# Phishing Tactics Are Evolving: An Empirical Study of Phishing Contracts on Ethereum

BOWEN HE*, Zhejiang University, China and Mohamed bin Zayed University of Artificial Intelligence, United Arab Emirates
XIAOHUI HU, Huazhong University of Science and Technology, China
YUFENG HU, Zhejiang University, China and BlockSec, China
TING YU, Mohamed bin Zayed University of Artificial Intelligence, United Arab Emirates
RUI CHANG, Zhejiang University, China
LEI WU, Zhejiang University, China and BlockSec, China
YAJIN ZHOU†, Zhejiang University, China and BlockSec, China

The prosperity of Ethereum has led to a rise in phishing scams. Initially, scammers lured users into transferring or granting tokens to Externally Owned Accounts (EOAs). Now, they have shifted to deploying phishing contracts to deceive users. Specifically, scammers trick victims into either directly transferring tokens to phishing contracts or granting these contracts control over their tokens. Our research reveals that phishing contracts have resulted in significant financial losses for users. While several studies have explored cybercrime on Ethereum, to the best of our knowledge, the understanding of phishing contracts is still limited.

In this paper, we present the first empirical study of phishing contracts on Ethereum. We first build a sample dataset including 790 reported phishing contracts, based on which we uncover the key features of phishing contracts. Then, we propose to collect phishing contracts by identifying suspicious functions from the bytecode and simulating transactions. With this method, we have built the first large-scale phishing contract dataset on Ethereum, comprising 37,654 phishing contracts deployed between December 29, 2022 and January 1, 2025. Based on the above dataset, we collect phishing transactions and then conduct the measurement from the perspectives of victim accounts, phishing contracts, and deployer accounts. Alarmingly, these phishing contracts have launched 211,319 phishing transactions, leading to $190.7 million in losses for 171,984 victim accounts. Moreover, we identify a large-scale phishing group deploying 85.7% of all phishing contracts, and it remains active at present. Our work aims to serve as a valuable reference in combating phishing contracts and protecting users' assets.

CCS Concepts: • **Security and privacy → Web application security**.

Additional Key Words and Phrases: Decentralized Finance, Ethereum, phishing contract detection

*Part of this work was done when the author was a research intern at BlockSec.
†Corresponding author: Yajin Zhou (yajin_zhou@zju.edu.cn).

Authors' Contact Information: Bowen He, bowen_os@zju.edu.cn, Zhejiang University, Hangzhou, Zhejiang, China and Mohamed bin Zayed University of Artificial Intelligence, Abu Dhabi, United Arab Emirates; Xiaohui Hu, xiaohui_hu@hust.edu.cn, Huazhong University of Science and Technology, Wuhan, Hubei, China; Yufeng Hu, yufenghu@zju.edu.cn, Zhejiang University, Hangzhou, Zhejiang, China and BlockSec, Hangzhou, Zhejiang, China; Ting Yu, ting.yu@mbzuai.ac.ae, Mohamed bin Zayed University of Artificial Intelligence, Abu Dhabi, United Arab Emirates; Rui Chang, crix1021@zju.edu.cn, Zhejiang University, Hangzhou, Zhejiang, China; Lei Wu, lei_wu@zju.edu.cn, Zhejiang University, Hangzhou, Zhejiang, China and BlockSec, Hangzhou, Zhejiang, China; Yajin Zhou, yajin_zhou@zju.edu.cn, Zhejiang University, Hangzhou, Zhejiang, China and BlockSec, Hangzhou, Zhejiang, China.


ACM 2476-1249/2025/6-ART32
https://doi.org/10.1145/3727138

**ACM Reference Format:**
Bowen He, Xiaohui Hu, Yufeng Hu, Ting Yu, Rui Chang, Lei Wu, and Yajin Zhou. 2025. Phishing Tactics Are Evolving: An Empirical Study of Phishing Contracts on Ethereum. *Proc. ACM Meas. Anal. Comput. Syst.* 9, 2, Article 32 (June 2025), 24 pages. https://doi.org/10.1145/3727138

## 1 Introduction

Since the emergence of Decentralized Finance (DeFi), Ethereum has consistently attracted capital and substantial investments from users. By December 2024, the Total Value Locked (TVL) on Ethereum had surpassed $76.8 billion, accounting for 57.0% of the overall TVL in the DeFi sector [12]. Unfortunately, alongside the influx of capital, phishing incidents have become increasingly prevalent and have caused over three billion dollar losses in the cryptocurrency ecosystem [1–7, 58]. Phishing attacks are already regarded as the most severe threat, prompting both academia and industry to propose various strategies to counteract them [18, 19, 58, 59].

However, phishing tactics are continuously evolving. *In particular, to make phishing attacks more stealthy, attackers are increasingly moving from Externally Owned Accounts (EOAs) to Contract Accounts (CAs).* In previously studied phishing processes, users are deceived into signing a transaction that authorizes or transfers funds directly to an EOA address [59]. Since legitimate projects typically deploy an official contract address for user interactions, users can easily recognize a phishing scam when they are asked to send or approve tokens to an EOA address. To deceive users, now scammers use smart contracts instead of EOAs in phishing attacks. As can be seen in Fig. 1, users are induced by scammers to visit fraudulent websites and sign transactions to invoke phishing contracts (instead of transferring tokens directly to an EOA). After signing these transactions, users transfer tokens to phishing contracts and receive nothing. According to our measurement results, scammers have stolen over $190 million from users with phishing contracts, highlighting the urgent need for effective detections and comprehensive analysis of phishing contracts to raise community awareness.

Although several studies have proposed to detect phishing scams through transaction analysis [55, 79, 81], a more effective approach is to identify phishing contracts at the time of deployment so that users are protected before phishing transactions occur. To do so, it requires analysis of the bytecode of phishing contracts, which imposes several challenges for our study. **Challenge#1** arises intuitively from the lack of a dedicated sample dataset. Specifically, the datasets of prior works are phishing transactions. By contrast, our sample dataset should consist of phishing contracts and their bytecode that characterize their key features for detection. Such dedicated datasets are not available in the literature [55, 57, 59, 66, 79, 81]. **Challenge#2** occurs during the collection process of sample phishing contracts because there are no dedicated labels or definitions for phishing contracts. Specifically, existing account labels typically categorize contracts as either scams or non-scams, without distinguishing specific scam types such as phishing, Ponzi schemes, or money laundering. Therefore, we must manually extract phishing contracts from the reported and labeled scam contracts. **Challenge#3** stems from the closed-source nature of phishing contracts. Without access to source code, detection must rely entirely on bytecode analysis, which provides significantly less information. This limitation not only complicates the identification of phishing contracts but also makes automated detection across a large volume of contracts considerably difficult.

**Our work.** In this paper, we present the first systematic study of phishing contracts on Ethereum. To address Challenge#1, we propose to build a sample dataset of phishing contracts from reported attacks and publicly acknowledged labels. To address Challenge#2, we manually extract the phishing contracts based on the report details, associated transactions, and reversed source code. As a result, we collect **790** phishing contracts as the sample dataset and analyze their phishing functions, highlighting their key features for detection (§4). To address Challenge#3, for a smart contract, we

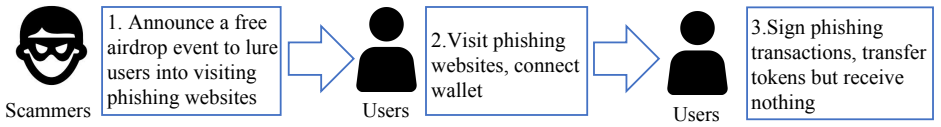



Fig. 1. A Scam Scenario Involving a Phishing Contract

first extract suspicious functions from its bytecode, then simulate the transactions, finally judge whether it is a phishing contract based on the simulation result (§5). In this way, we have compiled a comprehensive dataset, including **37,654** phishing contracts deployed between December 29, 2022 and January 1, 2025.

