated into one or several state transition functions, which can then be triggered by a transaction. Smart contracts can also trigger the functions of other contracts. Upon deployment, a constructor function may initialize the contracts' state.
- **Block State Transition*:** Both Ethereum and BNB smart chain record transactions with an ordered list of blocks. We denote B as the set of blocks, and use $b_i \in B$ to denote a block at height i. Each block b_i may include a list of n transactions, denoted by $\{tx^0_{b_i}, \ldots, tx^n_{b_i}\}$, $n \geq 0$. A block state $\mathcal{S}(b_{i+1})$ stems from the sequential execution of all transactions in block b_{i+1} on $\mathcal{S}(b_i)$ (cf. Equation 1).

$$\mathcal{S}(b_{i+1}) = \mathcal{T}(\ldots \mathcal{T}(\mathcal{T}(\mathcal{S}(b_i), tx^0_{b_{i+1}}), tx^1_{b_{i+1}}) \ldots) \quad (1)$$

- **SC and Layer 2 Blockchain (L2) Incentive Mechanism*:** DeFi protocols can operate on so-called L2 systems, such as side-chains[6], commit-chains [41] or its inspired successor optimistic-rollups [42], and zk-rollups[7]. Because L2 systems are created on top of Layer 1 blockchains (also known as L1, e.g, Ethereum and BNB Smart Chain), L2 systems often implement their consensus incentive mechanisms on L1 blockchains' SC layer to encourage benign activities [43].

(iv) DeFi Protocol Design Layer (PRO):

- **Cryptocurrency Protocols:** DeFi supports a variety of asset standards, which define a common set of rules and interfaces for the transfer and approval of cryptocurrency assets (e.g., ERC-20 [44]). DeFi protocols may, however, deviate from the common standard by proposing a newer variant with domain-specific features. The Ampleforth protocol is an example of a custom asset standard, which dynamically adjusts its total token supply to maintain a stable price (i.e., stablecoins) [45]. Newer standards may remain backward compatible, while extending the feature set (e.g., ERC-777 enables the injection of state transitions, i.e., hooks, during transfer calls [46]). Note that backward-compatible standards may however violate the security assumptions of existing protocols, thus empowering novel attack vectors.
- **Financial Protocols:** While DeFi protocols may appear inspired by traditional financial services, the blockchains' unique features (e.g., transparency, atomicity, and discrete batch transaction execution) enable novel designs. For instance, unlike CeFi, DeFi platforms are notably intertwined through atomic composability. For instance, leveraged liquidity mining protocols such as Alpha Homora [47] and Harvest Finance [48] integrate automated market makers (i.e., Uniswap [2]) and lending platforms (i.e., Compound [6]).
- **Protocol Layer Incentive Mechanism:** DeFi protocols may introduce PRO incentive mechanisms to encourage desired user behavior. One example is the airdrop of governance tokens in exchange for providing liquidity in decentralized exchanges [49], [50] (e.g., Sushiswap[8] and Curve[9].).

(v) Auxiliary Service Layer (AUX):

[6]For example, Polygon network (https://polygon.technology/)
[7]For example, zkSync (https://zksync.io/)
[8]Sushiswap staking: https://app.sushi.com/stake
[9]Curve staking: https://resources.curve.fi/crv-token/staking-your-crv

	Capability Description	Knowledge
C^1_{NET}	$\mathbb{A}$ may control network service providers (e.g., DNS).	K_3
C^2_{NET}	$\mathbb{A}$ may manipulate incoming messages to deceive a node's perception of current state (e.g., eclipse attacks [51]).	K_1 or K_3
C^3_{NET}	$\mathbb{A}$ may censor or delay the transmission of messages. For example in selfish mining, $\mathbb{A}$ may not broadcast the blocks appended to the competing chain [52].	K_1 or K_3
C^4_{NET}	$\mathbb{A}$ may transmit transactions to miners using FaaS.	K_1
C^1_{CON}	$\mathbb{A}$ may fork or append on a forked chain in an attempt to catch up and overwrite the longest chain.	K_2
C^2_{CON}	$\mathbb{A}$ may censor mempool transaction temporarily.	K_2
C^3_{CON}	$\mathbb{A}$ may *(i)* include, exclude, or re-order transactions within its blocks if $\mathbb{A}$ is/colludes with a sequencer, or *(ii)* engage in front-/back-running [30], [31], [33].	K_1 or K_2
C^1_{SC}	$\mathbb{A}$ may simulate state transition off-chain (cf. Equation 1) with any arbitrary transactions on forked blockchain states, instead of issuing transactions on-chain.	K_1
C^1_{PRO}	$\mathbb{A}$ may use mixer services to break account linkability.	K_1
C^2_{PRO}	$\mathbb{A}$ may borrow, use, and return liquidity from a decentralized cryptocurrency pool within a single atomic transaction using a flash loan [8].	K_1
C^3_{PRO}	$\mathbb{A}$ may compose the state transition from multiple DeFi protocols (composability).	K_1
C^4_{PRO}	$\mathbb{A}$ may compose all state transitions required in one single transaction, and execute atomically.	K_1
C^5_{PRO}	$\mathbb{A}$ may deploy or utilise a customised contract, which mimics the function interface (i.e., abi) of one or many DeFi protocols.	K_1
C^1_{3RD}	$\mathbb{A}$ may manipulate external oracle data [14].	K_3
C^2_{3RD}	$\mathbb{A}$ may compromise the wallet passphrase of specific DeFi users, operators and etc.	K_3

TABLE I: Adversarial capabilities and knowledge level at each layer of our system model.

Raw on-chain data	Raw P2P network data	Public side channel	Public data analysis	Private mempool	Sequencing rules	Next block early access	Oracle update early access	External price early access	Wallet passphrase access	Other miscellaneous K_3	Knowledge
✔	✔	✔	✔	✘	Δ	✘	✘	✘	✘	✘	Public (K_1)
✔	✔	✔	✔	✔	✔	✔	✘	✘	✘	✘	Sequencer (K_2)
✔	✔	✔	✔	✘	✘	✘	Δ	Δ	Δ	Δ	Insider (K_3)

TABLE II: Categorization of adversarial knowledge levels. "✔" has access, "✘" cannot access, "Δ" may have access.

Auxiliary services refer to any entity that is required or which facilitates DeFi's efficiency, but does not belong to any of the four above-mentioned system layers (i.e., NET, CON, SC, and PRO). For example, an operationally active DeFi protocol implementation may consist of: *(i)* front-end code; *(ii)* project developers realizing the protocol designs; *(iii)* "operators" with administrative powers, such as the privilege to deploy the code, upgrade the protocol, freeze or cease the activity of the operative DeFi protocol; *(iv)* off-chain oracle services which sync price data from centralized exchanges to on-chain smart contracts, etc.

B. Threat Model Taxonomy

In the following we provide a holistic view of the adversarial utilities, goals, knowledge and capabilities, to engender a common reference frame which we subsequently apply in Section III to relatively compare all observed DeFi attacks.

***(i)* What is a DeFi Incident:** An incident refers to a series of actions that result in an unexpected financial loss to one or more of the following entities: (i) users; (ii) liquidity providers; (iii) speculators; or (iv) operators. We classify incidents into the following two types:

- **Attacks:** An adversary, $\mathbb{A}$, may exploit vulnerabilities, in an attempt to disable, delay, or alter a DeFi protocol's expected state transition. Despite the fact that vulnerabilities exist on all five system layers, DeFi vulnerabilities are most commonly found in the following three layers (cf. Table III):
  1) *SC Layer Vulnerabilities* result from coding mistakes, such as arithmetic error, casting error, inconsistent access control, function reentrancy, etc;
  2) *PRO Layer Vulnerabilities* may resemble financial market manipulation instead of traditional system vulnerabilities (i.e., protocol design flaws, such as unsafe external protocol dependency or interactions). Yet, the practitioners' community as well as related works [8] classify market manipulations as attacks, which necessarily require a vulnerable system or system state; and
  3) *AUX Layer Vulnerabilities*, which includes both operational vulnerability (e.g., off-chain oracle manipulation, compromised private key, etc.) and "information asymmetry" attacks (e.g., backdoor, honeypot, phishing, etc.). Generally speaking, we observe that users may not always (or may not be able to) inspect and understand a DeFi protocol smart contract before providing financial assets, let alone evaluating its security and risks [53], [54]. As such, a user's understanding of a contract operation may be mostly based on marketing communications, rather than the factual contract source code, leading the user to unforeseen or unexpected circumstances.
- **Accidents:** Any incident that does not explicitly involve proactive adversaries is classified as a DeFi accident. For example, a user's fund may become permanently locked in a contract due to unintentional coding mistakes.

***(ii)* Adversarial Utility and Goal:** Throughout this work, we assume that $\mathbb{A}$ is a rational agent aiming to maximize its utility. We categorize utility into the following two categories:

- U_1**-Monetary:** The most common utility we find is of monetary nature. We define the monetary utility function as the total increase in market value of $\mathbb{A}$'s cryptocurrency asset portfolio, which $\mathbb{A}$ aims to maximize.
- U_2**-Non-monetary:** $\mathbb{A}$ may instead maximize non-monetary utilities, such as sense of accomplishment, or reputation. DeFi white hat hackers (also known as ethical hackers) are an example of a non-monetary adversary, as they attack in an attempt to minimize the loss from DeFi incidents.

***(iii)* Adversarial Knowledge:** Table II differentiates between the following three types of adversarial knowledge.

- K_1**-Public:** $\mathbb{A}$ can access public information, including: *(i)* Raw on-chain data such as blocks, uncle blocks, trans-

actions, accounts, balances, and deployed smart contract bytecode; *(ii)* Raw network data, such as P2P network transactions, pending blocks, discarded stale blocks, blockchain node IP addresses, port numbers, client version strings, etc; *(iii)* Public side channel, such as, open-source smart contract code, social media/chat messages; *(iv)* Public data analysis, such as inferred network topology, estimated sequencer location, and decompiled smart contract bytecode [55].

- K_2**-Sequencer:** $\mathbb{A}$ obtains the following information, if $\mathbb{A}$ is/colludes with a sequencer: *(i)* Pending transactions from private communication channels; *(ii)* Transaction ordering logic for the corresponding sequencer, including bribery preferences; *(iii)* Early access to block state before broadcast if the corresponding sequencer generates the next block.
- K_3**-Insider:** Privileged information asymmetry may occur for example if $\mathbb{A}$ has early access to external market prices, oracle updates, or the wallet passphrases of an operator[10].

***(iv)* Adversarial Capabilities:** Table I outlines the adversarial capabilities and required knowledge. Note that $\mathbb{A}$ with differing levels of knowledge may be able to achieve the same capability. Sequencers, for example, can control the transaction order of their generated blocks (K_2), whereas $\mathbb{A}$ without sequencer knowledge can also perform front-/back-running by competing on the public blockchain P2P network (K_1).

III. Data

In this section we present our methodology to sample a dataset of "works under investigation", including research papers, security tools (i.e., intrusion detection, intrusion prevention and vulnerability detection), audit reports, and real-world incidents. We manually label which incident types each work addresses (cf. Table III and IX). Our dataset serves as the foundation for the analysis in Sections IV, V and VI.

Academic Papers: We identify relevant papers in eight of the top security, software engineering, and programming language conferences (i.e., SSP, CCS, NDSS, USENIX, ICSE, ASE, POPL, PLDI) from 2018 to 2021. Our methodology first crawls papers using Google Scholar's keyword search[11], and then performs backward and forward reference searches to find additional relevant works. Our dataset omits: *(i)* papers irrelevant to DeFi, such as Bitcoin specific attacks or Monero privacy; and *(ii)* DeFi related papers that do not address any particular type of incidents, such as contract patching [56], model checking [57], bug bounties [58], and reverse engineering [55]. In total, our dataset captures 7 relevant surveys and SoKs, 29 security tools, and 42 attack papers. We manually label the incident types addressed in each academic paper and cross-validate our labels against the related works section.

[10]See Section II-A for the definition of an operator.

[11]with at least one of the following keywords: ["smart contract", "Decentralized Finance", "DeFi", "automated market maker", "AMM","decentralized exchange", "DEX", "price oracle", "miner extractable value", "MEV", "blockchain extractable value", "BEV", "Ethereum", "ETH", "BNB Smart Chain", "Binance Smart Chain", "BSC"]

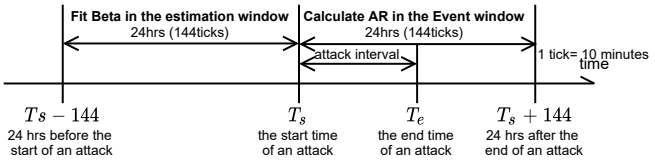


Fig. 3: Calculation of the abnormal return (AR) and the cumulative abnormal return (CAR).

