SoK: Decentralized Finance (DeFi) Incidents

Liyi Zhou †*, Xihan Xiong †*, Jens Ernstberger †, Stefanos Chaliosas †, Zhipeng Wang †, Ye Wang ***, Kaifu 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. 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 “permissionless 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.

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

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 II). Section III studies the collected DeFi incidents with statistical analysis. Section IV 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 VII related works in VIII and a closure in VIII.
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):**

- **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. 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. 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, Eden network, Bloxroute, and Ethermine). 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 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 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 \( \rho \), \( \rho \)–\( \rho \)\). 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.

**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:** \( T(s \in S, tx \in TX) \rightarrow S \) 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 a 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 \( S(b_{i+1}) \) stems from the sequential execution of all transactions in block \( b_{i+1} \) on \( S(b_i) \) (cf. Equation [1]).

\[
S(b_{i+1}) = T(\ldots T(T(S(b_i), tx_0, b_{i+1}), tx_1, b_{i+1}), \ldots)
\]

**SC and Layer 2 Blockchain (L2) Incentive Mechanism:**

- **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 \( \rho \) and Harvest Finance \( \rho \) integrate automated market makers (i.e., Uniswap \( \rho \)) and lending platforms (i.e., Compound \( \rho \)).

**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 \( \rho \), \( \rho \) (e.g., Sushiswap\( \rho \) and Curve\( \rho \)).
Attacks: the following two types:

1. **Section III to relatively compare all observed DeFi attacks.**

   - Utilities, goals, knowledge and capabilities, to engender a common reference frame which we subsequently apply in on-chain smart contracts, etc.

   - **Delegate off-chain oracle data to raw on-chain data**

   - **DeFi protocol implementation may consist of:**
     - **Operations:** resulting from coding mistakes, such as arithmetic error, casting error, inconsistent access control, function reentrancy, etc.

   - **DeFi white hat hackers (also known as ethical hackers) are project developers realizing the protocol designs;**

   - **Adversarial Knowledge:**
     - **Table II differentiates between 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.

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

     1. **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 classify market manipulations as attacks, which necessarily require a vulnerable system or system state; and

     2. **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. 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 another protocol, which has no money market, or to increase the monetary nature. We define the monetary utility function as the total increase in market value of $A$’s cryptocurrency asset portfolio, which $A$ aims to maximize.

   - **$U_2$: Non-monetary:** $A$ 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.

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

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

     2. **$K_2$: CON:** may use mixer services to break account linkability.

     3. **$K_3$: SC:** may fork or append on a forked chain in an attempt to catch up and overwrite the longest chain.

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

   - **TABLE II: Categorization of adversarial knowledge levels.**

   - **TABLE II**

   - **TABLE II**
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.

- **$K_2$-Sequencer**: $A$ obtains the following information, if $A$ 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 $A$ has early access to external market prices, oracle updates, or the wallet passphrases of an operator.

(iv) **Adversarial Capabilities**: Table I outlines the adversarial capabilities and required knowledge. Note that $A$ 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 $A$ 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 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, model checking, bug bounties, and reverse engineering. 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.

---

**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. 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: (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 III).

- **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

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---
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.

| Incident Cause | Incident Type | Se-Ki, Surveys | Tools | Academic Papers (We abbreviate Usenix Security as UNX) | Audit Reports | Gap Visualization |
|----------------|--------------|---------------|-------|---------------------------------------------------|--------------|-------------------|
| Network layer transparency | Transmission content transparency | ■ | | | | |
| | Propagation transparency | | | | | |
| |休闲层 | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
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.

- **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 the \( \beta \) 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 at tick \( t \in [T_{s-144}, T_s] \) respectively, \( \alpha_i \) is the constant, and \( \epsilon_{i,t} \) is the error term.

\[
R_{i,t} - r_{f,t} = \alpha_i + \beta_i \cdot (R_{mkt,t} - r_{f,t}) + \epsilon_{i,t}
\]  

2. Equation (3) calculates the ARs for each tick over the event time frame of \( [T_s, T_{s+144}] \), where \( \beta_i \) is the fitted \( \beta \) coefficient, \( E[R_{i,t}] \) 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 \( \beta_i \) which is fitted within the estimation window.

\[
AR_{i,t} = R_{i,t} - E[R_{i,t}] = R_{i,t} - (\alpha_i + \beta_i (R_{mkt,t} - r_{f,t}) + r_{f,t})
\]  

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)

\[
CAR_i = \min_t \left[ \sum_{t' \leq t} AR_{i,t'} \right]
\]

- **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\[14\] and DeFiLlama\[15\].