However, relying solely on contract data is insufficient for comprehensive measurement, as it requires the collection of related phishing transactions. To this end, for each transaction of the phishing contracts, we examine the invoked function selector to determine whether it is a phishing transaction (§6.1). For confirmed phishing transactions, we analyze the fund flow, identify victim accounts, and calculate the value of stolen tokens. As a result, we have identified **211,319** phishing transactions occurring between December 29, 2022 and January 8, 2025. Through these transactions, scammers earned **$190.7 million** from **171,984** victim accounts.

Next, based on the dataset, we perform a comprehensive analysis of phishing contracts from the perspectives of victim accounts (§6.2), phishing contracts (§6.3), and contract deployer accounts (§6.4). Specifically, **89.9%** of victim account losses are below $1,000. Many user accounts fail to revoke permissions or repeatedly sign phishing transactions, causing them to fall for phishing schemes multiple times. **96.2%** of phishing contracts has lifecycles shorter than one day. When analyzing these short-lifecycle phishing contracts, we find that scammers deploy phishing contracts daily or even use a single contract for one phishing attack to evade contract labeling security mechanisms. **85.6%** of phishing contract deployment funds can be traced to specific entities, such as CEXs, mixers, cross-chain bridges, and victim accounts. Notably, **63.2%** of them originate from victim accounts. What's more, we identify a large-scale phishing group that has deployed **85.7%** of all phishing contracts, which remains active at the time of writing. To safeguard users, we have reported our findings to Etherscan and the relevant security teams.

**Contributions.** We summarize our contributions as follows.

- **Anatomy of phishing contracts on Ethereum.** We manually build a sample dataset of phishing contracts on Ethereum and analyze the critical features of their phishing functions. This analysis serves as a foundation for the detection of phishing contracts.
- **First large-scale dataset.** We develop an automatic phishing contract detection system and build the first large-scale dataset of phishing contracts on Ethereum, which is 47 times larger than the sample dataset. We make efforts to ensure the accuracy and integrity of the dataset and share it with the research community. [1]
- **First in-depth study of phishing contracts.** We collect phishing transactions and conduct the measurement study from the perspectives of victim accounts, phishing contracts, and deployer accounts. We analyze scammers' strategies for evading contract labeling mechanisms and discover

[1] https://github.com/blocksecteam/phishing_contract_sigmetrics25

a large-scale phishing group. Our study can serve as a valuable reference for Ethereum service providers to combat phishing contracts.

## 2 Background

### 2.1 Ethereum

Ethereum is a decentralized blockchain platform empowering developers to deploy decentralized applications. Ethereum accounts are categorized into Externally Owned Accounts (EOAs) and Contract Accounts (CAs).

**EOAs.** EOAs are managed through private keys, allowing them to initiate transactions, send ETH, transfer tokens, and interact with smart contracts. Unlike smart contracts, they do not contain any on-chain code. Additionally, EOAs must pay gas fees in ETH when executing transactions.

**CAs.** CAs operate based on bytecode stored on the blockchain. They cannot send transactions on their own and can only execute functions when called by an EOA or another contract.

**Transactions.** In Ethereum, transactions serve as signed messages utilized for token transfers, invoking functions within smart contracts, or deploying smart contracts.

**Tokens.** Ethereum utilizes Ether (ETH) as its native token to facilitate transactions and cover computational costs. Additionally, it allows developers to create and distribute tokens via smart contracts. The smart contract code records the relationship between accounts and tokens. These tokens can be categorized into fungible and non-fungible tokens (NFTs). Each fungible token is identical to others, whereas each NFT is unique. Most of the fungible token contracts adhere to the ERC-20 interface [14], like USDT and USDC, commonly referred to as ERC20 tokens.

### 2.2 Invoking Smart Contracts on Ethereum

Developers typically write smart contracts using high-level languages such as Solidity [20] or Vyper [25], which are then compiled into Ethereum Virtual Machine (EVM) bytecode. To deploy a contract, developers launch a transaction containing the bytecode and any necessary constructor arguments. Besides, deploying a contract requires a gas fee, which is directly influenced by the length of the bytecode. Once a contract is deployed on Ethereum, only its bytecode is publicly available. The source code is accessible to others only if the deployer actively uploads it.

**Wallet user interface.** Users commonly use digital wallets to manage the private keys of their accounts and launch transactions. When interacting with a Decentralized Application, users first visit the associated front-end websites. Following this, they connect wallets and sign transactions to invoke the official contracts. Before users sign transactions, the wallet analyzes the invoked function name in the transaction window. Scammers leverage this to deceive users into signing phishing transactions.

**Invoking process.** For a transaction invoking a smart contract, its calldata consists of a function selector and arguments. The function selector is the first four bytes of the Keccak-256 hash of the function signature. And the arguments are encoded in the formats of Contract ABI Specification [10]. The compiled bytecode of the contract initiates at a dispatcher. Specifically, the dispatcher extracts the function selector from the calldata and directs the execution flow to the corresponding function if the selector matches any of the public functions in the contract. In the absence of matched functions, the dispatcher redirects to the fallback function if defined. If neither matched functions nor a fallback function is found, the transaction will revert. The gas fee for invoking a smart contract is determined by the opcodes executed during the transaction. Each executed opcode contributes to an increase in the overall gas fee.

**Payable functions.** In Ethereum, a payable function is a unique type of function within a smart contract designed to accept ETH during its execution. When a function is declared as payable, it can receive Ether from external accounts or other contracts. Conversely, if a function is not declared as payable, it cannot accept any Ether sent to it.

### 2.3 Transaction-based Phishing on Ethereum

In this paper, we focus on transaction-based phishing. Specifically, scammers lure victims to interact with phishing websites, prompting them to sign messages or transactions that allow scammers to withdraw their tokens to phishing accounts. As can be seen from Figure 1, a successful transaction-based phishing operation typically consists of three stages.

(1) **Scammers promote phishing websites.** Scammers typically spread fraudulent websites through social platforms such as Twitter, Telegram, Instagram, or Discord. They may also embed malicious advertisements directing users to phishing websites on search engines like Google. To maximize their reach, scammers often impersonate reputable projects or directly compromise their official accounts, leveraging their established reputation for promotion.
(2) **Victims visit phishing websites and connect wallets.** Enticed by words like "AIRDROP", victims often visit phishing websites without much deliberation. Many of these fraudulent websites are created using website cloning tools, making them nearly indistinguishable from legitimate ones. Consequently, victims are quick to connect their wallets.
(3) **Victims sign phishing transaction.** Once the victim's wallet address is captured, the phishing website's frontend program scans for valuable tokens. It then generates a phishing transaction or message designed to steal the victim's most valuable assets. Due to the website's convincing disguise, the victim fails to recognize its fraudulent nature and unknowingly signs the phishing transaction or message. Consequently, the victim's tokens are transferred to the scammer's accounts.

While He *et al.* [59] presents a method for identifying transaction-based phishing scams on Ethereum, it's limited to cases where phishing accounts are Externally Owned Accounts (EOAs). In contrast, our research focuses on cases where phishing accounts are contracts, which present greater sophistication and detection challenges.

## 3 Study Design

In this work, we seek to address the following research questions (RQs) to gain a comprehensive understanding of phishing contracts on Ethereum:

RQ1: **What are the common features of phishing contracts?** Despite the significant cryptocurrency losses caused by phishing contracts, the community still lacks an in-depth understanding of this emerging phishing tactic, due to insufficient insights into the operation and characteristics. Furthermore, even as exploits caused by phishing contracts continue to rise, the community has yet to explicitly label such contracts. Instead, they are often broadly categorized under the generic label of "scam". As a result, to thoroughly expose this attack pattern, we must first accurately label phishing contracts from existing reports, and uncover their common characteristics.

RQ2: **Can we build a large-scale dataset of phishing contracts with only bytecode accessible?** The reported phishing contracts are merely the tip of the iceberg, while unmarked phishing contracts will continue to exacerbate threats to the ecosystem. Even more concerning is that if phishing contracts cannot be identified before they steal assets, more of them will be deployed in the future, causing further financial losses. Besides, a large-scale dataset is essential for conducting effective measurement and analysis of phishing contract attacks.