Audit Reports: We collect and manually inspect 30 recent public audit reports from 6 security testing companies (Beosin, PeckShild, Slowmist, Consensys, Certik, Trial of Bits). We notice that the reports collected perform manual auditing and may not explicitly disclose what the auditors examined. For example, while each of the six companies checked the common vulnerability "inconsistent access control" in at least one audit report, only 19 of the 30 (63%) audit reports explicitly state it. For reproducibility and objectiveness, we can only be certain that an audit has addressed an incident type, if it: *(i)* explicitly warns about the risk of a potential incident, or *(ii)* explicitly states that the code passed the check of an incident type. This methodology, however, leads to an underestimation of the absolute number of incident types addressed in the audit reports[12]. Note that we are only attempting to quantify whether practitioners address certain incident types less frequently than the others, and therefore this unbiased underestimation should have no significant impact on our analysis.

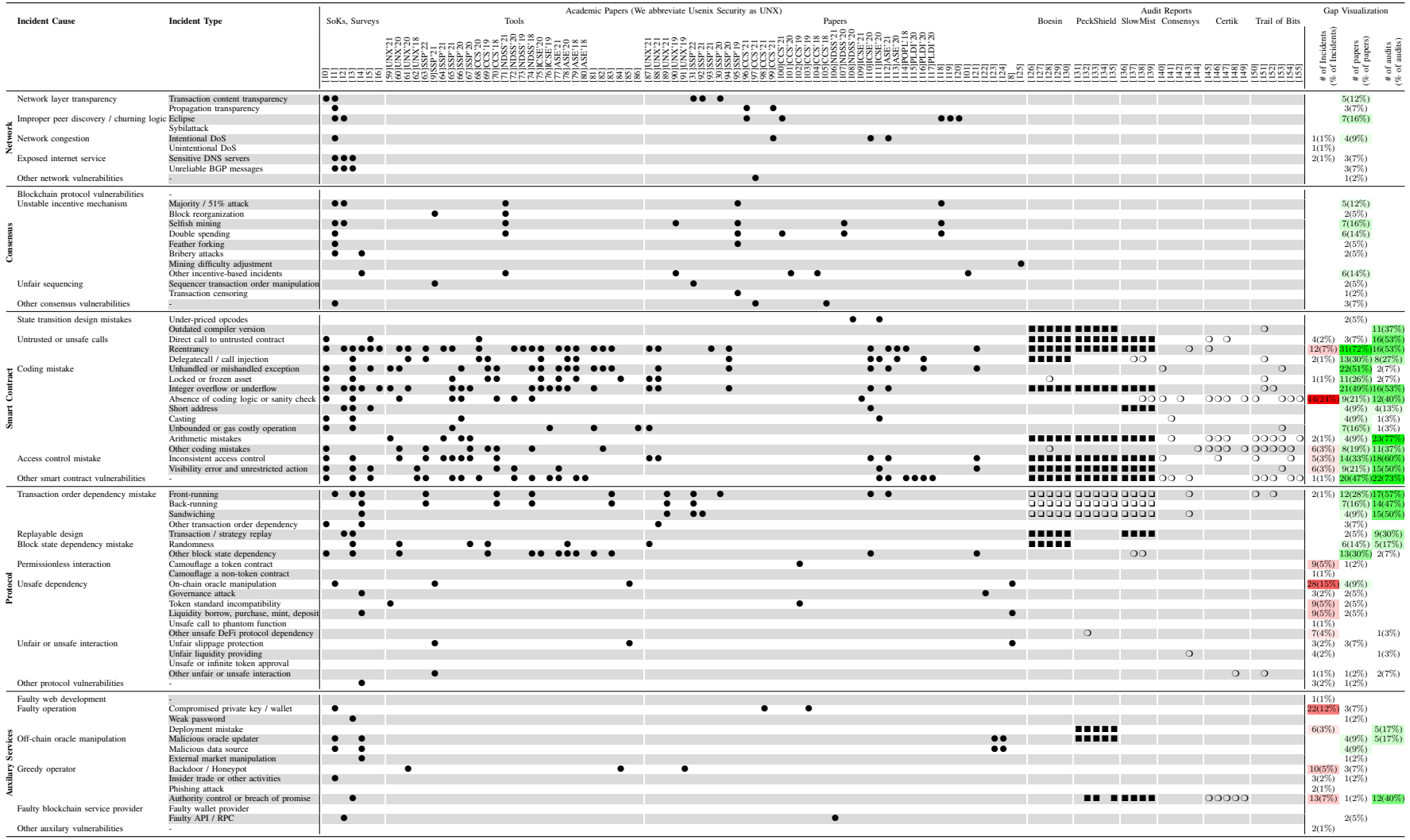
Incidents: Our dataset consists of 117 and 69 incidents on Ethereum and BSC respectively (in total 181 incidents) over a period of four years from Apr 30, 2018 to Apr 30, 2022. These incidents are gathered from the following three sources[13]: *(i)* Rekt News; *(ii)* Slowmist; and *(iii)* Cryptosec. We exclude non-DeFi incidents, such as blockchain-based gambling and gaming applications. The incidents of which we cannot identify the adversary are also excluded. We construct the following features for each of the incident:

- **Incident Type and Cause:** We manually label the type and cause of each incident (cf. Table III for incidents taxonomy, which is further discussed in Section VI). It should be noted that we may associate one incident with multiple types or causes across multiple system layers.
- **Adversaries:** When we can identify an incident's adversaries, we manually classify adversarial goal, knowledge, and capability based on our reference frame (cf. Section II).
- **Averaged Total Monetary Loss (in USD):** The most perceptible impact of harm is direct monetary loss. We collect the total monetary loss reported by the aforementioned data sources, where the victim can be either users, liquidity

[12]As an example, Trial of Bits does check for PRO layer incidents in other audit reports, such as sandwich in TOB-Computable-018 (https://github.com/trailofbits/publications/blob/master/reviews/computable.pdf), replay in TOB-HERMEZ-014 (https://github.com/trailofbits/publications/blob/master/reviews/hermez.pdf), etc., but are not included in our sampled dataset.

[13]Correspondingly: *(i)* https://rekt.news/; *(ii)* https://hacked.slowmist.io/en/; and *(iii)* https://cryptosec.info/defi-hacks/

TABLE III: DeFi incidents taxonomy. We label the incident types that each academic paper and auditing report address. We also group the incidents that occur in the wild. Despite that this table focuses on Ethereum and BSC, we anticipate the taxonomy remains generic and thus applicable to all DeFi enabled blockchains. ● - Incident type addressed; ■ - Incident type checked (likely with tools); ❑ - Incident cause checked (likely with tools); ❍ - Incident type checked (manually). Note that we can only be sure that an incident type has been addressed if an auditing report: (i) explicitly warns of the risk of a potential incident, or (ii) explicitly states that the code passed the check of an incident type. We visualize the gaps using a heat map, where a darker colour indicates a greater frequency of occurrences.



providers, speculators, or protocol operators. When applicable, we cross-validated the loss with on-chain transaction data, and then remove sources that report incorrect loss[14].

- **Cumulative Abnormal Return (CAR):** CAR reflects harm by measuring how token price responds to an incident. We expect rational investors' risk aversion to information shocks will diverge the token price in the equilibrium and lead to abnormal returns (ARs) [156], [157]. We choose the capital asset pricing model (CAPM) as the benchmark for normal returns. We derive CAR with the following three steps:
  1) Equation 2 fits β coefficient with the ordinary least square, where $R_{i,t}$, $R_{mkt,t}$, r_{f_t} denotes the token price, market price and risk-free rate[15] at tick $t \in [T_{s-144}, T_s)$ respectively, α_i is the constant, and $\epsilon_{i,t}$ is the error term.

$$R_{i,t} - r_{f_t} = \alpha_i + \beta_i \cdot (R_{mkt,t} - r_{f_t}) + \epsilon_{i,t} \quad (2)$$

  2) Equation 3 calculates the ARs for each tick over the event time frame of $[T_s, T_{s+144})$, where $\hat{\beta}_i$ is the fitted β coefficient, $\mathbb{E}[R_{i,t}]$ is the expected return (i.e., the normal return) of token i, and $t \in [T_s, T_{s+144}]$. If there is no information shock, ARs would approximate zero, since they are derived with the normal $\hat{\beta}_i$ which is fitted within the estimation window.

$$AR_{i,t} = R_{i,t} - \mathbb{E}[R_{i,t}] = R_{i,t} - (\alpha_i + \hat{\beta}_i(R_{mkt,t} - r_{f_t}) + r_{f_t}) \quad (3)$$

  3) Note that the extreme value of CAR represents the biggest anomaly of the price behavior and the change of the AR direction can be considered evidence of the (dis)appearance of an anomaly [159]–[162]. Hence, to capture the price change pattern within the appearance of anomaly, we report the adaptive CAR (i.e., minimal CAR for $t \in [T_s, T_{s+144}]$) in Equation 4.

$$CAR_i = \min_t [\sum\nolimits_{t' \leq t} AR_{i,t'}] \quad (4)$$

- **Total Value Locked (in USD):** TVL is calculated as the product of the total token balance held by a protocol's smart contracts and token price in USD [163]. Greater TVL indicates greater value of assets that can be potentially compromised under DeFi incidents. We attain the pre-attack TVL for 126 incidents using DeBank[16] and DeFiLlama [1].
- **Audit Status:** For each incident, we manually search auditing histories from the following four sources: *(i)* a protocol's website; *(ii)* a protocol's social media and blog post (e.g., Twitter and Medium); *(iii)* public git repositories; (iv) a search engine (i.e., Google). We then label each incident according to the following rules: *(Audited):* the victim smart contract is audited prior to the incident; *(Partially Audited):* audits are performed before the incident, but not for the specific victim smart contract or for an older version; and *(Not Audited):* no audit history is found prior to the incident.
- **Emergency Pause, Disclosure and Reimbursement:** We crawl the following three features in an attempt to measure a protocol's reactive defense: *(a)* Did the protocol disclose the incident within 20 days?[17] *(b)* Has the protocol reimbursed its users within 20 days? and *(c)* Did the protocol execute a circuit breaker [164] or emergency pause? We manually search for auditing histories from the following three sources: *(i)* public announcements on a protocol's website; *(ii)* a protocol's social media and blog post (e.g., Twitter and Medium); and *(iii)* the protocol's main discussion forum.

Limitations: Our methodology has the following limitations:

- **Soundness:** Because our data crawling process is heavily reliant on manual labor, human errors may occur. To mitigate this limitation, we cross-validate our data with external sources whenever possible. Additionally, we conduct internal data reviews through pull requests. Each incident is reviewed by at least two paper authors before the pull request is merged.
- **Completeness:** - Despite that Ethereum and BSC account for 77% of the total DeFi NVL (cf. Section I), incidents' features, such as adversarial behavior and deployed defense, on other DeFi enabled blockchains can differ. To ensure the paper's reproducibility, we only consider fully disclosed incidents that can be found through public sources. While incomplete, this DeFi incident dataset is the largest available collection that we are aware of.
- **Bias:** - Our incidents dataset is gathered from three publicly sources (e.g., Rekt News, Slowmist and Peckshield). These three sources are, to our knowledge, the most extensive DeFi incident databases accessible. Unfortunately, none of these three sources explicitly document their data collection process. As a result, we are unable to evaluate whether these sources contain bias, and our dataset may therefore inherit the sampling bias from these sources[18].

IV. Analysis

A. Incident Frequency

Figure 4 shows the monthly number of incidents in relation to the total monthly loss. We find that the majority of the DeFi incidents occur after late 2020, with the peak in August 2021, when nearly 600 million dollars are lost in a single month.

Despite the fact that BSC is a relatively new blockchain, it experienced 69 DeFi incidents. We discover that 29 of the BSC incidents are exploiting PRO layer design flaws. In particular, between the 19th of May and the 3rd of June 2021, we observe

[14]We rely on Uniswap, Sushiswap, Pancakeswap and Bakery swap as our price oracles when validating on-chain transaction

[15]The common practice is to use the 1- or 3-month US treasury bill yield as a proxy of r_{f_t} (cf. Figure 3). We assume $r_{f_t} = 0$ [158], since the high-frequency US treasury bill yield data (i.e., 10 minute per tick) is unavailable. We obtain the token price from Uniswap, Sushiswap, Pancakeswap and Bakeryswap's on chain smart contracts, and then use the average price of Bitcoin and Ethereum during the same timeframe as a proxy of market price.

[16]https://open.debank.com/, accessed on September 30, 2021

[17]We choose a custom time frame as an example.

[18]For example, our methodology only includes BEV incidents disclosed in the three abovementioned sources. For detailed BEV studies, we refer the interested reader to the rich corpus of previous works [30]–[33], [89]

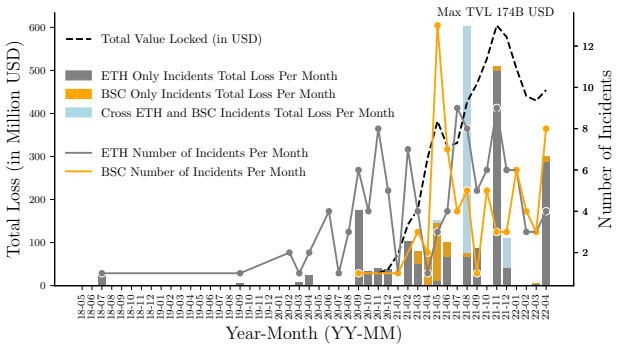


Fig. 4: Monthly number of DeFi incidents and total loss (in million USD) for Ethereum and BNB Smart Chain from Apr 30, 2018 to Apr 30, 2022, in comparison to the total value locked. According to our data, the frequency, and monthly loss increase as the TVL increases.