- **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.

**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 \[30–33, 89\]
recurring exploits on a group of forked protocols\textsuperscript{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 million USD (30.2%). In contrast, DEX protocols have the biggest TVL (27.7 million 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)\textsuperscript{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

\textsuperscript{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 the two weeks, the copycats Autoshaft, MerlinLab and PancakeHunny were exploited in a similar fashion: the adversary (i) exploited the vulnerability of \texttt{mintFor/mintForV2} function to manipulate LP token prices and (ii) used cross-chain bridge and TC to launder money.
hybrid of confirmatory factor analysis (CFA) and latent structural regression. An SEM model encompasses two sub-models (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.

### Why SEM

The literature 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 and dimensional reduction techniques. 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: 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. We fit our SEM using an open-sourced library, semopy (cf. Figure 6).

### Fitness

Our model is examined using a collection of indices, including (i) the adjusted Chi-square ($\chi^2_{adj}$) (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.

### 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.

### D. Emergency Pause

DeFi protocols may support an emergency mechanism, which is analogous to circuit breakers in conventional centralized exchanges. This section examines the speed at which DeFi protocols initiate an emergency mechanism (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 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. 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.

### 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

| Duration after the incident starts | Number of protocols | Percentage (out of 87 protocols) |
|-----------------------------------|---------------------|---------------------------------|
| ≤ 1h                              | 1                   | 2%                              |
| ≤ 6h                              | 24                  | 47%                             |
| ≤ 12h                             | 11                  | 22%                             |
| ≤ 24h                             | 7                   | 14%                             |
| ≤ 48h                             | 8                   | 16%                             |
between the adversarial contract deployment ($t_{\text{deploy}}$) and the execution time of the first malicious state transition ($t_{\text{first}}$). 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 $t_{\text{deploy}}$ and $t_{\text{first}}$ 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 $A$ 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 ± 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.

**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.

| Category | Similarity Threshold | 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 | 56 | | 31 | 13 | 18 |
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 chooses 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 sophisticated 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. $A$ 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.

- 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 $A$ is linked to the withdrawal, but not whether the withdrawal 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 mixer 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\textsuperscript{24} to identify the addresses of centralized exchanges and cross-chain bridges.

\textsuperscript{24}https://etherscan.io/labelcloud and https://bscscan.com/labelcloud

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

| Layers | Surveys/SoKs | Tools | Papers | Audit reports | Incidents |
|--------|-------------|-------|--------|---------------|-----------|
| Total  | 4 (7%) ♦ 29  | ♦ -   | ♣ -   | ♣ -           | ♣ -       |
| NET    | 4 (7%) ♦ 19% | ♣ -   | ♣ -   | ♣ -           | ♣ - 8%    |
| CON    | 3 (4%) ♦ 13% | ♣ -   | ♣ -   | ♣ -           | ♣ - 12%   |
| SC     | 6 (9%) ♦ 21% | ♣ -   | ♣ -   | ♣ -           | ♣ - 20%   |
| PRO    | 5 (7%) ♦ 13% | ♣ -   | ♣ -   | ♣ -           | ♣ - 14%   |
| AUX    | 4 (7%) ♦ 16% | ♣ -   | ♣ -   | ♣ -           | ♣ - 56%   |

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 blockchain.

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 $C^5_{\text{PRO}}$ 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] 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 ($C^5_{\text{PRO}}$), 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., $C^5_{\text{PRO}}$). 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

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

26i.e. the following incident types: (i) token standard incompatibility; (ii) camouflage a token contract or (iii) camouflage a non-token contract
Categorize However, their examinations on protocol layer vulnerabilities (i.e., the capability $C^\text{PRO}_A$). Surprisingly, our empirical results support Su et al. [11] (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 tracking 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, Swiss-Borg SA, Nimiq Foundation, and Lucerne University of Applied Sciences and Arts, Switzerland.