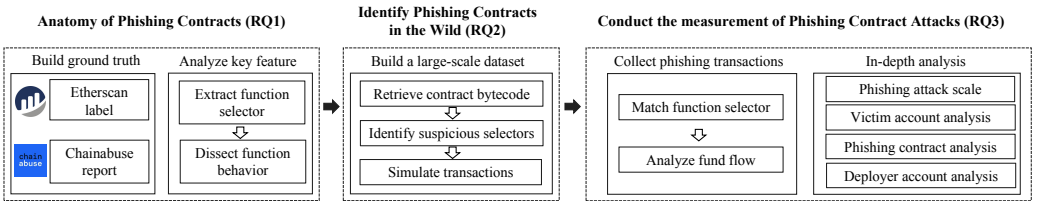



Fig. 2. An Overview of Our Work

Furthermore, many scammers deliberately avoid uploading the source code of phishing contracts to obscure their malicious activities. Consequently, it becomes imperative to develop methods for identifying phishing contracts directly from their bytecode.

RQ3: **What are the harmful effects and implications introduced by phishing contracts?** Leveraging the large-scale dataset, we aim to measure the impacts of phishing contracts across multiple dimensions, pertaining to attack transaction scale, stolen funds, as well as victims and attackers. Therefore, it is necessary to investigate: *RQ3.1) What is the volume of attack transactions, and how many victims fall prey to phishing contract attacks?* Answering these questions will provide insight into the overall scale of this emerging phishing strategy. *RQ3.2) What is the distribution of victim losses, and what are the on-chain behaviors of victims before signing phishing transactions?* This will help highlight the losses for each victim account and identify the types of users most vulnerable to these scams. *RQ3.3) How much profit does each phishing contract generate from these attacks? How long is each one active?* Analyzing the profit and lifecycle of each contract offers a clear and intuitive insight into the profit distribution and behavioral patterns of phishing contracts. And *RQ3.4) How many accounts deploy these phishing contracts? What are the sources of deployment funds? Are there any connections among the primary phishing contract deployers?* Figuring out the key characteristics of deployer accounts can help us discover the primary phishing groups that are consistently deploying different phishing contracts.

Fig. 2 shows the workflow of our study. We begin by constructing a ground truth dataset from the community reports, comprising 790 phishing contracts. Using this dataset, we analyze the key features of phishing functions within these contracts. Then, based on these features, we develop a system to identify phishing contracts that *have been deployed but remain unlabeled* in the wild. Our tool successfully identifies 37,654 phishing contracts, which have been manually verified and reported to Etherscan and the relevant security teams. Finally, we gather phishing transactions and deployment transactions associated with these phishing contracts. Utilizing these datasets, we perform an in-depth analysis of phishing contract attacks, focusing on victim accounts, phishing contracts, and deployer accounts.

## 4 Building Sample Dataset of Phishing Contracts

In this section, we build a sample dataset of phishing contracts, which is crucial for gaining an initial understanding of phishing contracts and defining the decision boundaries for their identification. **Solution to Challenge#1:** We first collect exposed scam contracts and transactions from reported attacks and publicly acknowledged labels. **Solution to Challenge#2:** From the exposed scam contracts, we manually extract the phishing contracts according to the report details, associated transactions, and reversed source code. Based on the extracted sample dataset, we characterize them and provide clear definitions for identifying a phishing contract. These efforts can serve as a foundation for collecting phishing contracts on a large scale.

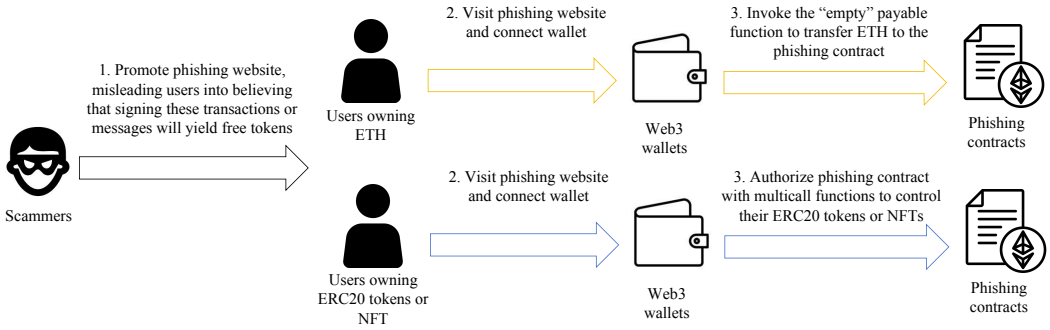



Fig. 3. The Scam Process Involving Phishing Contracts. For victims possessing ETH, they will be lured to invoke the payable phishing function, like *Claim* in Listing 1. For victims owning ERC20 tokens or NFTs, they will be induced to grant the scammer control over their tokens, via off-chain signatures or transactions. Then the *multicall* function in the phishing contract will be invoked to transfer victims' tokens.

Table 1. Analysis of Phishing Functions and Contracts in the Sample Dataset.

| Function Type | "Empty" payable function | Multicall function |
|---|---|---|
| Steal Targets | ETH | ERC20 tokens & NFTs |
| Suspicious Features | Enticing Name | Only specific phishing accounts can invoke |
| | Short Length | |
| | Lacking necessary on-chain operations | |
| Function Number | 626 | 524 |

### 4.1 Collecting Exposed Scam Contracts & Transactions

In this section, we collect sample phishing contracts from reported attacks on Chainabuse [9], a prominent platform for reporting malicious crypto activities, and account labels on Etherscan [16], the leading Ethereum blockchain explorer. Since users often use the generic label "scam" to report some malicious contracts on Ethereum, including fake tokens, phishing, and other types of scams, there lack specific labels for phishing contracts. Hence, we first collect scam contracts and then extract phishing contracts from them. We carried out the dataset collection of scam contracts between December 5, 2024, and December 8, 2024. In statistics, there are 13,327 labeled and reported scam contracts deployed after January 1, 2023. From this dataset, we identify phishing contracts based on the report details, transactions.

### 4.2 Extracting Phishing Contracts From Exposed Scam Contracts

Although we have collected numerous scam contracts in the previous section, many of them share identical codes. In fact, there are 4,521 distinct scam contracts among them. Therefore, we only need to verify these distinct scam contracts individually. For each scam contract, we examine the report details to check whether it mentions phishing contract attacks. Additionally, we review the transaction details to identify any phishing transactions. Specifically, if the contract is invoked to transfer victims' tokens without returning anything, we classify it as a phishing contract. Among all scam contracts in the dataset, there are 9,818 fake token contracts, 2,608 contracts that disseminate fake tokens, and 790 phishing contracts. The remaining 111 contracts engage in activities such as money laundering and honeypot, distinct from phishing. Hence, the 790 phishing contracts are regarded as our initial ground truth and form the sample dataset.

```
contract ClaimRewards {
    address private phishing_account;
    constructor() public{  phishing_account=0x3eb2xxxxc218;  }
    function Claim() public payable {} // function to steal ETH.
    struct CallData {
        address contract;
        bytes Bytes;
    }
    function multicall(CallData[] calls) public {
        require(phishing_account == msg.sender);
        // ensure that only another phishing account can invoke.
        for(uint i = 0; i < calls.length; i++) {
            calls[i].contract.call(calls[i].Bytes);
        }
    } // function to steal ERC20 tokens & NFTs.
}
```

Listing 1. Example of a Phishing Contract. We simplify its source code to highlight the key components. The *Claim* function is designed to steal ETH. The *multicall* function is implemented to steal ERC20 tokens & NFTs.