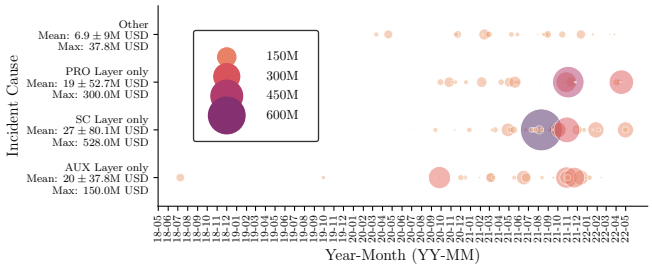


Fig. 5: Loss (in million USD) and frequency of DeFi incidents on Ethereum and BNB Smart Chain from Apr 30, 2018 to Apr 30, 2022 grouped by incident cause. Each circle represents a unique incident, and the size of the circle is proportional to the estimated monetary loss in USD.

recurring exploits on a group of forked protocols[19]. The time frame of 15 days suggests that attackers do not yet have automated tools to scan and reproduce similar incidents.

Figure 5 illustrates the incident frequency per group and the involved system layer. Overall, we find that the frequency of all incident types increase over time from 3.1 per month in 2020 to 8.5 per month in the first four months of 2022 on average (2.74×). We also observe that the most common incident cause are SC Layer (42%), PRO Layer (40%), and AUX Layer (30%) vulnerabilities.

B. DeFi Protocol Types

Table IV groups the incidents that we collect based on their protocol/application type. We find that yield farming protocols and cross-chain bridges incur 44% of the total monetary loss, although their total TVL is only 20.6 billion

[19]PancakeBunny suffered a performance fee minting attack on May 19, 2021, where the adversary manipulated the on-chain oracle and siphoned $\$45M$ in profit. Within two weeks, the copycats Autoshark, MerlinLab and PancakeHunny were exploited in a similar fashion: the adversary *(i)* exploited the vulnerability of `mintFor/mintForV2` function to manipulate LP token prices and *(ii)* used cross-chain bridge and TC to launder money.

	Yield	Bridge	Lending	DEX	Stablecoin	DAO	Payment	Derivatives	Insurance	Others
Is the monetary loss related to the type of the DeFi protocol?										
Loss (in M USD)	868	860	485	450	286	200	72	32	14	713
Pct. of Total Loss	22%	22%	13%	12%	7%	5%	2%	1%	0%	18%
Is the number of the security incidents related to the type of the DeFi protocol?										
Num. of Incidents	50	10	22	28	7	7	7	6	3	49
Pct. of Incidents	27%	5%	12%	15%	4%	4%	4%	3%	2%	27%
TVL (in B USD)	9.2	11.4	18.2	27.7	-	-	0.5	2.2	0.6	-
Is the vulnerability type related to the type of the DeFi protocol?										
SC layer related	48%	60%	50%	39%	43%	0%	0%	50%	33%	43%
AUX layer related	20%	30%	18%	29%	0%	43%	71%	50%	33%	47%
PRO layer related	52%	10%	59%	39%	86%	29%	43%	17%	33%	24%
NET layer related	0%	0%	5%	4%	0%	14%	0%	0%	0%	2%

TABLE IV: Loss (in million USD) and frequency of DeFi incidents grouped by application type, on Ethereum and BNB Smart Chain from Apr 30, 2018 to Apr 30, 2022. We crawl TVL for each category from DeFiLlama on Aug 6, 2022.

Latent Variable	Observed Variable	Description
Preventive Defense	$PD1$	Was the victim protocol audited before the incident?
	$PD2$	Does the victim protocol support emergency pause?
Asset	$A1$	Total value locked (TVL, in USD)
Reactive Defense	$RD1$	Duration between incident occurrence and emergency pause
	$RD2$	Was the incident disclosed?
Harm	$H1$	Cumulative abnormal return (CAR) (in %)
	$H2$	Total monetary loss (in USD)

TABLE V: Latent and observed variables we construct in structural equation modeling (SEM).

USD (30.2%). In contrast, DEX protocols have the biggest TVL (27.7 billion USD, 40.6%), but have only incurred a loss of 450 million (12%). In addition, we observe that the distribution of vulnerabilities varies per protocol type. For example, 86% and 59% of the incidents related to stablecoins and lending involve PRO layer vulnerabilities respectively, which is significantly higher than other protocol types.

C. Structural Equation Modeling

In this section, we apply *Structural Equation Modeling* (SEM) [165]–[175] to test and measure causal relationships between variables (cf. Figure 6 and Table V).

- **What is SEM:** SEM refers to a collection of techniques to examine "latent variables" that are assumed to exist but cannot be directly observed. In more detail, SEM is a multivariate analysis technique that supports a flexible

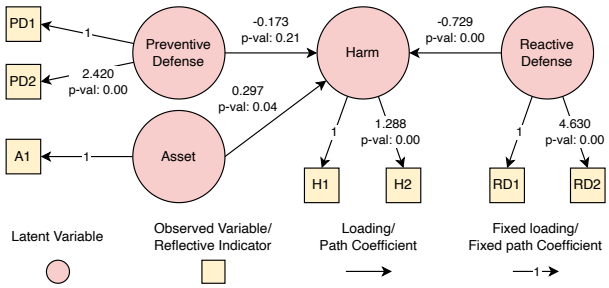


Fig. 6: Structural equation model (SEM) after fitting.

hybrid of confirmatory factor analysis (CFA) [176]–[178] and latent structural regression [175], [179]. An SEM model encompasses two sub-models [180] (cf. Equation 5): *(i)* a *measurement model* that conducts CFA to test the hypothesized relationships between a given latent variable and its corresponding observed variables; and *(ii)* a *structural model* that performs latent structural regression to infer the causal relationships between different latent variables.

$$\begin{cases} \eta = B\eta + \varepsilon & \text{(structural model)} \\ y = \Lambda\eta + \delta & \text{(measurement model)} \qquad \textit{where}: \end{cases} \tag{5}$$
η and y are vectors of latent and observed variables;
ε and δ are independent error terms.

- **Why SEM:** The literature [181] utilized SEM to study latent variables in cyber risks. In this work, we apply similar techniques to measure the causal relationships in DeFi incidents. To this end, we do not consider approaches that are unable to support causal inference in the presence of latent variables, such as linear mixed models [182] and dimensional reduction techniques [183]. Previous literature suggests the causal Bayesian network being the best alternative to SEM. However, it requires at least 1000 samples to get a satisfactory performance. With limited samples of DeFi incidents, we consider SEM a more suitable approach.
- **Specification:** Our model consists of four latent variables, including one endogenous/dependent variables (i.e., *harm*), and three exogenous/independent variables (i.e., *asset*, *preventive defense* and *reactive defense*). We measure one or two observed variables for each latent variable (cf. Table V). To construct the causal graph, we employ a variation of the hypothesis by Wood and Böhme [181]: *preventive defense*, *reactive defense* and *asset* jointly affect *harm*.
- **Estimation:** We utilize a logarithmic price scale to transform monetary values (e.g., TVL and monetary loss). We then further apply min-max normalization to convert continuous variables to values in range $[0, 1]$. Categorical values are mapped into ordinal values[20][21][22]. We fit our SEM using an open-sourced library, *semopy* [180] (cf. Figure 6).
- **Fitness:** Our model is examined using a collection of indices, including *(i)* the adjusted Chi-square ($\frac{\chi^2}{DoF}$) [184]; *(ii)* goodness of fit index (GFI) [167]; *(iii)* comparative fit index (CFI) [185]; and *(iv)* normed fit index (NFI) [186]. The majority of indices conform to their commonly accepted value in the literature except adjusted Chi-square[23].
- **Analysis:** Our findings suggest that the latent variable "harm" increases with "asset exposure", which conforms with previous security research. We also find that harm decreases if the latent variable "reactive defense" increases. To our surprise, the p-value for preventive defense is high (0.21), meaning that our model does not find strong evidence to suggest preventive defense reduces harm.
- **Limitations:** Our primary limitation is the relatively small sample size. In the event that the number of DeFi incidents increases in the future, our model should be re-evaluated and cross-validated using additional causal experiments.

[20]PD1: $\{\text{Not Audited} \to 0, \text{Partially Audited} \to 0.5, \text{Audited} \to 1\}$

[21]PD2: $\{\text{No Emergency Pause} \to 0, \text{Supports Emergency Pause} \to 1\}$

[22]RD2: $\{\text{Not Disclosed} \to 0, \text{Disclosed} \to 1\}$

[23]Previous works suggest that the fit should be ≤ 5 for adjusted Chi-square [187], and ≥ 0.9 for GFI [188], CFI [189] and NFI [186]. Our model yields an adjusted Chi-square of 4.44, GFI of 0.96, CFI of 0.97, NFI of 0.96.

Duration after the incident starts	$\leq 1h$	$\leq 6h$	$\leq 12h$	$\leq 24h$	$\leq 48h$
Number of protocols	1	24	11	7	8
Percentage (out of 87 protocols)	2.%	47%	22%	14%	16%

TABLE VI: We quantify the speed at which DeFi protocols execute their emergency pause. Out of the 87 DeFi protocols that allow an emergency pause, the fastest has initiated a pause within the first hour of an incident.

D. Emergency Pause

DeFi protocols may support an emergency pause, which is analogous to circuit breakers [164] in conventional centralized exchanges. This section examines the speed at which DeFi protocols initiate an emergency pause (cf. Table VI). According to our data, 87 of the 183 victims support the emergency pause mechanism (47.5%). However, only 51 of the 87 protocols (58.6%) pause their protocol within 48 hours, and only one protocol pauses within the first hour of the incident. Our statistics suggest that DeFi protocols may not yet have just-in-time intrusion detection mechanisms to identify abnormal protocol states or malicious transactions, which limits the effectiveness of an emergency pause mechanism.

E. Effectiveness of Security Audits

Section IV-C studies the influence of security audits on harm, by performing causal inference analysis (e.g., SEM) on past incidents only. In the following section, we will attempt to estimate the effectiveness of security audits.

- **Additional Data Crawling** To quantify the effectiveness of security audits, we perform the following steps: *(i)* We crawl all DeFi protocols using DeFiLama's public API [1]. Out of the 1080 protocols listed on DeFiLama, 776 are relevant to Ethereum and BNB Smart Chain. *(ii)* We map the DeFiLama dataset with our incident dataset and find that 56 of the 776 protocols have been exploited before Apr 30, 2022. *(iii)* We construct a new audit dataset by taking snapshots and merging two public databases on June 20, 2022 [190].
- **Result** According to our data, 4.09% of the 56 audited protocols have been attacked at least once, whereas 15.49% of the non-audited protocols have been attacked. Hence, our data indicates that a security audit can decrease the average probability of an exploit by a factor of four. Due to the relatively small sample size of only 56 matched incidents, our result can only be considered as a rough approximation.

V. Incident Defense

A. Rescue and Incident Time Frame

In the following, we investigate the rescue and the incident time frame (cf. Figure 7). The rescue time frame is the time

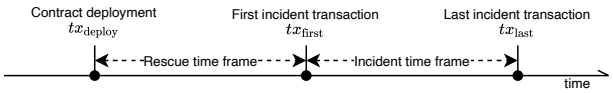


Fig. 7: An adversary $\mathbb{A}$ can deploy a smart contract with transaction tx_{deploy} and then initiate an incident by calling the contract with tx_{first}. Alternatively, the adversary may directly initiate the incident with tx_{first} in one of two ways: *(i)* without using a smart contract; or *(ii)* by batching the contract deployment and the initiation in a single transaction.