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), Genova, May 2022.

[5] “Aave,” https://github.com/aave/aave-protocol, 2020.

[6] “Compound finance,” https://compound.finance/.

[7] “SyntheticX,” https://www.syntheticx.io/.

[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. Reisjbergen, 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. Nijlla, 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. Nijlla, 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. Sanmreen and M. H. Alali, “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. Nadihalli, and R. Wattenhofer, “TWP 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/.

[23] A. Hinteregger and B. Hashifier, “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. Barrang, 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, “AnM: Autonomous coin mixer with privacy preserving reward distribution,” ACM Advances in Financial Technologies, AFT, 2021.

[27] K. Qin, L. Zhou, Y. Afonin, L. Lazzarette, 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. Golfdeler, T. Keh, 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, “Frontrunning 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. Golfdeler, 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. Baterina, “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: Erc-777 token standard,” 2017, https://eips.ethereum.org/EIPS/eip-777.

[47] J. A. Berg, R. Fritsch, L. Heimbach, and R. Wattenhofer, “Behavior of Liquidity Providers in Decentralized Exchanges,” in 2021 Crypto Valley Con-
T. D. Nguyen, L. H. Pham, and J. Sun, “Sguard: Smart contracts made
P. Bose, D. Das, Y. Chen, Y. Feng, C. Kruegel, and G. Vigna, “Sailfish: Automated checking of temporal properties in smart contracts,” in 27th {USENIX} Security Symposium ({USENIX} Security 21), 2021.
J. Frank, C. Aschermann, and T. Holz, “{EETHMC}: A bounded model checker for smart contracts,” in 29th {USENIX} Security Symposium ({USENIX} Security 20), 2020, pp. 2757–2774.
L. Breidenbach, P. Daian, F. Tramer, 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.
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.
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.
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.
J. Krupp and C. Rossow, “teether: Gnawing at ethereum to automati-
cation of real-world attacks and defenses in ethereum ecosystem,” in 30th {USENIX} Security Symposium ({USENIX} Security 21), 2021.
J. Krupp, F. Bödeker, and C. Rossow, “Smartian: Enhancing smart contract fuzzing with static and dynamic data-flow analyses,” in The 36th {IEEE} International Conference on Automated Software Engineering. {IEEE}, 2021.
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 2018 ACM SIGSAC Conference on Computer and Communications Security, 2020, pp. 635–643.
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.
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.
L. Su, S. Chen, X. Du, X. Liao, W. 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.
D. Perez and B. Livshits, “Smart contract vulnerabilities: Vulnerable data-flow analyses,” in 30th {USENIX} Security Symposium ({USENIX} Security 21), 2021.
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.
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.
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.
Hoyle, Rick H., “The structural equation modeling approach: Basic concepts and fundamental issues.” 1995.

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.

R. E. Schumacker and R. G. Lomax, A beginner's guide to structural equation modeling. psychology press, 2004.

J. B. Ullman and F. M. Bentler, “Structural equation modeling,” Handbook of Psychology, Second Edition, vol. 2, 2012.

B. M. Byrne, Structural equation modeling with Mplus: Basic concepts, applications, and programming. routledge, 2013.

R. B. Kline, Principles and practice of structural equation modeling. Guilford publications, 2015.

D. Harrington, Confirmatory factor analysis. Oxford university press, 2009.

T. A. Brown and M. T. Moore, “Confirmatory factor analysis,” Handbook of structural equation modeling, vol. 361, p. 379, 2012.

T. A. Brown, Confirmatory factor analysis for applied research. Guilford publications, 2015.

E. J. Wolf and T. A. Brown, “Structural equation modeling,” in The Oxford Handbook of Research Strategies for Clinical Psychology, 1996.

G. Meshcheryakov, A. A. Igolkin, and M. G. Samsonova, “sempoly 2: A structural equation modeling package with random effects in python,” arXiv preprint arXiv:2106.01140, 2021.

D. W. Woods and R. Böhme, “Systematization of knowledge: Quantifying cyber risk,” in IEEE Symposium on Security & Privacy, 2021.