### 4.3 Key features: *"empty" payable* functions and *multicall* functions

For each phishing contract in the sample dataset, we collect their phishing functions that are invoked to steal tokens from victims' accounts. To this end, we first analyze their phishing transactions and extract the corresponding phishing functions. Additionally, we analyze its source code or decompile the bytecode using Dedaub [11] to explore the detailed implementation of these phishing functions. In total, we collect 1,150 phishing functions. Next, we classify these phishing functions based on their implementation methods and the types of assets they target. Our analysis reveals that phishing functions typically fall into two main categories: an *"empty" payable function* to steal ETH and a *multicall* function to steal ERC20 tokens & NFTs. Furthermore, Table 1 highlights suspicious characteristics that distinguish phishing functions from those found in legitimate contracts. The scam process involving phishing contracts is depicted in Fig. 3.

**"Empty" payable functions to steal ETH.** For victims possessing ETH, scammers design enticing "empty" payable functions to pilfer ETH. When a contract function is declared payable, users can transfer ETH to the contract while invoking that function. If the function is not payable and users attempt to transfer ETH during invocation, the transaction will revert. Consider the function *Claim* in Listing 1 as an example; despite its name suggesting a reward claim, this function lacks any implementation. For some phishing contracts, this function may include one meaningless operation, like updating a local variable. However, from the user's perspective, invoking this function merely withdraws users' ETH without returning anything. So we refer to this category of phishing function as an "empty" payable function.

To gain a more thorough insight into these "empty" payable functions, we collect 50 legitimate payable functions from official contracts for comparison. These functions also have enticing names like claim and mint. Invoking these official functions requires users to transfer ETH to the contracts. However, upon invocation, they return tokens to users, accompanied by the generation of corresponding on-chain event logs. Since the operations to return tokens or generate on-chain event logs incur significant gas fees, designing phishing functions to be either 'empty' or comprised of a single trivial operation can minimize costs. As a result, *they neither return tokens to users nor generate any on-chain event logs*. This characteristic can be utilized for detection purposes.

**Multicall function to steal ERC20 tokens & NFTs.** Phishing transactions stealing ERC20 tokens & NFTs can be complex. For instance, phishing websites can trick victims into granting scammers control over their tokens by initiating an approval transaction. Then scammers launch transactions to transfer their tokens. In NFT phishing scenarios, phishing websites may ask victims to sign a zero-dollar purchase order [26]. With this signed order, scammers can interact with NFT marketplace contracts to transfer victims' NFTs at no expense. In summary, different phishing schemes require different types of phishing transactions. As a result, they leverage *multicall* function, as demonstrated in Listing 1, to launch specific phishing transactions according to the phishing schemes. Typically, the *multicall* function takes an array of encoded function calls as input. Each item in the array contains a target contract address, function signature, and parameters. This allows it to batch-execute multiple contract calls within a single transaction based on provided parameters.

In Listing 1, the *multicall* function is similar to that in official contracts, such as Uniswap V3: multicall 2 except caller check [22, 23]. For comparison, we collect 20 official *multicall* contracts from well-known legitimate projects and manually inspect their source codes. We discover that in official contracts, the *multicall* function can be invoked by any account and serves as an auxiliary function for batch-invoking multiple contracts. In contrast, *the multicall function in phishing contracts can only be invoked by certain phishing accounts*. This check is necessary because without caller verification, after victims authorize their tokens to the *multicall* phishing contract via approval transactions, anyone can invoke it to transfer the victim's tokens. This feature can be leveraged for detection.

In summary, the sample dataset contains 1,150 phishing functions. 54.4% of them (626) are "empty" payable functions and 45.6% of them (524) are multicall functions, as can be seen from Table 1.

**Answer to RQ1:** Phishing contracts generally include two types of phishing functions. The first type is "empty" payable functions with enticing names, used to steal ETH. The second type is multicall functions, which can only be invoked by specific phishing accounts, designed to steal ERC20 tokens and NFTs. These characteristics can serve as a foundation for the following large-scale collection of phishing contracts in the wild.

## 5 Collecting Phishing Contracts in the Wild

In Section 1, we highlight that the primary challenge in identifying phishing contracts stems from the reliance on bytecode. Based on the two patterns *i.e.*, *"empty" payable* functions stealing ETH and *multicall* functions stealing ERC20 tokens & NFTs, summarized in Section 4, we strive to identify all in-the-wild phishing contracts.

**Solution to Challenge#3:** Algorithm 1 describes the procedure to identify phishing contracts. For phishing targeting ETH, we begin by detecting *payable functions* with suspicious names from the bytecode. Next, we simulate transactions to determine whether any on-chain events are triggered. If no on-chain events are observed, the contract is classified as a phishing contract. For phishing schemes targeting ERC20 tokens and NFTs, we detect *multicall* functions by analyzing their calldata loading operations. Through transaction simulation, we evaluate whether these functions can only be executed by specific phishing accounts. If this condition is met, the contract is also categorized as a phishing contract.

To automate this process and scale it to a large set of contracts, our detection system operates as follows: (1) collecting all on-chain contract data; (2) identifying suspicious functions within the bytecode; (3) simulating transactions to invoke the identified functions; and (4) determining phishing contracts based on the simulation results. Finally, we validate efficiency of our tool and the accuracy of the detection results.

**Algorithm 1** Phishing Contract Detection Algorithm

**Require:** Contract bytecode $\mathcal{B}$
**Ensure:** Contract type $\mathcal{T}$
Initialize $empty\_payable\_phishing \leftarrow$ **False** ▷ Flag for empty payable phishing function
Initialize $multicall\_phishing \leftarrow$ **False** ▷ Flag for multicall phishing function
Initialize $\mathcal{T} \leftarrow legitimate\_contract$ ▷ Default contract type
**for** each function $f$ in $\mathcal{B}$ **do**
    **if** $f$ is a suspicious payable function **and** invoking $f$ generates no on-chain events **then**
        $empty\_payable\_phishing \leftarrow$ **True** ▷ Mark as empty payable phishing
    **end if**
    **if** $f$ is a multicall function **and** $f$ is invoked only by phishing accounts **then**
        $multicall\_phishing \leftarrow$ **True** ▷ Mark as multicall phishing
    **end if**
**end for**
**if** $empty\_payable\_phishing \wedge \neg multicall\_phishing$ **then**
    $\mathcal{T} \leftarrow empty\_payable\_phishing\_contract$
**else if** $\neg empty\_payable\_phishing \wedge multicall\_phishing$ **then**
    $\mathcal{T} \leftarrow multicall\_phishing\_contract$
**else if** $empty\_payable\_phishing \wedge multicall\_phishing$ **then**
    $\mathcal{T} \leftarrow empty\_payable\&\&multicall\_phishing\_contract$
**end if**
**return** $\mathcal{T}$

### 5.1 Gathering Ethereum Contracts

We have deployed a local Ethereum node [21] to gather all contracts deployed between December 29, 2022 and January 1, 2025. During this period, we collect a total of 13,215,401 contracts. For each contract, we record its bytecode, deployer account, and deploy timestamp.

### 5.2 Identifying Suspicious Functions

In this part, we aim to analyze contract bytecode, then identify suspicious payable functions and *multicall* functions, which could potentially become phishing functions.

**Identify suspicious payable functions via selectors.** In Section 2.2, we mention that the function selector is the first four bytes of the Keccak-256 hash of the function signature. *The dispatcher within the bytecode matches the function selector with its corresponding function and redirects to the fallback functions if no match is found. In the absence of matched functions or a fallback function, the transaction will revert.* When invoking a smart contract, the digital wallet will extract the function selector from the calldata. It then checks the Ethereum Signature Database [15], a public database that maps function selectors to their corresponding method names. If the method name is located, it will display method names on the confirmation screen [13]. If the method name is not found, the wallet will display the hexadecimal function selector instead. *To lure users with enticing method names, scammers must initially register the method names in the database.*

To determine whether a function is payable on the bytecode level, we investigate the implementation of compilers [20, 25]. We discover that, for a non-payable function, the compiler will add a callvalue check in the bytecode after matching the selector. Specifically, it retrieves the callvalue via the `CALLVALUE` opcode. If the value is non-zero, the transaction will revert.