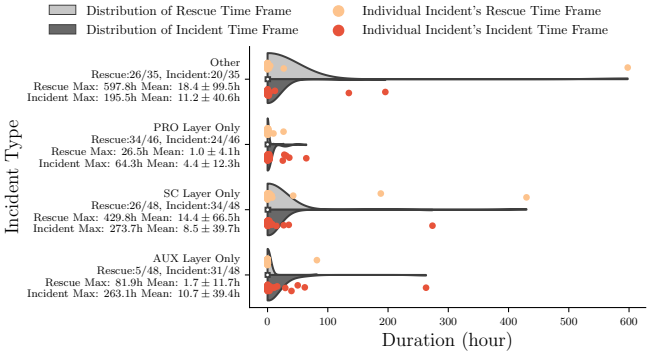


Fig. 8: The incident and rescue time frame per incident type. For example, we observe that 34 of the 46 PRO layer only incidents (74%) deploy smart contract(s) prior to the incident. The average rescue time frame for PRO layer is 1 ± 4.1 hours, with the longest rescue time frame being 26.5 hours.

between the adversarial contract deployment (tx_{deploy}) and the execution time of the first malicious state transition (tx_{first}). While the adversarial smart contract bytecode is already publicly available in the rescue time frame, the incident has not yet occurred. As such, defensive tools can theoretically reverse engineer the contract bytecode and determine its strategy using methods such as symbolic analysis, static analysis, and fuzzing, potentially mitigating or preventing harm. To our knowledge, no such just-in-time tool exists yet, which may explain why adversaries do not batch tx_{deploy} and tx_{first} into a single transaction yet (cf. Figure 8). The incident time frame, is the time that elapses between the execution of the first and last harmful state transition transactions. An $\mathbb{A}$ may prefer to keep the incident period as short as possible to maximize the attack's success rate, which however may not always be possible due to gas constraints, protocol design, etc.

Figure 8 lays out the durations of the attack and rescue time frames. We discover that 103 (56%) attacks are not executed atomically, granting a rescue time frame for defenders. PRO layer incidents have the shortest average rescue time frame duration of 1h $\pm$ 4.1. The "Formation.Fi" incident has the longest rescue time frame, lasting approximately 25 days.

B. Bytecode Similarity Analysis

In the smart contracts ecosystem, code cloning has been utilized to measure the code similarity of deployed contracts [191], identify plagiarized dApps [192], and vulnerability detection [193]. In this work, we employ code cloning to quantify bytecode similarity between all exploited DeFi protocols and adversarial contracts studied in this work. Note that we choose to perform our study at the deployed bytecode level as opposed to the source code level, because smart contract developers can close-source the contract code.

Category	Similarity Threshold	Contracts			Unique Incidents		
		Total	Clusters	Detectable	Total	Clusters	Detectable
Vulnerable	100%	38	7	31	5	2	3
	80%	85	26	59	50	23	27
Adversarial	100%	29	6	23	0	0	0
	80%	73	23	50	31	13	18

TABLE VII: We perform bytecode similarity analysis on our incident dataset, which includes in total 173 vulnerable and 155 adversarial contracts. We identify 7 clusters of "exact match" vulnerable contracts (in total 38 vulnerable contracts), where contracts within the same cluster have a pairwise similarity score of 100%. Therefore, we infer that $38 - 7 = 31$ vulnerable contracts could be detected prior to the incident by comparing with previous known vulnerable contracts. Similarly, we infer that 23 adversarial contracts could be detected by comparing with previous known attacks.

Methodology: Our code cloning detection method is inspired by the works of Kiffer et al. [191] and He et al. [192]. Specifically, to group similar smart contracts, we first identify and remove the Swarm code part from the bytecodes as it is not served for execution purposes. Then, we disassemble the bytecodes and remove the PUSH instructions' arguments. Next, similar to [191], we compute hypervectors of n-grams ($n = 5$) of Ethereum opcodes for each contract. In order to compare two contracts, we compute the Jaccard similarity of their respective hypervectors. Finally, to cluster smart contracts into groups, we require a similarity score greater than 80% that the previous study suggests [191] [192].

Results: Table VII presents the results of the similarity analysis. We apply the above-mentioned methodology to cluster 173 vulnerable contracts and 155 adversarial contracts in our dataset. Using a similarity score threshold of 80%, we group vulnerable and adversarial smart contracts into 26 and 23 clusters, respectively. In addition, we note that in some clusters, all contracts are associated with a single incidence. To address more intriguing questions, such as how many comparable adversarial contracts attack different protocols (or different vulnerabilities in the same protocol), we restrict each cluster to a single contract per incident (c.f. Table VII).

We manually investigate the remaining clusters to acquire additional insights. For the vulnerable contracts, the clusters contain contracts that are part of DeFi protocols with similar functionalities (e.g., bridges and yield farming applications). Additionally, the exploitation of identical contracts is nearly equal (e.g., exploiting the same issue with equivalent transactions). In contrast, for similar vulnerable contracts, the exploits are not the same, but the incident cause is typically the same.

For example, we identify two adversaries that exploit an issue on the same function in two smart contracts used as bridges, which fork the same smart contract. Specifically, although the implementation of the function is slightly different in the two contracts, both protocols introduce a vulnerability in the exact function while forking and modifying the same contract.

The most notable outcome of our similarity analysis is the identification of clusters of adversarial smart contracts that target distinct DeFi protocols with similar vulnerabilities (e.g., oracle manipulation). An analysis of historical blockchain data could reveal more adversarial smart contracts. Furthermore, we could potentially identify adversarial smart contracts in real-time, given that the time frame is long enough, by applying a more sophisticated similarity detection technique that could work on a more fine-grained level (e.g., function-level). Combining this with other program analysis techniques could potentially mitigate or prevent exploits (c.f. Section V-A).

Limitations: Our methodology cannot cluster similar contracts that employ different compilers and optimization choices. In addition, if an adversary choose to obfuscate the bytecode by, for example, injecting unused function code into the contract, our method becomes less effective. We therefore highlight the application of more sophisticate strategies as an interesting avenue for future work [194].

C. Front-Running as a Service (FaaS) Usage

FaaS are servers to which a trader's transactions can be privately forwarded to miners that peer with the FaaS. We find that at least 18 incidents are executed through FaaS using Flashbots API on Ethereum. The first attack going through Flashbots happened on July 12, 2021.

- **Arbitrageurs Accelerate Attacks:** We manually examined each Flashbots bundle and discover that 6 of the 18 incidents appear to be accelerated by, e.g., arbitrage traders. We find that this is due to adversaries conducting incidents with sub-optimal strategy, resulting in extractable BEV opportunities. Trading bots will then compete for these BEV opportunities by back-running incident transactions with FaaS.
- **Private Adversarial Transactions:** Adversaries can execute an incident using FaaS services, without broadcasting any transactions on the public blockchain P2P network. As a result, only entities with sequencer knowledge (K_2) are able to defend against these adversaries (e.g., perform bytecode similarity analysis) prior to transaction confirmation.

D. Money Tracing

Adversaries require a source of funds to issue transactions to execute incidents. $\mathbb{A}$ may attempt to break the linkability of their source of funds to evade potential legal ramifications. This section proposes a money tracing methodology to analyze the pre-incident flow of funds (cf. Figure 9). An incident's source of funds is usually originating from a native coin transfer, e.g., from an address X to an address Y, i.e., $X \rightarrow Y$. We apply Algorithm 1 in the appendix to identify the funding transaction $X \rightarrow Y$ for address Y. We abbreviate our notation with $X \xrightarrow{h} Y$, representing h hops transfer (i.e., $X \rightarrow I_1 \rightarrow \ldots \rightarrow I_{h-1} \rightarrow Y$). To our knowledge, the current literature has not proposed any methodology to trace an incident's source of funds on an account-based ledger.

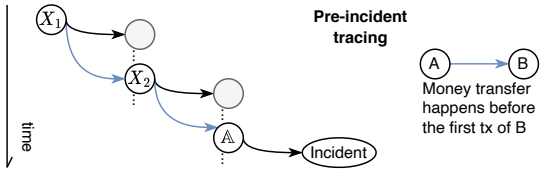


Fig. 9: Overview of the money tracing methodology. We start with the adversarial address ($\mathbb{A}$), then iteratively determine the addresses that provide the initial source of funds (i.e., X_2 and X_1, analogous to depth-first search).

Pattern	Total	$h=1$	$h=2$	$h \geq 3$
Pre-incident (76 incidents in total, excluding U_2-non-monetary adversaries)				
Centralized Exchange $\xrightarrow{h} \mathbb{A}$	128(49.0%)	40(15.3%)	23(8.8%)	65(24.9%)
Tornado.Cash $\xrightarrow{h} \mathbb{A}$	94(36.0%)	67(25.7%)	19(7.3%)	8(3.1%)
Typhoon.Network $\xrightarrow{h} \mathbb{A}$	9(3.4%)	6(2.3%)	2(0.8%)	1(0.4%)
Mining Pool $\xrightarrow{h} \mathbb{A}$	7(2.7%)	-	1(0.4%)	6(2.3%)
Cross-chain Bridge $\xrightarrow{h} \mathbb{A}$	5(1.9%)	3(1.1%)	2(0.8%)	0(0.0%)
Unknown	18(6.9%)			

TABLE VIII: Source of funds identified for all 261 adversaries. h represents the number of hops (i.e. transactions) from the source of funds, e.g., In total, $73(28.0\%)$ adversaries $(92(50.8\%)$ incidents) source the funds directly from a mixer.

- **Centralized Exchange:** We observe that $12(7.3\%)$ (on Ethereum) and $21(8.0\%)$ (on BSC) adversaries directly withdraw from exchange wallets ($h = 1$). The identities of these attackers can be revealed if the corresponding exchanges comply with Know Your Customer (KYC) requirements. For indirect exchange withdrawals ($h > 1$), we can only determine that $\mathbb{A}$ is linked to the withdrawer, but not whether the withdrawer is the attacker.
- **Mixer:** $55(21\%)$ (on ETH) and $12(4.6\%)$ (on BSC) adversaries receive their initial funds directly from a mixer ($h = 1$). Note that we classify a mixer as the source of funds only if a so-called relayer executes the withdrawal transaction (i.e., a third-party paying the transaction fees in the native blockchain coin); otherwise, we assume that the withdrawal fee payer is linked to the withdrawer and continue tracing the money flow. Relayers help to break address linkability, by paying the transaction fees (gas fee) of mixer withdrawal transactions in exchange for a commission on the withdrawal value.
- **Cross-chain Bridge:** Four attackers directly withdraw their source of funds from a blockchain bridge ($h = 1$).

Linked Incidents We discover that the adversarial address in 13 incidents can be linked to another incident's adversary within three hops (cf. Table X in the appendix).

Limitations We utilize Ether- and Bscscan[24] to identify the addresses of centralized exchanges and cross-chain bridges.

[24]https://etherscan.io/labelcloud and https://bscscan.com/labelcloud.

Layers	Surveys/SoKs		Tools		Papers		Audit reports		Incidents
	✦	✧	✦	✧	✦	✧	✦	✧	✦
Total	7		29		42		30		181
NET	4(57%)	19%	-	-	12(29%)	4%	-	-	4(2%)
CON	3(43%)	13%	2(7%)	2%	11(26%)	5%	-	-	0(0%)
SC	6(86%)	31%	26(90%)	20%	15(36%)	4%	29(97%)	35%	77(42%)
PRO	5(71%)	13%	15(52%)	6%	12(29%)	3%	19(63%)	14%	73(40%)
AUX	4(57%)	10%	2(7%)	1%	6(14%)	2%	14(47%)	5%	56(30%)

TABLE IX: Distributions of works under investigation according to the DeFi reference frame (cf. Section II-A). ✦ - the number and percentage of research items related to a system layer; ✧ - the average ratio of incident types each research item covers. For example, 15 of the 29 tools (52%) relate to PRO layer incidents, but each tool on average only covers 6% of the common PRO layer incident causes we identify.

Our dataset therefore inherits potential data completeness issues from Ether- and Bscscan.

VI. Discussion

DeFi Incidents — Another Cat and Mouse Game: Analog to traditional information security, DeFi incidents can be perceived as a cat-and-mouse game, in which defenders attempt to minimize the security risk surface while attackers breach defenses. In the following, we extract insights on the current state of this contest, highlight key findings, discuss their implications and make recommendations for future research.