B. T. West, K. B. Welch, and A. T. Galecki, Linear mixed models: a practical guide using statistical software. Chapman and Hall/CRC, 2006.

P. G. Freund and M. A. Rubin, “Dynamics of dimensional reduction,” Letters B, vol. 97, no. 2, pp. 233–235, 1980.

A. Satorra and P. M. Bentler, “Corrections to test statistics and standard errors in covariance structure analysis.” 1994.

P. M. Bentler, “Comparative fit indexes in structural models.” Psychological bulletin, vol. 107, no. 2, p. 238, 1990.

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.

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.

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.

P. Barrett, “Structural equation modelling: Adjudging model fit,” Personality and Individual differences, vol. 42, no. 5, pp. 815–824, 2007.

DeFiYield. (2022) DeFiYield. https://defiyield.app.

L. Kiffer, D. Levin, and A. Mislove, “Analyzing ethereum’s contract topology,” in Proceedings of the Internet Measurement Conference 2018, pp. 494–499, 2018.

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.

Z. Gao, L. Jiang, X. Xia, D. Lo, and J. Grundy, “Checking smart contracts with structural code embedding.” IEEE Transactions on Software Engineering, 2020.

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.

J. Quesnelle, “On the linkability of zcash transactions,” arXiv preprint arXiv:1712.01210, 2017.

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.

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.

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.

K. G. Joreskog, “A general method for estimating a linear structural equation system,” ETS Research Bulletin Series, vol. 1970, no. 2, pp. 1–41, 1970.

C. Fornell and D. F. Larcker, “Structural equation models with unobservable variables and measurement error: Algebra and statistics,” 1981.

K. G. Joreskog and D. Sorbom, “Recent developments in structural equation modeling,” Journal of marketing research, vol. 19, no. 4, pp. 404–416, 1982.

K. G. Joreskog, Karl G and Sorbom, Dag, LISREL 8: Structural equation modeling with the SIMPLIS command language. Scientific Software International, 1993.

R. H. Hoyle, Structural equation modeling: Concepts, issues, and applications. Sage, 1995.
Algorithm 1: Source of Funds Tracing Algorithm

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

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

1. $b_{\text{first}} \leftarrow$ Binary search between block 0 and
2. $b_{\text{current}}$ where $T$'s nonce equals 0 in $b_{\text{first}}$, and $T$'s nonce greater than 0 in $b_{\text{first}} + 1$.
3. $b_{\text{funding}} \leftarrow$ Binary search between block 0 and
4. $b_{\text{first}}$ where $T$'s balance is greater than 0 in $b_{\text{funding}}$ and $T$'s balance equals 0 in $b_{\text{funding}} - 1$.

5. foreach $tx \in \{t_{\text{funding}}^0, \ldots\}$ do
   6. if Replay $tx$ and finds native token transfer to $T$ then
      7. return $tx$
   8. end
9. end

Suspects ($A^*$) | Pattern | Incident | Date |
--- | --- | --- | --- |
$A^* \rightarrow A^*$ | WildCard | May 27, 2021 |
$A^* \rightarrow A^*$ | DeFiBox | Oct 08, 2020 |
$A^* \rightarrow A^*$ | DODO | Mar 08, 2021 |
$A^* \rightarrow A^*$ | VisorFinance | Nov 26, 2021 |
$A^* \rightarrow A^*$ | MakerDAO | Dec 12, 2020 |
$A^* \rightarrow A^*$ | BuccanerFi | Mar 29, 2020 |
$A^* \rightarrow A^*$ | InfinityToken | Jan 26, 2022 |
$A^* \rightarrow A^*$ | SodaFinance | Sep 20, 2020 |
$A^* \rightarrow A^*$ | BoozeFi | Aug 24, 2020 |
$A^* \rightarrow A^*$ | ForceDAO | Apr 04, 2021 |
$A^* \rightarrow A^*$ | PancakeBunny | Jun 03, 2021 |
$A^* \rightarrow A^*$ | BobbleFinance | May 22, 2021 |
$A^* \rightarrow A^*$ | MakerDAO | Dec 12, 2020 |
$A^* \rightarrow A^*$ | RadixDAO | 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.