To identify suspicious payable functions, we first *extract the function selectors from the dispatcher in the bytecode*. To judge if a function is payable, we *inspect whether a call value check exists in its implementation*. Then we search Ethereum Signature Database to determine whether the method names are registered. As discussed in Section 4.3, scammers must register the method name, allowing users to see it when invoking contract functions. To lure users into signing transactions, scammers often use deceptive keywords such as "claim" or "reward". Hence, if *the selector is registered in the Ethereum Signature Database and method name contains or is similar to a suspicious keyword*, we label it as suspicious. Additionally, we employ edit distance to represent the similarity ratio using the Levenshtein.ratio function, considering any ratio above 0.8 as indicative of similarity. From the sample dataset collected in Section 2, we identify 54 suspicious payable function selectors. Subsequently, we extract 12 suspicious keywords from these sources for detection. Additionally, we incorporate 23 suspicious keywords identified in existing studies [58, 59, 80]. In total, we identify 35 suspicious keywords for detection, which will be included in our open-source dataset. For simplicity, we directly identify 104,315 registered functions from the Ethereum Signature Database whose names contain or are similar to these suspicious keywords. If payable functions with these names are found in the bytecode, we extract them and proceed to the next step.

**Identify multicall functions via bytecode operations.** Currently, for *multicall* functions, there are two types of parameters: (address,bytes)[] and (address[],bytes[]). Inspired by SigRec [56], we can deduce the function parameter types via the EVM bytecode operations to load calldata. For example, if the parameter type is (address,bytes)[], the EVM bytecode will first load the array length using `CALLDATALOAD`, then load the address with `CALLDATALOAD`, and finally load the bytes with `CALLDATACOPY`. Thus, by recording the sequences and parameters of EVM bytecode operations used to load calldata, we can determine the function's parameter types. For an unknown function, we invoke it with parameters in the form of (address,bytes)[] or (address[],bytes[]). If we observe operations loading addresses and bytes accordingly, we classify the function as a *multicall* function.

### 5.3 Simulating Transactions

After identifying suspicious functions, we will generate transaction parameters, according to the function selectors identified in Section 5.2, automatically. For suspicious payable functions, we randomly set the value of the transaction to send ETH. For *multicall* functions, we envision a phishing scenario. Here, we first approve our USDT tokens to the contract, then invoke the *multicall* function to transfer the tokens. Then we simulate the transactions with a local geth node deployed by us. Since the simulated transactions won't be recorded on-chain, there is no risk of token loss. We will record the transaction results, including account balance changes, emitted events, and internal transactions. Finally we will analyze these results to detect phishing contracts.

**Alter transaction parameters based on revert information.** Since scammers cannot predict victims' account addresses when stealing ETH, they do not verify a specific victim's account in these phishing attacks. Therefore, we can trigger the ETH phishing function using a random address. Yet, for the *multicall* phishing function, it will revert if the caller check fails to pass. In the EVM bytecode, it employs the "EQ" and "SUB" operations to compare the caller's address with the phishing account address. For this case, we utilize the "debug_traceCall" API offered by the geth node to obtain operations and their parameters. Specifically, we can retrieve the account address from the parameters of the "EQ" operation. Then we will modify the transaction parameters to pass the caller check.

### 5.4 Determining Phishing Contracts

In this part, we detect phishing contracts with the simulation results. Specifically, we use suspicious key features outlined in Table 1 for detection. For suspicious payable functions, if the transaction just receives caller's ETH without any return or other on-chain events, we identify them as phishing functions. For *multicall* functions, if only accounts flagged as phishing by Etherscan can invoke them and transfer victims' tokens, we categorize them as phishing functions.

### 5.5 Evaluating the Detection System

**Small-scale Verification.** To ensure the coverage and effectiveness of our detection system, we build a dataset consisting of phishing contracts and legitimate contracts for evaluation. The phishing contract dataset is the sample dataset collected in Section 4. For the benign contract dataset, we collect 400 top token contracts and 400 Decentralized Exchanges (DEXs) official contracts from Etherscan. Our system successfully detects all phishing contracts in the sample dataset without mistakenly identifying official contracts as phishing. In other words, our system produces neither false positives nor false negatives in the small-scale verification experiment.

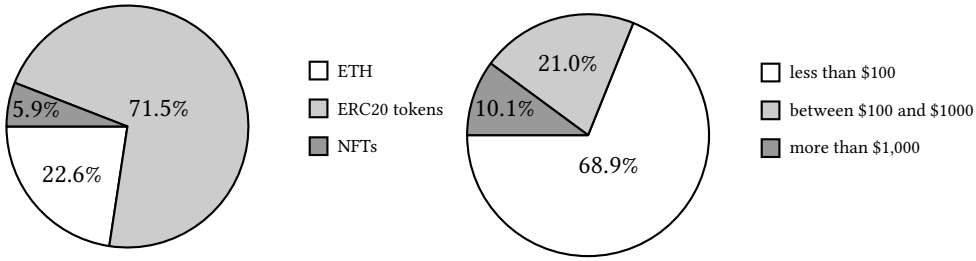
**Large-scale Dataset Collection.** Between December 29, 2022 and January 1, 2025, we identified 37,654 phishing contracts in the wild, with 1,450 unique contract hashes. In total, we identified 4,601 suspicious payable functions that steal ETH. Among the 35 suspicious keywords, the top ten most frequently appearing keywords are: claim, securityupdate, connect, swap, mint, confirm, verify, withdraw, gift, and airdrop. The frequency distribution of their occurrences is as follows: 42.7%, 13.5%, 8.6%, 6.1%, 5.0%, 4.2%, 3.3%, 3.1%, 2.3%, 2.3%, respectively. These contracts constitute our large-scale dataset for the subsequent measurement analysis. To ensure the dataset's reliability, we implemented a thorough validation process. A team of three experienced security analysts, all with extensive expertise in phishing detection on Ethereum, was assembled to verify the detection results. For contracts sharing the same bytecode, we perform the check only once. Each unique phishing contract was randomly assigned to two analysts for manual validation, ensuring an independent and accurate review.

For each contract, we verified the presence of associated phishing transactions and confirmed whether related phishing functions were successfully identified by our system. For contracts without any phishing transactions, we manually reviewed their source code. If their source code were closed-source, we employed Dedaub [11] to reverse-engineer it. Next, we attempted to simulate phishing transactions. If the phishing transactions were successfully simulated, we classified the contract as a phishing contract. In summary, among the 1,450 unique contracts with distinct bytecodes, 895 have associated phishing transactions, while 555 lack transactions. Of the contracts without transactions, 199 are open-source smart contracts, and 356 are closed-source. Through this rigorous validation process, we did not identify any false positives generated by our system.

> **Answer to RQ2:** We can collect phishing contracts by identifying suspicious functions, simulating transactions, and analyzing simulation results. Phishing contracts are prevalent in the ecosystem. Specifically, we have built a large-scale dataset including 37,654 phishing contracts deployed between December 29, 2022 and January 1, 2025, which is 47 times larger than the sample dataset collected from the community.

## 6 Investigating Phishing Contracts

With the validated large-scale phishing contracts, we're looking to peek into this snake pit, figuring out the characteristics of such special contracts. However, relying solely on the contract data does not provide enough basis to delve into these topics; it is required to collect related phishing




(a) Distribution of Victim Losses by token types (b) Distribution of Victim Account Losses

Fig. 4. Distribution of Victim Account Losses and Lost Token Types

transactions. In this section, we first identify phishing transactions invoking the collected contracts. Finally, we conduct measurements from the perspectives of victims, phishing contracts, and contract deployment patterns.