1) **Insight - Understudied NET and CON incidents:** We observe that NET and CON-related incidents are studied in 29% and 26% of academic papers (excluding tools, SoKs and surveys). However, only two tools (SquirRL [71], DeFiPoser [9]) as well as 2% and 0% of the in-the-wild-incidents relate to the NET and CON layers, respectively. While related works have surprisingly identified evidence of miner misbehavior in block header timestamps for financial gain [125], we note that: *(a)* it is not trivial to identify NET and CON incidents with absolute certainty (e.g., transaction censoring, selfish mining attack and block reorganization attack); and *(b)* to our knowledge, no publicly available tool can comprehensively detect potential NET and CON incidents in DeFi. As such, we suspect that more incidents have yet to be discovered. Furthermore, we notice that none of the industrial DeFi audit reports explicitly address potential NET and CON incidents, while some companies have previously performed NET and CON auditing for layer 1 and 2 blockchains[25].
2) **Challenge - Low coverage for PRO incidents:** SC and PRO layer incidents are the most common incident type (42% and 40%, respectively). Security tools, however, only cover 52% of the PRO layer incident types on average, which is less than SC layer (90%). As such, our dataset indicates that most defense tools still focus on SC vulnerabilities. The literature, however, suggests that the development of effective and generic PRO incident defense tools remains an open security challenge [9]. This is mainly due to DeFi's composability feature, which leads to action path explosion in detecting PRO layer vulnerabilities.
3) **Insight - Repeated on-chain oracle manipulation:** We discover 28 (15%) on-chain oracle manipulation incidents on Ethereum and BSC, which is the most common PRO layer incident type. On-chain oracle manipulation is one type of composability attack, which implies the adversary has C^3_{PRO} capability. Repeated on-chain oracle manipulation indicates the need for tools to automatically identify such attack. To our knowledge, only DeFiRanger [85] and DeFiPoser [9] can detect oracle manipulation vulnerabilities. DeFiRanger can only identify observed attack transactions, whereas DeFiPoser can identify new vulnerabilities in real-time, but necessitates manual and costly modeling of the captured DeFi protocols.
4) **Insight - Permissionless interactions are dangerous:** The permissionless interactions between various DeFi protocols can further broaden the attack surface. According to our dataset, in 19 (10.5%) incidents, adversaries utilize or deploy a contract (C^5_{PRO}), which complies with the accepted ABI interface, but contains incompatible implementation logic that causes harm[26]. The underlying cause of these incidents is that the victims only constrain the contract function interface, not how the contract is implemented. We are, however, unaware of any viable way to efficiently verify code implementation on-chain due to the limitations of the current SC layer design. An alternative solution for constraining the contract with which a protocol or its user interacts is to implement a whitelist, where a DeFi protocol can only interact with other protocols in the whitelist.
5) **Insight - The identities of the attackers may still be revealed:** Although mixers are available on both Ethereum and BSC, our empirical result shows that only 38% of attackers obtain their source of funds from mixers (i.e., C^1_{PRO}). The majority of attackers interact with AUX services, such as centralized exchanges, and mining pools, which may provide stored personally identifiable information upon regulatory requests. Note that we naively assume mixers leaking the least side-channel information compared to other methodologies. Wang *et al.* [25] develop heuristics to reduce the anonymity set of Tornado.cash and Typhoon mixers on Ethereum and BSC. Quesnelle *et al.* [195] and Kappos *et al.* [196] investigate Zcash and show that the anonymity set size can be significantly reduced using simple heuristics to link transactions. Tran *et al.* [197] and Pakki *et al.* [198] show that existing mixer services are vulnerable to various threats such as permutation leak.
6) **Insight - Adversaries can be front-run during the rescue time frame:** Su *et al.* [87] discover that blockchain adversaries test their code by sending several transactions to the victim protocol before the actual attack. We initially

[25]TrailOfBits for example audits many L1 and L2 blockchain projects, such as Arbitrum, THORChain, ZCash, etc. (https://github.com/trailofbits/publications#blockchain-protocols-and-software)

[26]i.e. the following incident types: *(i)* token standard incompatibility; *(ii)* camouflage a token contract or *(iii)* camouflage a non-token contract

questioned this finding because anyone can inspect the adversarial smart contract bytecode and transactions on the P2P layer, and therefore can front-run the adversaries to rescue the victim protocol. The optimal strategy for $\mathbb{A}$ is to emulate the state transitions off-chain, then deploy and exploit in one single transaction (i.e., the capability C^4_{PRO}). Surprisingly, our empirical results support Su *et al.* [87] (cf. Section V-A). We encourage the development of tools to front-run adversaries during this rescue time frame.

7) **Challenge - Absence of intrusion detection tools:** Only one incident in our dataset has triggered the emergency pause within the first hour of the incident. This indicates the absence of intrusion detection tools to automatically trigger emergency pauses. We anticipate that just-in-time detection of abnormal protocol states or malicious transactions will receive increased attention in future studies.
8) **Insight - Adversarial and vulnerable contracts are detectable:** We show that SoTA similarity analysis can detect vulnerable and adversarial contracts. For instance, we identify 31/23 exactly matching vulnerable/adversarial contracts (i.e., bytecode similarity score of 100%) when compared to previously known incidents.

VII. Related Works

Cyber Risks: Sheyner *et al.* [199] outline an algorithm that can automatically generate attack graphs and analyze network security. Wang *et al.* [200] present a framework for measuring various aspects of network security metrics based on attack graphs. Khan *et al.* [201] propose a generalized mathematical model for cybersecurity that quantifies a set of parameters including risk, vulnerability, threat, attack, consequence, and reliability. Amin *et al.* [202] adopt the structural Bayesian Network to capture the relationship between financial loss, cyber risk and resilience, as well as developed a scorecard based approach to qualitatively assess the level of cyber risk. We refer interested readers to an SoK that thoroughly categorizes previous cyber risk studies [181]. While the research literature of cyber risks span over 30 years, DeFi is a relatively recent area with fewer works (cf. Table III).

DeFi Security: This paper proposes a five-layer system model as well as a comprehensive taxonomy of threat models that are used to measure and compare DeFi incidents. In the following, we present an overview of the most recent DeFi related survey and SoK papers, while highlighting the differences to contrast our work. Praitheeshan *et al.* [10] identify 19 software security issues and 16 Ethereum smart contract vulnerabilities, with 14 of them on smart contract layer. Homoliak *et al.* [11] present a stacked security model with four layers and systemized the vulnerabilities, threats, and countermeasures for each layer. Unfortunately, this research is not able to cover any smart contract layer vulnerabilities. Saad *et al.* [12] categorize 22 attack vectors in terms of its vulnerability origins (i.e., blockchain structure, P2P system and blockchain applications) and analyze the entities (e.g., miners, mining pools, users, exchanges, etc.) involved in each types of attacks. However, their examinations on protocol layer vulnerabilities and third-party vulnerabilities are conspicuously inadequate. Chen *et al.* [13] provide a comprehensive systematization of vulnerabilities, attacks, and defenses on four blockchain layers with detailed discussion on the relationships between them. Despite being able to cover in total of 40 vulnerabilities, this study does not state any vulnerabilities that are related to DeFi composability. Werner *et al.* [14] present a systematization of DeFi protocols and dissected DeFi related vulnerabilities with respect to technical and economic security. Nonetheless, this study lacks in-depth analysis of consensus and network layer vulnerabilities and does not provide generic measures to quantify the harm of DeFi incidents. Atzei *et al.* [15] investigate the security vulnerability on Ethereum and provided a taxonomy of the common programming pitfalls. Nevertheless, the vulnerability coverage of this work is unsatisfactory as it exclusively focuses on smart contract layer. Samree *et al.* [16] identify 8 application level security vulnerabilities on the smart contract layer, analyze past attack incidents and categorize detection tools. However, this study also focuses on addressing smart contract vulnerabilities. Wan *et al.* [109] conduct 13 interviews and 156 surveys to investigate the practitioners' perceptions and practices on smart contract security. They, however, do not reveal how much effort was allocated into the security of each system layer. For studies and tools related to specific incidents, we refer interested readers to Table III.

Code Cloning: Code clone detection has been extensively explored in the literature for both source code [203] and binary programs [204]. Token based [205], tree based [206], graph based [207], text based [208], and deep learning based [209] techniques are the most prevalent techniques explored for code cloning. Applications of code cloning include bug detection, malware detection, patch analysis, plagiarism detection, and code similarity [203], [204], [210], [211]. Smart contract code cloning has been utilized primarily for computing duplication [191]–[193], [212]–[215] and vulnerability search [193], [212]. In this work, we apply a code cloning detection for comparing vulnerable and adversarial smart contracts.

Blockchain money tracing and account linking: Androulaki *et al.* [216] evaluate the privacy provisions in Bitcoin and show that nearly 40% of user profiles can be recovered. Meiklejohn *et al.* [217] apply heuristic clustering to group Bitcoin wallets. Yousaf *et al.* [218] develop heuristics allowing to trace transactions across blockchains. Victor [219] proposes heuristics to cluster Ethereum addresses by analyzing the phenomena surrounding deposit addresses, multiple participation in airdrops and token transfer authorization on Ethereum. The most relevant paper to this study is Su *et al.* [87], which analyze adversarial footprints and operational intents on Ethereum. In this work, we examine adversarial money flow before the attack to determine the source of funds.

VIII. Conclusion

This paper constructs a DeFi reference frame that categorizes 77 academic papers, 30 audit reports, and 181 incidents, which reveals the differences in how academia and the practitioners' community defend and inspect incidents. We

investigate potential defense mechanisms, such as comparing victim/adversarial smart contract bytecodes, quantifying attack time frames, and tracing each attacker's source of funds. Our results suggest that DeFi security is still in its nascent stage, with many potential defense mechanisms requiring further research and implementation.

IX. Acknowledgement

This work is partially supported by Chainlink labs, SwissBorg SA, Nimiq Foundation, Lucerne University of Applied Sciences and Arts Switzerland, and the Federal Ministry of Education and Research of Germany [27].

References

[1] DeFillama. (2022) DeFillama Dashboard. https://defillama.com/.
[2] H. Adams, N. Zinsmeister, M. Salem, R. Keefer, and D. Robinson, "Uniswap v3 core," 2021.
[3] M. Egorov, "Stableswap-efficient mechanism for stablecoin liquidity," *Retrieved Feb*, vol. 24, p. 2021, 2019.
[4] J. A. Berg, R. Fritsch, L. Heimbach, and R. Wattenhofer, "An Empirical Study of Market Inefficiencies in Uniswap and SushiSwap," in *2nd Workshop on Decentralized Finance (DeFi), Grenada*, May 2022.
[5] "Aave," https://github.com/aave/aave-protocol, 2020.
[6] "Compound finance," https://compound.finance/, 2019.
[7] "Synthetix," https://www.synthetix.io/, 2020.
[8] K. Qin, L. Zhou, B. Livshits, and A. Gervais, "Attacking the defi ecosystem with flash loans for fun and profit," in *International Conference on Financial Cryptography and Data Security*. Springer, 2021, pp. 3–32.
[9] L. Zhou, K. Qin, A. Cully, B. Livshits, and A. Gervais, "On the just-in-time discovery of profit-generating transactions in defi protocols," *arXiv preprint arXiv:2103.02228*, 2021.
[10] P. Praitheeshan, L. Pan, J. Yu, J. Liu, and R. Doss, "Security analysis methods on ethereum smart contract vulnerabilities: a survey," *arXiv preprint arXiv:1908.08605*, 2019.
[11] I. Homoliak, S. Venugopalan, D. Reijsbergen, Q. Hum, R. Schumi, and P. Szalachowski, "The security reference architecture for blockchains: Toward a standardized model for studying vulnerabilities, threats, and defenses," *IEEE Communications Surveys & Tutorials*, vol. 23, 2020.
[12] M. Saad, J. Spaulding, L. Njilla, C. Kamhoua, S. Shetty, D. Nyang, and A. Mohaisen, "Exploring the attack surface of blockchain: A systematic overview," *arXiv preprint arXiv:1904.03487*, 2019.
[13] H. Chen, M. Pendleton, L. Njilla, and S. Xu, "A survey on ethereum systems security: Vulnerabilities, attacks, and defenses," *ACM Computing Surveys (CSUR)*, vol. 53, no. 3, pp. 1–43, 2020.
[14] S. M. Werner, D. Perez, L. Gudgeon, A. Klages-Mundt, D. Harz, and W. J. Knottenbelt, "Sok: Decentralized finance (defi)," *arXiv preprint arXiv:2101.08778*, 2021.
[15] N. Atzei, M. Bartoletti, and T. Cimoli, "A survey of attacks on ethereum smart contracts," in *International conference on principles of security and trust*. Springer, 2017, pp. 164–186.
[16] N. F. Samreen and M. H. Alalfi, "A survey of security vulnerabilities in ethereum smart contracts," *arXiv preprint arXiv:2105.06974*, 2021.
[17] S. Nakamoto and A. Bitcoin, "A peer-to-peer electronic cash system," *Bitcoin.–URL: https://bitcoin. org/bitcoin. pdf*, vol. 4, 2008.
[18] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, 2014.
[19] T. Mackinga, T. Nadahalli, and R. Wattenhofer, "TWAP Oracle Attacks: Easier Done than Said?" in *4th IEEE International Conference on Blockchain and Cryptocurrency (ICBC), Virtual Conference*, May 2022.
[20] S. N. Steve Ellis, Ari Juels, "Chainlink: A decentralized oracle network," 2017.
[21] "Tornado.cash," https://tornado.cash/.
[22] "Zcash," https://z.cash.

[27]The programme of "Souverän. Digital. Vernetzt.". Joint project 6G-life, project identification number: 16KISK002

[23] A. Hinteregger and B. Haslhofer, "An Empirical Analysis of Monero Cross-Chain Traceability," *CoRR*, vol. abs/1812.02808, 2018. [Online]. Available: http://arxiv.org/abs/1812.02808
[24] S.-F. Sun, M. H. Au, J. K. Liu, and T. H. Yuen, "Ringct 2.0: A compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency monero," in *European Symposium on Research in Computer Security*. Springer, 2017, pp. 456–474.
[25] Z. Wang, S. Chaliasos, K. Qin, L. Zhou, L. Gao, P. Berrang, B. Livshits, and A. Gervais, "On how zero-knowledge proof blockchain mixers improve, and worsen user privacy," *arXiv preprint arXiv:2201.09035*, 2022.
[26] D. V. Le and A. Gervais, "Amr: Autonomous coin mixer with privacy preserving reward distribution," *ACM Advances in Financial Technologies, AFT*, 2021.
[27] K. Qin, L. Zhou, Y. Afonin, L. Lazzaretti, and A. Gervais, "Cefi vs. defi–comparing centralized to decentralized finance," *arXiv preprint arXiv:2106.08157*, 2021.
[28] R. Braden, "Rfc1122: Requirements for internet hosts-communication layers," 1989.
[29] S. K. Kim, Z. Ma, S. Murali, J. Mason, A. Miller, and M. Bailey, "Measuring Ethereum network peers," in *Proceedings of the Internet Measurement Conference 2018*. ACM, 2018, pp. 91–104.
[30] P. Daian, S. Goldfeder, T. Kell, Y. Li, X. Zhao, I. Bentov, L. Breidenbach, and A. Juels, "Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020.
[31] K. Qin, L. Zhou, and A. Gervais, "Quantifying blockchain extractable value: How dark is the forest?" in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022.
[32] Y. Wang, Y. Chen, H. Wu, L. Zhou, S. Deng, and R. Wattenhofer, "Cyclic Arbitrage in Decentralized Exchanges," in *The Web Conference 2022 (WWW), Lyon, France*, April 2022.
[33] Y. Wang, P. Züst, Y. Yao, Z. Lu, and R. Wattenhofer, "Impact and User Perception of Sandwich Attacks in the DeFi Ecosystem," in *ACM CHI Conference on Human Factors in Computing Systems (CHI), New Orleans, LA, USA*, May 2022.
[34] L. Heimbach and R. Wattenhofer, "Eliminating Sandwich Attacks with the Help of Game Theory," in *ACM Asia Conference on Computer and Communications Security (ASIA CCS), Nagasaki, Japan*, June 2022.
[35] Q. Wang, J. Yu, S. Chen, and Y. Xiang, "Sok: Diving into dag-based blockchain systems," *arXiv preprint arXiv:2012.06128*, 2020.
[36] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A secure sharding protocol for open blockchains," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 17–30.
[37] M. Zamani, M. Movahedi, and M. Raykova, "RapidChain: Scaling blockchain via full sharding," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2018, pp. 931–948.
[38] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "Omniledger: A secure, scale-out, decentralized ledger via sharding," in *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2018, pp. 583–598.
[39] H. Dang, T. T. A. Dinh, D. Loghin, E.-C. Chang, Q. Lin, and B. C. Ooi, "Towards scaling blockchain systems via sharding," in *Proceedings of the 2019 international conference on management of data*, 2019, pp. 123–140.
[40] L. Heimbach and R. Wattenhofer, "SoK: Preventing Transaction Reordering Manipulations in Decentralized Finance," in *4th ACM Conference on Advances in Financial Technologies (AFT), Cambridge, Massachusetts, USA*, September 2022.
[41] R. Khalil, A. Gervais, and G. Felley, "NOCUST–A Securely Scalable Commit-Chain," 2018.
[42] H. Kalodner, S. Goldfeder, X. Chen, S. M. Weinberg, and E. W. Felten, "Arbitrum: Scalable, private smart contracts," in *27th USENIX Security Symposium (USENIX Security 18)*, 2018, pp. 1353–1370.
[43] L. Gudgeon, P. Moreno-Sanchez, S. Roos, P. McCorry, and A. Gervais, "Sok: Layer-two blockchain protocols," in *International Conference on Financial Cryptography and Data Security*. Springer, 2020, pp. 201–226.
[44] F. Vogelsteller and V. Buterin, "Eip-20: Erc-20 token standard," 2015, https://eips.ethereum.org/EIPS/eip-20.
[45] E. Kuo, B. Iles, and M. R. Cruz, "Ampleforth: A new synthetic commodity," 2019.

[46] J. Dafflon, J. Baylina, and T. Shababi, "Eip-777: Erc777 token standard," 2017, https://eips.ethereum.org/EIPS/eip-777.

[47] "Home - alpha finance lab," https://alphafinance.io.

[48] "Harvest finance," https://harvest.finance/.

[49] L. Heimbach, Y. Wang, and R. Wattenhofer, "Behavior of Liquidity Providers in Decentralized Exchanges," in *2021 Crypto Valley Conference on Blockchain Technology (CVCBT), Rotkreuz, Switzerland*, October 2021.

[50] L. Heimbach, E. Schertenleib, and R. Wattenhofer, "Risks and Returns of Uniswap V3 Liquidity Providers," in *4th ACM Conference on Advances in Financial Technologies (AFT), Cambridge, Massachusetts, USA*, September 2022.

[51] A. Gervais, H. Ritzdorf, G. O. Karame, and S. Capkun, "Tampering with the delivery of blocks and transactions in bitcoin," in *Conference on Computer and Communications Security*. ACM, 2015.

[52] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," in *Financial Cryptography and Data Security*. Springer, 2014, pp. 436–454.

[53] F. Schär, "Decentralized finance: On blockchain-and smart contract-based financial markets," *FRB of St. Louis Review*, 2021.

[54] "Coingecko yield farming survey 2020," https://www.coingecko.com/buzz/yield-farming-survey-2020.

[55] Y. Zhou, D. Kumar, S. Bakshi, J. Mason, A. Miller, and M. Bailey, "Erays: reverse engineering ethereum's opaque smart contracts," in *27th {USENIX} Security Symposium ({USENIX} Security 18)*, 2018, pp. 1371–1385.

[56] M. Rodler, W. Li, G. O. Karame, and L. Davi, "Evmpatch: timely and automated patching of ethereum smart contracts," in *30th {USENIX} Security Symposium ({USENIX} Security 21)*, 2021.

[57] J. Frank, C. Aschermann, and T. Holz, "{ETHBMC}: A bounded model checker for smart contracts," in *29th {USENIX} Security Symposium ({USENIX} Security 20)*, 2020, pp. 2757–2774.

[58] L. Breidenbach, P. Daian, F. Tramèr, and A. Juels, "Enter the hydra: Towards principled bug bounties and exploit-resistant smart contracts," in *27th {USENIX} Security Symposium ({USENIX} Security 18)*, 2018, pp. 1335–1352.

[59] S. So, S. Hong, and H. Oh, "Smartest: Effectively hunting vulnerable transaction sequences in smart contracts through language model-guided symbolic execution," in *30th {USENIX} Security Symposium ({USENIX} Security 21)*, 2021.

[60] M. Zhang, X. Zhang, Y. Zhang, and Z. Lin, "{TXSPECTOR}: Uncovering attacks in ethereum from transactions," in *29th {USENIX} Security Symposium ({USENIX} Security 20)*, 2020, pp. 2775–2792.

[61] S. Zhou, M. Möser, Z. Yang, B. Adida, T. Holz, J. Xiang, S. Goldfeder, Y. Cao, M. Plattner, X. Qin *et al.*, "An ever-evolving game: Evaluation of real-world attacks and defenses in ethereum ecosystem," in *29th {USENIX} Security Symposium ({USENIX} Security 20)*, 2020, pp. 2793–2810.

[62] J. Krupp and C. Rossow, "teether: Gnawing at ethereum to automatically exploit smart contracts," in *27th {USENIX} Security Symposium ({USENIX} Security 18)*, 2018, pp. 1317–1333.

[63] P. Bose, D. Das, Y. Chen, Y. Feng, C. Kruegel, and G. Vigna, "Sailfish: Vetting smart contract state-inconsistency bugs in seconds," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022.

[64] T. D. Nguyen, L. H. Pham, and J. Sun, "Sguard: Smart contracts made vulnerability-free," 2021.

[65] J. Stephens, K. Ferles, B. Mariano, S. Lahiri, and I. Dillig, "Smartpulse: Automated checking of temporal properties in smart contracts," in *2021 IEEE Symposium on Security and Privacy (SP)*, 2021.

[66] A. Permenev, D. Dimitrov, P. Tsankov, D. Drachsler-Cohen, and M. Vechev, "Verx: Safety verification of smart contracts," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020.

[67] S. So, M. Lee, J. Park, H. Lee, and H. Oh, "Verismart: A highly precise safety verifier for ethereum smart contracts," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 1678–1694.

[68] C. Schneidewind, I. Grishchenko, M. Scherer, and M. Maffei, "ethor: Practical and provably sound static analysis of ethereum smart contracts," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 621–640.

[69] J. He, M. Balunović, N. Ambroladze, P. Tsankov, and M. Vechev, "Learning to fuzz from symbolic execution with application to smart contracts," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 531–548.

[70] P. Tsankov, A. Dan, D. Drachsler-Cohen, A. Gervais, F. Buenzli, and M. Vechev, "Securify: Practical security analysis of smart contracts," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 67–82.

[71] C. Hou, M. Zhou, Y. Ji, P. Daian, F. Tramer, G. Fanti, and A. Juels, "Squirrl: Automating attack analysis on blockchain incentive mechanisms with deep reinforcement learning," in *NDSS*, 2021.

[72] T. Chen, R. Cao, T. Li, X. Luo, G. Gu, Y. Zhang, Z. Liao, H. Zhu, G. Chen, Z. He *et al.*, "Soda: A generic online detection framework for smart contracts." in *NDSS*, 2020.

[73] M. Rodler, W. Li, G. O. Karame, and L. Davi, "Sereum: Protecting existing smart contracts against re-entrancy attacks," 2019.

[74] S. Kalra, S. Goel, M. Dhawan, and S. Sharma, "Zeus: Analyzing safety of smart contracts." in *Ndss*, 2018, pp. 1–12.

[75] T. D. Nguyen, L. H. Pham, J. Sun, Y. Lin, and Q. T. Minh, "sfuzz: An efficient adaptive fuzzer for solidity smart contracts," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, 2020, pp. 778–788.

[76] N. Grech, L. Brent, B. Scholz, and Y. Smaragdakis, "Gigahorse: thorough, declarative decompilation of smart contracts," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 2019, pp. 1176–1186.

[77] J. Choi, G. Grieco, D. Kim, A. Groce, S. Kim, and S. K. Cha, "Smartian: Enhancing smart contract fuzzing with static and dynamic data-flow analyses," in *The 36th IEEE/ACM International Conference on Automated Software Engineering*. IEEE/ACM, 2021.

[78] Y. Feng, E. Torlak, and R. Bodik, "Summary-based symbolic evaluation for smart contracts," in *2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2020, pp. 1141–1152.

[79] B. Jiang, Y. Liu, and W. Chan, "Contractfuzzer: Fuzzing smart contracts for vulnerability detection," in *2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2018, pp. 259–269.

[80] H. Liu, C. Liu, W. Zhao, Y. Jiang, and J. Sun, "S-gram: towards semantic-aware security auditing for ethereum smart contracts," in *2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2018, pp. 814–819.

[81] 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*, 2016, pp. 254–269.

[82] L. Brent, A. Jurisevic, M. Kong, E. Liu, F. Gauthier, V. Gramoli, R. Holz, and B. Scholz, "Vandal: A scalable security analysis framework for smart contracts," *arXiv preprint arXiv:1809.03981*, 2018.

[83] E. Albert, P. Gordillo, B. Livshits, A. Rubio, and I. Sergey, "Ethir: A framework for high-level analysis of ethereum bytecode," in *International symposium on automated technology for verification and analysis*. Springer, 2018, pp. 513–520.

[84] I. Nikolić, A. Kolluri, I. Sergey, P. Saxena, and A. Hobor, "Finding the greedy, prodigal, and suicidal contracts at scale," in *Proceedings of the 34th Annual Computer Security Applications Conference*, 2018, pp. 653–663.

[85] S. Wu, D. Wang, J. He, Y. Zhou, L. Wu, X. Yuan, Q. He, and K. Ren, "Defiranger: Detecting price manipulation attacks on defi applications," *arXiv preprint arXiv:2104.15068*, 2021.

[86] T. Chen, X. Li, X. Luo, and X. Zhang, "Under-optimized smart contracts devour your money," in *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2017, pp. 442–446.

[87] L. Su, X. Shen, X. Du, X. Liao, X. Wang, L. Xing, and B. Liu, "Evil under the sun: Understanding and discovering attacks on ethereum decentralized applications," in *30th {USENIX} Security Symposium ({USENIX} Security 21)*, 2021.

[88] D. Perez and B. Livshits, "Smart contract vulnerabilities: Vulnerable does not imply exploited," in *30th {USENIX} Security Symposium ({USENIX} Security 21)*, 2021.

[89] C. F. Torres, R. Camino, and R. State, "Frontrunner jones and the raiders of the dark forest: An empirical study of frontrunning on the ethereum blockchain," *arXiv preprint arXiv:2102.03347*, 2021.

[90] P. Szalachowski, D. Reijsbergen, I. Homoliak, and S. Sun, "Strongchain: Transparent and collaborative proof-of-work consensus," in *28th {USENIX} Security Symposium ({USENIX} Security 19)*, 2019, pp. 819–836.