### 6.1 Collecting Phishing Transactions

Not all transactions invoking a phishing contract are malicious phishing transactions, *e.g.*, scammers can invoke the contract to transfer stolen funds to another address or set ownership of phishing contracts. For instance, phishing contract 0xa9eb was invoked in transaction 0xaa4b to withdraw tokens to phishing account 0xa9eb [50]. The phishing contract 0x66dc was invoked in transaction 0x59c3, allowing phishing account 0x8d79 to invoke it [49]. Therefore, it is required to identify transactions used to execute phishing activity. As a follow-up to the previous analysis of contract bytecodes in Section 4.3, transactions calling only the "empty" payable functions and multicall functions should be considered phishing transactions, and we use function selectors to filter them.

**Calculating lost token values.** For each collected phishing transaction, we analyze fund flows, identify the victim account, and calculate the value of lost tokens. For ETH and ERC20 tokens, we retrieve token values using CoinGecko [52], the largest independent cryptocurrency data aggregator. For NFTs, we examine their sales history on marketplaces via Alchemy [8], selecting the price from the sale transaction closest in time to the phishing transaction as the token's value.

**Phishing Transaction Scale.** Between December 29, 2022 and January 8, 2025, we have collected 211,319 phishing transactions for phishing contracts detected in Section 5. There are 148,582 transactions stealing ETH, 57,630 transactions stealing ERC20 tokens, and 5,107 transactions stealing NFTs. These 37,654 phishing contracts earned $190.7 million from 171,984 victim accounts. During this period, over 200 phishing transactions occur daily, bringing huge losses for Web3 users. On January 23, 2024, we observed a loss of $3.8M from 1,199 phishing transactions, representing the highest loss amount in a single day within our dataset.

> **Answer to RQ3.1:** Phishing contracts have caused significant losses to users. Between December 29, 2022 and January 8, 2025, we have collected 148,582 phishing transactions, impacting 171,984 victims and resulting in total losses of $190.7 million.

### 6.2 Victim Account Analysis

In this section, we provide a comprehensive analysis of victim accounts targeted by phishing contracts. We analyze the total lost values for each type of tokens and the distribution of losses for each victim account. Besides, we explore the reasons why many users fall for phishing schemes

multiple times. Finally, we analyze victims' on-chain behavior before signing phishing transactions to identify their behavioral patterns.

**Distribution of victim losses across different tokens.** Fig. 4a shows the distribution of user losses across different token types. *ETH accounts for 22.6% of the total losses, ERC20 tokens dominate with 71.5%, and NFTs contribute 5.9%.* In total, we discover 3,285 types of lost ERC20 tokens, with the top three being stETH ($15.5 million), USDC ($9.6 million), and USDT ($5.3 million). Additionally, there are 1,425 types of lost NFTs, with the most valuable being BAYC #606, valued at $0.5 million. [2]

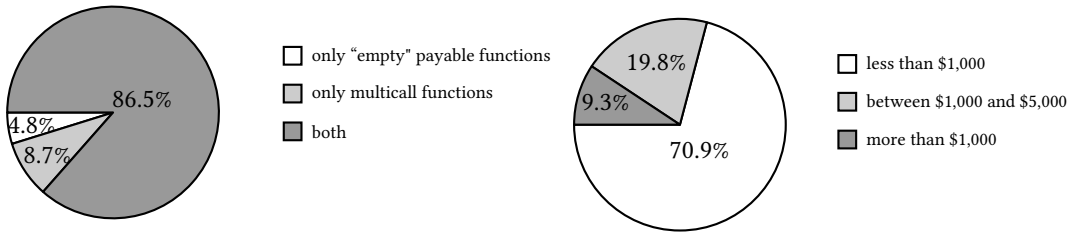
**Distribution of victim account losses.** We track the losses of each victim account. While the total losses from phishing contracts are significant, the majority of victim accounts have suffered relatively small individual losses. As shown in Fig. 4b, *68.9% of victim accounts experienced losses under $100.* This is due to the fact that most accounts hold only a small amount of tokens. Furthermore, *only 10.1% of victim accounts faced losses exceeding $1,000.* Also, there exist several cases where victim account losses are huge. For instance, on 2024-01-27, the account 0xc9f3 was defrauded of tokens worth $2.3 million [43]. The victim first authorized SuperVerse tokens in the account to a multicall phishing contract. The scammer then swiftly invoked the phishing contract to withdraw these tokens. The interval between the victim's approval transaction and the scammer's withdrawal transaction was merely 12 seconds, highlighting the scammer's efficiency. Additionally, we observed that the victim later signed five more phishing transactions, sending ETH and approving four other tokens to phishing contracts.

**Loss occurrences per victim.** When analyzing the frequency of losses for each victim, we find that *26,226 victim accounts have experienced losses more than once.* Among them, *10.1% accounts failed to revoke permissions in a timely manner.* For instance, the victim account `0x285c` granted approval for its Prime tokens to a multicall phishing contract on November 20, 2023 [51], allowing the tokens to be quickly withdrawn. Despite this, the victim did not revoke the approval. On November 24, 2023, they acquired new Prime tokens via Uniswap, and them these tokens were withdrawn again. Besides, *94.6% of these victims signed multiple phishing transactions* and lost over once, similar to the example mentioned above.

**On-chain behavior of victims before phishing.** To analyze the on-chain behavior of victim accounts before they signed phishing transactions, we collected data on their transaction counts over the six months preceding these incidents. Our analysis revealed that *75.6% of the accounts had fewer than 20 transactions during this period.* For example, the account `0x4a4e` received 0.02 ETH from Remitano, a well-known cryptocurrency exchange [53]. Subsequently, it invoked an "empty" payable function and transferred ETH to a phishing contract. Prior to this phishing transaction, the account had only a single transaction, which involved receiving tokens from the exchange. This suggests that the victim was likely an inexperienced Web3 user, making them more susceptible to scams. For comparison, we identified 372 Binance user accounts from transactions involving a well-known Binance deposit address between January 12, 2025, and January 14, 2025 [28]. We analyzed their transaction activity over the previous six months and found that, on average, these accounts conducted 252 transactions during that time. It highlights that most phishing victims had minimal on-chain activity, indicating a lack of experience in Web3 environments.

**Answer to RQ3.2:** 89.9% of victim losses are under $1,000. Many users fall for phishing schemes multiple times, due to failing to revoke permissions or signing multiple phishing transactions. Those with less Web3 experience are particularly vulnerable to phishing attacks.

[2] 0x773c5ea5fb02e6b735f7a05d407fd1335aa2b4bc46914676cbb2135ec1a6fda7




(a) Distribution of Phishing Contract Types (b) Distribution of Phishing Contract Profits

Fig. 5. Distribution of Phishing Contract Types and Profits

### 6.3 Phishing Contract Analysis

In this section, we analyze the profit and lifespan of each phishing contract to identify the most active phishing contracts and their behavioral patterns. Additionally, we reveal their approaches to evading contract labeling mechanisms. Furthermore, we find that they leverage the “delegatecall" function to minimize phishing contract deployment costs.

**Distribution of phishing contract types.** In Section 5, we note that phishing contracts can be categorized into three types: those with only “empty" payable functions, those with only multicall functions, and those with both. As shown in Fig. 5a, the majority of phishing contracts in our dataset have both types of phishing functions. Additionally, many phishing contracts share identical bytecode. For instance, the largest phishing contract group comprises 16,304 contracts that are identical to phishing contract `0x0001` [42]. All these contracts were deployed by a single phishing account [32]. Similarly, the second-largest group comprises 4,756 contracts identical to phishing contract `0x0005` [44], also deployed by a single phishing account [40]. In total, we identified seven phishing contract groups, each containing over 1,000 identical phishing contracts.

**Distribution of phishing contract profits.** We analyze the profits of phishing contracts, as illustrated in Fig. 5b. While the overall profit is significant, the majority of phishing contracts yield relatively modest earnings. Specifically, *70.9% of phishing contracts generate profits below $1,000, whereas only 9.3% earn over $5,000.* Notably, the contract `0x0000` emerges as the most profitable phishing contract in our dataset, amassing $3.9 million in earnings [31]. This contract executed 12,478 phishing transactions, targeting 11,689 victim accounts to steal ETH.

**Lifespan of Phishing Contracts.** To study the active timeline of phishing contracts, we collect data on their first and last phishing transactions. We define the interval between these transactions as their lifespan. Among all phishing contracts, *only 1.9% has remained active for more than a month.* For example, the contract `0xacbd` remained active for 575 days, with its first phishing transaction on April 7, 2023, and its last on November 3, 2024. During this period, it stole approximately $0.3 million from 1,095 victim accounts [45]. In contrast, *96.2% phishing contracts have a lifespan of less than a day.*

This difference can be attributed to the contract labeling mechanism. In this mechanism, security researchers report phishing accounts and their associated transactions to Etherscan and wallet providers. Etherscan then labels these phishing accounts, and wallets block transactions to them. Specifically, before displaying the transaction window, wallets check if the target address is blacklisted. If it is, an alert window appears, preventing users from signing the transaction. Therefore, once a contract is labeled as phishing, it becomes difficult for it to deceive users and steal tokens.

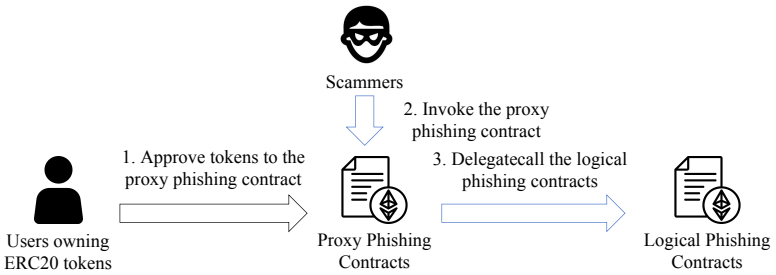



Fig. 6. An Example of Delegatecall in Phishing Contracts. Users first approve tokens to the proxy phishing contracts. Scammers then invoke the proxy contracts, which utilize delegatecall to execute the code in the logical phishing contracts.

As of the time of writing this paper, the contract `0xacbd` [45] has not yet been labeled, which helps explain why it has remained active for such a long period.

**Attempts to combat contract labeling mechanism.** Generally, there is a delay between the deployment time of phishing contracts and the time they are labeled. As a result, *scammers can evade this mechanism by deploying new phishing contracts frequently.* For example, Pink Drainer [17], a large-scale phishing group that has now disappeared, deployed a new phishing contract each day from Mar 18, 2024, to May 16, 2024 [41]. We have identified all 61 of these phishing contracts in our database.

Moreover, we have identified many phishing contracts that are used for only a single phishing transaction. For instance, phishing contract `0x60cf` was deployed on November 13, 2024, and quickly stole tokens from victim account `0xd98d` [47], but it did not generate any further phishing transactions. While these contracts can still be used for phishing, many of them share the same contract bytecode and deployer accounts, with each one typically associated with only a single phishing transaction. This leads us to conclude that these phishing contracts are intended for one-time use. Despite the significant costs for scammers to deploy new phishing contracts, this approach makes it easier to deceive users, as they are constantly faced with new accounts that have not yet been labeled. We have identified 31,156 instances of these one-time phishing contracts.

**Leveraging “delegatecall" to reduce deployment fee.** Delegatecall enables a proxy smart contract to execute code from a logical contract while operating within the storage and context of the calling contract [74]. It is frequently utilized to facilitate code reuse without duplication or to implement proxy contracts for upgradeable architectures. And we observe that scammers utilize delegatecall to reduce deployment fee. Figure 6 illustrates a real-world example of this tactic. On January 14, 2025, account `0xc6ba` approved its IQ tokens to a proxy phishing contract 0xfd7a. Subsequently, the scammer `0x0000` invoked the proxy phishing contract, which used delegatecall to execute the multicall phishing code from the logical phishing contract, transferring the IQ tokens [48]. In our dataset, there are 114 proxy phishing contracts, one logical phishing contract, and 3,196 related phishing transactions.

**Answer to RQ3.3:** 86.5% of phishing contracts include both “empty" payable functions and multicall functions to steal various types of tokens. 70.9% of contract profits are below $1,000. 96.2% of contracts have a lifespan shorter than one day. Scammers deploy phishing contracts daily or even use a contract for a single phishing attack to evade contract labeling mechanisms. They also utilize “delegatecall" to minimize the deployment costs of phishing contracts.

**Algorithm 2** Deployment Fund Tracing Algorithm

**Require:** Deployment transaction $\mathcal{T}$, Target account $\mathcal{A}_t$
**Ensure:** Tracing results $\mathcal{R}$

$\mathcal{R} \leftarrow \emptyset$ ▷ Initialize results storage
Extract deployer account $\mathcal{A}_d$ and deployment funds amount $\mathcal{F}_d$ from $\mathcal{T}$
$\mathcal{A}_{current} \leftarrow \mathcal{A}_d$ ▷ Start with deployer account
Depth $\leftarrow 0$ ▷ Initialize tracing depth
**while** True **do**
    Trace funds flowing into $\mathcal{A}_{current}$ ▷ Trace Fund Flow
    **if** $\mathcal{A}_{current}$ is a CEX, cross-chain, or mixer account **or** transaction is a phishing transaction **or** Depth $> 5$ **then**
        **Terminate tracing** and store results $\mathcal{R}$
        **Break**
    **end if**
    Extract the sender account $\mathcal{A}_{sender}$ that transferred ETH to $\mathcal{A}_{current}$
    $\mathcal{A}_{current} \leftarrow \mathcal{A}_{sender}$
    Depth $\leftarrow$ Depth + 1
**end while**
**return** $\mathcal{R}$

### 6.4 Deployer Account Analysis

In this section, we analyze the distribution of deployer accounts and the deployment fund sources of phishing contracts. Furthermore, we explore the relationships among major phishing contract deployers to identify large-scale phishing groups that consistently deploy such contracts.

**Overview of deployer accounts.** A total of 2,178 accounts deployed 37,654 phishing contracts in our dataset. We find that certain phishing accounts are responsible for deploying thousands of phishing contracts. Specifically, *nine phishing accounts have deployed over 500 phishing contracts, totaling 34,312, which accounts for 91.1% of the total.* Among them, the phishing account `0x0000` has deployed 16,322 phishing contracts [32], the highest number among all accounts. It began large-scale phishing contract deployment on November 2, 2023. Additionally, it uses contract `0xed0e` to automate phishing contract deployment [33]. As of this writing, it is still actively invoking contract `0xed0e` for each new phishing contract deployment.

**Distribution of deployment fund source.** To investigate the sources of funds used for deploying phishing contracts, we present a tracing algorithm outlined in Algorithm 2. The algorithm is composed of four key steps:

Step1: *Extract Information.* First, we extract the deployer account and the deployment funds amount from each contract's deployment transaction.
Step2: *Trace Fund Flow.* Next, we trace the funds flowing into the deployer account.
Step3: *Evaluate Trace Continuation.* We then evaluate whether to terminate the tracing process. If the sender account is a centralized exchange (CEX), cross-chain, or mixer account, it becomes difficult to continue tracing, so we terminate the search and store the results. If the transaction being traced is identified as a phishing transaction, we also terminate the process and label the sender as a victim account. Additionally, if the tracing depth exceeds five steps, the process is terminated.
Step4: *Repeat or Terminate.* If none of the termination conditions are met, we extract the sender account that transferred ETH to the target account and return to Step 2.