[91] C. F. Torres, M. Steichen *et al.*, "The art of the scam: Demystifying honeypots in ethereum smart contracts," in *28th {USENIX} Security Symposium ({USENIX} Security 19)*, 2019, pp. 1591–1607.

[92] L. Zhou, K. Qin, C. F. Torres, D. V. Le, and A. Gervais, "High-frequency trading on decentralized on-chain exchanges," in *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 428–445.

[93] E. Cecchetti, S. Yao, H. Ni, and A. C. Myers, "Compositional security for reentrant applications," *arXiv preprint arXiv:2103.08577*, 2021.

[94] J. Jiao, S. Kan, S.-W. Lin, D. Sanan, Y. Liu, and J. Sun, "Semantic understanding of smart contracts: Executable operational semantics of solidity," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 1695–1712.

[95] R. Zhang and B. Preneel, "Lay down the common metrics: Evaluating proof-of-work consensus protocols' security," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 175–192.

[96] M. Saad, A. Anwar, S. Ravi, and D. Mohaisen, "Revisiting nakamoto consensus in asynchronous networks: A comprehensive analysis of bitcoin safety and chainquality," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 988–1005.

[97] A. Lewis-Pye and T. Roughgarden, "How does blockchain security dictate blockchain implementation?" in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 1006–1019.

[98] P. Das, A. Erwig, S. Faust, J. Loss, and S. Riahi, "The exact security of bip32 wallets," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 1020–1042.

[99] K. Li, Y. Wang, and Y. Tang, "Deter: Denial of ethereum txpool services," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 1645–1667.

[100] M. Saad, S. Chen, and D. Mohaisen, "Syncattack: Double-spending in bitcoin without mining power," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 1668–1685.

[101] M. Mirkin, Y. Ji, J. Pang, A. Klages-Mundt, I. Eyal, and A. Juels, "Bdos: Blockchain denial-of-service," in *Proceedings of the 2020 ACM SIGSAC conference on Computer and Communications Security*, 2020, pp. 601–619.

[102] T. Chen, Y. Zhang, Z. Li, X. Luo, T. Wang, R. Cao, X. Xiao, and X. Zhang, "Tokenscope: Automatically detecting inconsistent behaviors of cryptocurrency tokens in ethereum," in *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, 2019, pp. 1503–1520.

[103] P. Das, S. Faust, and J. Loss, "A formal treatment of deterministic wallets," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 651–668.

[104] I. Tsabary and I. Eyal, "The gap game," in *Proceedings of the 2018 ACM SIGSAC conference on Computer and Communications Security*, 2018, pp. 713–728.

[105] L. Kiffer, R. Rajaraman, and A. Shelat, "A better method to analyze blockchain consistency," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 729–744.

[106] K. Li, J. Chen, X. Liu, Y. Tang, X. Wang, and X. Luo, "As strong as its weakest link: How to break blockchain dapps at rpc service," in *28th Annual Network and Distributed System Security Symposium, NDSS*, 2021, pp. 21–25.

[107] G. Bissias and B. N. Levine, "Bobtail: Improved blockchain security with low-variance mining." in *NDSS*, 2020.

[108] D. Perez and B. Livshits, "Broken metre: Attacking resource metering in evm," 2020.

[109] Z. Wan, X. Xia, D. Lo, J. Chen, X. Luo, and X. Yang, "Smart contract security: a practitioners' perspective," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 2021, pp. 1410–1422.

[110] T. Durieux, J. F. Ferreira, R. Abreu, and P. Cruz, "Empirical review of automated analysis tools on 47,587 ethereum smart contracts," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, 2020, pp. 530–541.

[111] S. Hwang and S. Ryu, "Gap between theory and practice: An empirical study of security patches in solidity," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, 2020, pp. 542–553.

[112] L. Liu, L. Wei, W. Zhang, M. Wen, Y. Liu, and S.-C. Cheung, "Characterizing transaction-reverting statements in ethereum smart contracts," *arXiv preprint arXiv:2108.10799*, 2021.

[113] Y. Xue, M. Ma, Y. Lin, Y. Sui, J. Ye, and T. Peng, "Cross-contract static analysis for detecting practical reentrancy vulnerabilities in smart contracts," in *2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2020, pp. 1029–1040.

[114] S. Grossman, I. Abraham, G. Golan-Gueta, Y. Michalevsky, N. Rinetzky, M. Sagiv, and Y. Zohar, "Online detection of effectively callback free objects with applications to smart contracts," *Proceedings of the ACM on Programming Languages*, vol. 2, no. POPL, pp. 1–28, 2017.

[115] A. Li, J. A. Choi, and F. Long, "Securing smart contract with runtime validation," in *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2020, pp. 438–453.

[116] L. Brent, N. Grech, S. Lagouvardos, B. Scholz, and Y. Smaragdakis, "Ethainter: A smart contract security analyzer for composite vulnerabilities," in *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2020, pp. 454–469.

[117] S. M. Beillahi, G. Ciocarlie, M. Emmi, and C. Enea, "Behavioral simulation for smart contracts," in *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2020, pp. 470–486.

[118] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the security and performance of proof of work blockchains," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 3–16.

[119] K. Wüst and A. Gervais, "Ethereum eclipse attacks," ETH Zurich, Tech. Rep., 2016.

[120] Y. Marcus, E. Heilman, and S. Goldberg, "Low-resource eclipse attacks on ethereum's peer-to-peer network." *IACR Cryptol. ePrint Arch.*, vol. 2018, p. 236, 2018.

[121] R. Rahimian and J. Clark, "Tokenhook: Secure erc-20 smart contract," *arXiv preprint arXiv:2107.02997*, 2021.

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

[123] J. Guarnizo and P. Szalachowski, "Pdfs: practical data feed service for smart contracts," in *European Symposium on Research in Computer Security*. Springer, 2019, pp. 767–789.

[124] L. Breidenbach, C. Cachin, B. Chan, A. Coventry, S. Ellis, A. Juels, F. Koushanfar, A. Miller, B. Magauran, D. Moroz *et al.*, "Chainlink 2.0: Next steps in the evolution of decentralized oracle networks," 2021.

[125] A. YAISH, G. STERN, and A. ZOHAR, "Uncle maker:(time) stamping out the competition in ethereum," 2022.

[126] "Wakaswap smart contract security audit," https://waka-finance-2.gitbook.io/waka-finance/documentation/audit, 2021, beosin.

[127] "Sato smart contract security audit," https://sato.trade/Smart_contract_security_audit_report%E2%80%94SATO.pdf, 2021, beosin.

[128] "Pinecone smart contract security audit," https://safefiles.defiyield.info/safe/files/audit/pdf/REP_Pinecone_Finance_2021_09_28.pdf, 2021, beosin.

[129] "Ctoken smart contract security audit," , 2021, beosin.

[130] "Beatsqure smart contract security audit," 2021, beosin.

[131] "Rabbit.fi smart contract security audit," https://github.com/peckshield/publications/blob/master/audit_reports/PeckShield-Audit-Report-Rabbit-v1.0.pdf, 2021, peckshield.

[132] "Hegic smart contract security audit," https://safefiles.defiyield.info/safe/files/audit/pdf/PeckShield_Audit_Report_Hegic_v1_0.pdf, 2021, peckshield.

[133] "Deri v2 smart contract security audit," https://github.com/peckshield/publications/blob/693bdb69e3e3e422b4f7e1f3130d841e631b4dab/audit_reports/PeckShield-Audit-Report-DeriV2-v1.0.pdf, 2021, peckshield.

[134] "Coin98 smart contract security audit," https://safefiles.defiyield.info/safe/files/audit/pdf/PeckShield_Audit_Report_COIN98_v1_0.pdf, 2021, peckshield.

[135] "Angrymining smart contract security audit," https://github.com/peckshield/publications/blob/master/audit_reports/PeckShield-Audit-Report-AngryMining-v1.0rc.pdf, 2021, peckshield.

[136] "Jswap smart contract security audit," https://www.slowmist.com/en/security-audit-certificate.html?id=

928799684ad96ef4ed4b0c0fb12a5fae085456f874b19dc4300195b32a5a143f, 2021, slowMist.
[137] "Supremex smart contract security audit," https://www.slowmist.com/security-audit-certificate.html?id=769a2454892441cfc9730e3fc39db48b75e9bb05ad33527ce1736342ff8ea8e3, 2021, slowMist.
[138] "Solyard smart contract security audit," https://www.slowmist.com/security-audit-certificate.html?id=53e38102e25c3c6d8a8136edc7e859fde08ed93189c1535d642bb1cd656e581f, 2021, slowMist.
[139] "Cook finance smart contract security audit," https://github.com/slowmist/Knowledge-Base/blob/master/open-report/SlowMist%20Audit%20Report%20-%20Cook%20Finance.pdf, 2021, slowMist.
[140] "Defi saver smart contract security audit," https://github.com/defisaver/defisaver-v3-contracts/blob/main/audits/Consensys-Mar-2021.pdf, 2021, consensys.
[141] "Fei tribechief smart contract security audit," https://consensys.net/diligence/audits/2021/07/fei-tribechief/, 2021, consensys.
[142] "Gitcoin smart contract security audit," https://consensys.net/diligence/audits/2021/04/gitcoin-token-distribution/, 2021, consensys.
[143] "Wheat smart contract security audit," https://consensys.net/diligence/audits/2021/06/growthdefi-wheat/, 2021, consensys.
[144] "Umbra smart contract security audit," https://consensys.net/diligence/audits/2021/03/umbra-smart-contracts/, 2021, consensys.
[145] "Zoo smart contract security audit," https://www.certik.com/projects/zoocrypto, 2021, certik.
[146] "Trister's lend smart contract security audit," https://www.certik.com/projects/tristerlend, 2021, certik.
[147] "Rezerve smart contract security audit," https://www.certik.com/projects/rezerve, 2021, certik.
[148] "Lfw smart contract security audit," https://www.certik.com/projects/legendfantasywar, 2021, certik.
[149] "gamedao smart contract security audit," https://www.certik.com/projects/gamedao, 2021, certik.
[150] "Complifi smart contract security audit," https://github.com/trailofbits/publications/blob/master/reviews/CompliFi.pdf, 2021, trail of Bits.
[151] "Frax finance smart contract security audit," https://github.com/trailofbits/publications/blob/master/reviews/FraxFinance.pdf, 2021, trail of Bits.
[152] "Yearnv2 smart contract security audit," https://github.com/trailofbits/publications/blob/master/reviews/YearnV2Vaults.pdf, 2021, trail of Bits.
[153] "Alpha homora smart contract security audit," https://blog.openzeppelin.com/alpha-homora-v2/, 2021, open Zeppelin.
[154] "Celo smart contract security audit," https://blog.openzeppelin.com/celo-contracts-audit/, 2021, open Zeppelin.
[155] "Fei smart contract security audit," https://blog.openzeppelin.com/fei-protocol-audit/, 2021, open Zeppelin.
[156] G. A. Akerlof and J. L. Yellen, "A near-rational model of the business cycle, with wage and price inertia," *The Quarterly Journal of Economics*, vol. 100, pp. 823–838, 1985.
[157] N. Strong, "Modelling abnormal returns: A review article," *Journal of Business Finance & Accounting*, vol. 19, no. 4, pp. 533–553, 1992.
[158] A. Brauneis and R. Mestel, "Cryptocurrency-portfolios in a mean-variance framework," *Finance Research Letters*, 2019.
[159] G. M. Caporale and A. Plastun, "Daily abnormal price changes and trading strategies in the forex," *Journal of Economic Studies*, 2020.
[160] S. Shanaev and B. Ghimire, "Efficient scholars: academic attention and the disappearance of anomalies," *The European Journal of Finance*, vol. 27, no. 3, pp. 278–304, 2021.
[161] I. Venezia *et al.*, "Appearance and disappearance of anomalies," *World Scientific Book Chapters*, pp. 223–233, 2018.
[162] J. Cotter and N. McGeever, "Are equity market anomalies disappearing? evidence from the uk," 2018.
[163] "Defi plus." [Online]. Available: https://defipulse.com/
[164] "Circuit breaker definition," https://www.investopedia.com/terms/c/circuitbreaker.asp.
[165] K. G. Jöreskog, "A general method for estimating a linear structural equation system," *ETS Research Bulletin Series*, vol. 1970, no. 2, pp. i–41, 1970.
[166] C. Fornell and D. F. Larcker, "Structural equation models with unobservable variables and measurement error: Algebra and statistics," 1981.
[167] K. G. Jöreskog and D. Sörbom, "Recent developments in structural equation modeling," *Journal of marketing research*, vol. 19, no. 4, pp. 404–416, 1982.
[168] Jöreskog, Karl G and Sörbom, Dag, *LISREL 8: Structural equation modeling with the SIMPLIS command language*. Scientific Software International, 1993.
[169] R. H. Hoyle, *Structural equation modeling: Concepts, issues, and applications*. Sage, 1995.
[170] Hoyle, Rick H, "The structural equation modeling approach: Basic concepts and fundamental issues." 1995.
[171] D. Gefen, D. Straub, and M.-C. Boudreau, "Structural equation modeling and regression: Guidelines for research practice," *Communications of the association for information systems*, vol. 4, no. 1, p. 7, 2000.
[172] R. E. Schumacker and R. G. Lomax, *A beginner's guide to structural equation modeling*. psychology press, 2004.
[173] J. B. Ullman and P. M. Bentler, "Structural equation modeling," *Handbook of Psychology, Second Edition*, vol. 2, 2012.
[174] B. M. Byrne, *Structural equation modeling with Mplus: Basic concepts, applications, and programming*. routledge, 2013.
[175] R. B. Kline, *Principles and practice of structural equation modeling*. Guilford publications, 2015.
[176] D. Harrington, *Confirmatory factor analysis*. Oxford university press, 2009.
[177] T. A. Brown and M. T. Moore, "Confirmatory factor analysis," *Handbook of structural equation modeling*, vol. 361, p. 379, 2012.
[178] T. A. Brown, *Confirmatory factor analysis for applied research*. Guilford publications, 2015.
[179] E. J. Wolf and T. A. Brown, "Structural equation modeling," in *The Oxford Handbook of Research Strategies for Clinical Psychology*, 1996.
[180] G. Meshcheryakov, A. A. Igolkina, and M. G. Samsonova, "semopy 2: A structural equation modeling package with random effects in python," *arXiv preprint arXiv:2106.01140*, 2021.
[181] D. W. Woods and R. Böhme, "Systematization of knowledge: Quantifying cyber risk," in *IEEE Symposium on Security & Privacy*, 2021.
[182] B. T. West, K. B. Welch, and A. T. Galecki, *Linear mixed models: a practical guide using statistical software*. Chapman and Hall/CRC, 2006.
[183] P. G. Freund and M. A. Rubin, "Dynamics of dimensional reduction," *Physics Letters B*, vol. 97, no. 2, pp. 233–235, 1980.
[184] A. Satorra and P. M. Bentler, "Corrections to test statistics and standard errors in covariance structure analysis." 1994.
[185] P. M. Bentler, "Comparative fit indexes in structural models." *Psychological bulletin*, vol. 107, no. 2, p. 238, 1990.
[186] P. M. Bentler and D. G. Bonett, "Significance tests and goodness of fit in the analysis of covariance structures." *Psychological bulletin*, vol. 88, no. 3, p. 588, 1980.
[187] B. Wheaton, B. Muthen, D. F. Alwin, and G. F. Summers, "Assessing reliability and stability in panel models," *Sociological methodology*, vol. 8, pp. 84–136, 1977.
[188] J. Sun, "Assessing goodness of fit in confirmatory factor analysis," *Measurement and evaluation in counseling and development*, vol. 37, no. 4, pp. 240–256, 2005.
[189] P. Barrett, "Structural equation modelling: Adjudging model fit," *Personality and Individual differences*, vol. 42, no. 5, pp. 815–824, 2007.
[190] DeFiYield. (2022) DeFiYield. https://defiyield.app.
[191] L. Kiffer, D. Levin, and A. Mislove, "Analyzing ethereum's contract topology," in *Proceedings of the Internet Measurement Conference 2018*, 2018, pp. 494–499.
[192] N. He, L. Wu, H. Wang, Y. Guo, and X. Jiang, "Characterizing code clones in the ethereum smart contract ecosystem," in *International Conference on Financial Cryptography and Data Security*. Springer, 2020, pp. 654–675.
[193] Z. Gao, L. Jiang, X. Xia, D. Lo, and J. Grundy, "Checking smart contracts with structural code embedding," *IEEE Transactions on Software Engineering*, 2020.
[194] D. Zhu, J. Pang, X. Zhou, and W. Han, "Similarity measure for smart contract bytecode based on cfg feature extraction," in *2021 International Conference on Computer Information Science and Artificial Intelligence (CISAI)*. IEEE, 2021, pp. 558–562.
[195] J. Quesnelle, "On the linkability of zcash transactions," *arXiv preprint arXiv:1712.01210*, 2017.
[196] G. Kappos, H. Yousaf, M. Maller, and S. Meiklejohn, "An empirical analysis of anonymity in zcash," in *27th {USENIX} Security Symposium ({USENIX} Security 18)*, 2018, pp. 463–477.

[197] M. Tran, L. Luu, M. S. Kang, I. Bentov, and P. Saxena, "Obscuro: A bitcoin mixer using trusted execution environments," in *Proceedings of the 34th Annual Computer Security Applications Conference*, 2018, pp. 692–701.

[198] J. Pakki, Y. Shoshitaishvili, R. Wang, T. Bao, and A. Doupé, "Everything you ever wanted to know about bitcoin mixers (but were afraid to ask)," in *International Conference on Financial Cryptography and Data Security*. Springer, 2021, pp. 117–146.

[199] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. M. Wing, "Automated generation and analysis of attack graphs," in *Proceedings 2002 IEEE Symposium on Security and Privacy*. IEEE, 2002, pp. 273–284.

[200] L. Wang, A. Singhal, and S. Jajodia, "Toward measuring network security using attack graphs," in *Proceedings of the 2007 ACM workshop on Quality of protection*, 2007.

[201] M. A. Khan and M. Hussain, "Cyber security quantification model," in *Proceedings of the 3rd international conference on Security of information and networks*, 2010, pp. 142–148.

[202] Z. Amin, "A practical road map for assessing cyber risk," *Journal of Risk Research*, vol. 22, no. 1, pp. 32–43, 2019.

[203] C. K. Roy, J. R. Cordy, and R. Koschke, "Comparison and evaluation of code clone detection techniques and tools: A qualitative approach," *Science of computer programming*, vol. 74, no. 7, pp. 470–495, 2009.

[204] I. U. Haq and J. Caballero, "A survey of binary code similarity," *ACM Computing Surveys (CSUR)*, vol. 54, no. 3, pp. 1–38, 2021.

[205] B. S. Baker, "On finding duplication and near-duplication in large software systems," in *Proceedings of 2nd Working Conference on Reverse Engineering*. IEEE, 1995, pp. 86–95.

[206] I. D. Baxter, A. Yahin, L. Moura, M. Sant'Anna, and L. Bier, "Clone detection using abstract syntax trees," in *Proceedings. International Conference on Software Maintenance (Cat. No. 98CB36272)*. IEEE, 1998, pp. 368–377.

[207] R. Komondoor and S. Horwitz, "Using slicing to identify duplication in source code," in *International static analysis symposium*. Springer, 2001, pp. 40–56.

[208] S. Ducasse, M. Rieger, and S. Demeyer, "A language independent approach for detecting duplicated code," in *Proceedings IEEE International Conference on Software Maintenance-1999 (ICSM'99).'Software Maintenance for Business Change'(Cat. No. 99CB36360)*. IEEE, 1999, pp. 109–118.

[209] M. White, M. Tufano, C. Vendome, and D. Poshyvanyk, "Deep learning code fragments for code clone detection," in *2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2016, pp. 87–98.

[210] M. Novak, M. Joy, and D. Kermek, "Source-code similarity detection and detection tools used in academia: a systematic review," *ACM Transactions on Computing Education (TOCE)*, vol. 19, no. 3, pp. 1–37, 2019.

[211] C. K. Roy and J. R. Cordy, "A survey on software clone detection research," *Queen's School of computing TR*, vol. 541, no. 115, pp. 64–68, 2007.

[212] H. Liu, Z. Yang, Y. Jiang, W. Zhao, and J. Sun, "Enabling clone detection for ethereum via smart contract birthmarks," in *2019 IEEE/ACM 27th International Conference on Program Comprehension (ICPC)*. IEEE, 2019, pp. 105–115.

[213] X. Chen, P. Liao, Y. Zhang, Y. Huang, and Z. Zheng, "Understanding code reuse in smart contracts," in *2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2021, pp. 470–479.

[214] M. Kondo, G. A. Oliva, Z. M. J. Jiang, A. E. Hassan, and O. Mizuno, "Code cloning in smart contracts: a case study on verified contracts from the ethereum blockchain platform," *Empirical Software Engineering*, vol. 25, no. 6, pp. 4617–4675, 2020.

[215] D. Zhu, F. Yue, J. Pang, X. Zhou, W. Han, and F. Liu, "Bytecode similarity detection of smart contract across optimization options and compiler versions based on triplet network," *Electronics*, vol. 11, no. 4, p. 597, 2022.

[216] E. Androulaki, G. O. Karame, M. Roeschlin, T. Scherer, and S. Capkun, "Evaluating user privacy in bitcoin," in *International Conference on Financial Cryptography and Data Security*. Springer, 2013, pp. 34–51.

[217] S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, D. McCoy, G. M. Voelker, and S. Savage, "A fistful of bitcoins: characterizing payments among men with no names," in *Proceedings of the 2013 conference on Internet measurement conference*. ACM, 2013, pp. 127–140.

[218] H. Yousaf, G. Kappos, and S. Meiklejohn, "Tracing transactions across cryptocurrency ledgers," in *28th {USENIX} Security Symposium ({USENIX} Security 19)*, 2019, pp. 837–850.

[219] F. Victor, "Address clustering heuristics for ethereum," in *International Conference on Financial Cryptography and Data Security*. Springer, 2020, pp. 617–633.

Algorithm 1: Source of Funds Tracing Algorithm

Input: Current highest block $b_{current}$; Tracing address T; Starting block for post-incident tracing b_{post};

 # Transaction nonce equals the number of transaction sent; **Algorithm** *OneHopPreIncidentTracing(T, $b_{current}$)*:

- $b_{first} \leftarrow$ Binary search between block 0 and $b_{current}$ where T's nonce equals 0 in b_{first}, and T's nonce greater than 0 in $b_{first}+1$.
- $b_{funding} \leftarrow$ Binary search between block 0 and b_{first} where T's balance is greater than 0 in $b_{funding}$ and T's balance equals 0 in $b_{funding}-1$.
- **foreach** $tx \in \{tx^0_{b_{funding}}, \ldots\}$ **do**
  - **if** *Replay tx and finds native token transfer to T* **then**
    - **return** tx
  - **end**
- **end**

end

Suspects ($\mathbb{A}^*$)	Pattern	Incident	Date
0x8641dF2D7C730A8A24db86693fc39F7A74Dd4e9D	$\mathbb{A}^* \xrightarrow{2} \mathbb{A}$	WildCredit	May 27, 2021
	$\mathbb{A}^* \xrightarrow{2} \mathbb{A}$	DeFiSaver	Oct 08, 2020
	$\mathbb{A}^* \xrightarrow{1} \mathbb{A}$	DODO	Mar 08, 2021
	$\mathbb{A}^* \xrightarrow{2} \mathbb{A}$	VisorFinance	Nov 26, 2021
	$\mathbb{A}^* \xrightarrow{1} \mathbb{A}$	MakerDAO	Mar 12, 2020
0x5b1839B202b67Db64e402a1501cf4f52f5eff03c	$\mathbb{A}^* \xrightarrow{3} \mathbb{A}$	BuccaneerFi	Mar 27, 2020
	$\mathbb{A}^* \xrightarrow{1} \mathbb{A}$	InfinityToken	Jan 26, 2022
0xC1A065a2d29995692735c82d228B63Df1732030E	$\mathbb{A}^* \xrightarrow{2} \mathbb{A}$	SodaFinance	Sep 20, 2020
	$\mathbb{A}^* \xrightarrow{1} \mathbb{A}$	BuccaneerFi	Aug 24, 2020
0xE4b3dD9839ed1780351Dc5412925cf05F07A1939	$\mathbb{A}^* \xrightarrow{2} \mathbb{A}$	bZx	Sep 13, 2020
	$\mathbb{A}^* \xrightarrow{1} \mathbb{A}$	ForceDAO	Apr 04, 2021
0x6bE5A267B04E9f24CdC1824fd38d63c436be91aB	$\mathbb{A}^* \xrightarrow{2} \mathbb{A}$	PancakeHunny	Jun 03, 2021
	$\mathbb{A}^* \xrightarrow{1} \mathbb{A}$	BoggedFinance	May 22, 2021
0x22B84d5FFeA8b801C0422AFe752377A64Aa738c2	$\mathbb{A}^* \xrightarrow{8} \mathbb{A}$	MakerDAO	Mar 12, 2020
	$\mathbb{A}^* \xrightarrow{9} \mathbb{A}$	BadgerDAO	Nov 21, 2021

TABLE X: Linked adversaries based on pre-incident trace.

APPENDIX A
TRACING SOURCE OF FUND

Algorithm 1 identifies the funding transaction $X \rightarrow Y$ for any arbitrary address Y. Table X shows the linked adversaries based on source of fund tracing. We have successfully identified in total six clusters, where the adversaries in five of the clusters are linked with three hops.