Table 2. Type of Phishing Contract Fund Source. For a contract, if all of its deployment funds are from the same entity type, we categorize it as the type that fund originates from the entity.

| Type | Example | Number | Proportion |
|---|---|---|---|
| From CEXs | 0x5190f328c68af2b3feb8a39e9dda716ed1dd5538 | 5,524 | 14.7% |
| From cross-chain bridges | 0xc39993830eebb88a02b2b2b74a695871a61acad2 | 2,913 | 7.7% |
| From mixers | 0xd2f586e44c2a6d22c5fc5edcb9f25e158bbb9751 | 18 | 0.04% |
| From victims | 0x9db1532603662538fa3952801f613e08068457a7 | 23,782 | 63.2% |


Fig. 7. An Example of Scammers Using Victims' tokens to Deploy New Phishing Contracts

For a contract, if all of its deployment funds are from the same type of entity, we mark that its deployment fund originates from the entity. For example, if a contract's deployment funds come from a CEX, we mark the contract's deployment funds as originating from the CEX. We discover that 85.6% of phishing contract deployment fund can be traced to a specific entity. Among them, deployment funds of 5,524 phishing contracts come from CEX. Our statistics show that the top five CEXs funding phishing contract deployments are Binance, SideShift, ChangeNow, Coinbase, and KuCoin, with proportions of 35.2%, 12.3%, 6.4%, 6.0%, and 3.4%, respectively. By supplying substantial criminal evidence of phishing contracts and their related fund flows to CEXs, victims can quickly uncover the true identities of the scammers. Besides, 63.2% of phishing contracts are funded directly from victim accounts. As can be seen from Fig. 7, scammers use ETH stolen from victims to deploy new phishing contracts and target new victims. Table 2 highlights example contracts with deployment funds originating from various entities.

- *CEX*: Phishing account `0x0000` received ETH from FixedFloat, then deployed the phishing contract `0x5190`.
- *Cross-chain bridges*: Phishing account `0xf672` obtained ETH from Stargate Finance, transferred it to `0x0000`, which then deployed the phishing contract `0xc399`.
- *Mixers*: Phishing account `0xc07a` received ETH from Tornado.Cash and subsequently deployed the phishing contract `0xd2f5`.
- *Victims*: Phishing account `0xa763` received ETH from victim `0x2f60` via phishing contract `0x6f3f`, transferred it to another phishing account `0xf813`, and then deployed the phishing contract `0x9db1`.

**Relationships between major contract deployers.** While tracing deployer funds, we observed fund flow connections between different deployer accounts. For instance, on November 13, 2024, the second-largest contract deployer transferred 2 ETH to the largest contract deployer [46]. Although this does not conclusively prove they are operated by the same entity, it indicates a connection between them. This observation motivates us to cluster the nine largest deployer accounts, each deploying over 500 phishing contracts, based on their fund flow relationships. Specifically, we consider two deployer accounts to be associated and cluster them if they have ever transferred tokens to each other, sent tokens to the same phishing account, or received tokens from the same

phishing account. Using this clustering rule, we find that *eight of the nine largest deployer accounts belong to the same phishing group* [32, 34–40]. Together, *they have deployed 32,290 phishing contracts since November 2, 2023, accounting for 85.7% of the total.* Currently, they are still actively deploying new phishing contracts. Furthermore, based on address labels from Etherscan, these accounts include phishing entities associated with both Inferno [43] and Angel Drainer [27], two major Web3 phishing groups. Recent reports suggest that Angel Drainer has absorbed Inferno's crypto-draining platform [29, 30], which explains why these two phishing groups appear clustered together in our analysis.

**Answer to RQ3.4:** Nine accounts deploy 91.1% of all phishing contracts. Scammers often use tokens stolen from victims to fund the deployment of new phishing contracts. In statistics, 85.6% of phishing contract deployment fund can be traced to a specific entity, including CEXs, mixers, cross-chain bridges, and victim accounts. Notably, 63.2% of them can be traced back to victim accounts. Eight of these nine major deployers exhibit fund flow connections, indicating they belong to the same phishing group. Collectively, They have deployed 85.7% of all phishing contracts.

## 7 Discussion and Implications

### 7.1 Mitigation Methods

Our work reveals the widespread prevalence of phishing contracts on Ethereum and the significant losses they have caused to users. Hence, we propose practical and effective strategies to protect users from these threats.

**User perspective.** When accessing a decentralized application and requesting services, users should closely inspect the website, including the URL, main page, sublinks, Twitter, and Discord links. Before signing a transaction, users should carefully review the transaction details, including the account and function call parameters. Additionally, they can verify the address label on Etherscan to determine if it is an official account.

**Wallet perspective.** For wallet developers, it is their responsibility to offer services that safeguard users from phishing attacks. They should regularly update and manage lists of phishing websites and accounts, ensuring users are alerted when interacting with potential threats. Furthermore, our system's insights can be integrated into wallets. As mentioned in Section 5.2, when users interact with a smart contract and send ETH, the wallet can verify whether the transaction is a phishing transaction that steals ETH without returning anything. Likewise, when users grant permissions for their ERC-20 tokens or NFTs to a smart contract, the wallet should assess whether it includes a multicall function that can only be invoked by a phishing account. If such risks are detected, the wallet should alert users.

**Other service provider perspective.** Meanwhile, CEXs should refuse to process tokens originating from phishing accounts. Cross-chain bridges should also block any cross-chain transactions that come from these accounts. Furthermore, features from specific projects, such as the USDC blacklist [24], can be leveraged to block phishing accounts and verify transactions effectively. Moreover, as indicated by our findings in Section 6.4, if certain accounts utilize tokens from CEXs or cross-chain bridges to deploy phishing contracts, these service providers should refuse to provide service to them. In Section 6.3, we observe that many phishing contracts share the same bytecode. The explorer should record this bytecode and label other contracts that match it. Additionally, the explorer should track major contract deployers and promptly label phishing contracts deployed by them.

### 7.2 Generalization of our Work

Our work aims to enhance public awareness of phishing contracts and furnish valuable insights into the whole ecosystem. Currently, we focus on phishing contracts on Ethereum. Nevertheless, it is important to note that our comprehensive measurement framework, from data collection to characterization, can be directly applied to all EVM-compatible chains (e.g., BSC and Tron). Indeed, we have identified instances of phishing contracts on alternative blockchains, such as 0x48c8 on Binance Smart Chain (BSC). [3] While we are currently dedicated to studying phishing contracts on Ethereum, our future goal is to develop a comprehensive system that can detect and analyze phishing contracts across various EVM-compatible chains.

### 7.3 Limitations of our Work

Once the details of our system are disclosed, scammers may modify their phishing contracts to evade detection. For example, they may change the ETH phishing function to behave like a legitimate function, returning valueless tokens. They could also alter the parameters and names of *multicall* functions to evade detection. Since we are unable to predict features of the modified phishing contracts, our detection system is limited to identifying phishing contracts based on their existing features. Nevertheless, as our system extracts and analyzes features from smart contract bytecodes for phishing contract detection, it can identify new phishing attacks if their bytecodes exhibit these characteristics. What's more, it has demonstrated remarkable effectiveness in identifying phishing contracts, making it a valuable data source for measurement purposes.

## 8 Related Work

### 8.1 Exploring Cybercrime on Ethereum

As various forms of cybercrime continue to emerge on Ethereum, numerous research efforts have been dedicated to analyzing and categorizing these threats [57–59, 63, 66, 67, 73, 78, 80]. Most of these studies focus on scams like rug pulls, fake tokens, honeypots, and giveaway scams, which differ from transaction-based phishing. For example, several studies propose techniques to extract transaction frequency and train AI models for detecting phishing accounts [57, 66]. Xia *et al.* [78] suggest extracting time-series, transactions, investors, and Uniswap-specific attributes to train a classifier for identifying scam tokens. Roy *et al.* [73] investigate and identify phishing scams related to NFT promotions on Twitter. Li *et al.* [67] analyze giveaway scams where users send tokens to a specified address, expecting to double their amount, but ultimately receive nothing in return. He *et al.* [59] detect and analyze transaction-based phishing websites on Ethereum that trick users into initiating phishing transactions, resulting in the withdrawal of all their tokens. Chen *et al.* [58] dissect payload-based transaction phishing on Ethereum. Huang *et al.* [63] carry out an empirical study and create a prototype detection system to identify NFT rug pulls. Ye *et al.* [80] present an in-depth analysis of visual scams associated with cryptocurrency wallets.

### 8.2 Investigating Blockchain Transactions

Several studies have focused on analyzing various types of blockchain transactions [61, 62, 68, 76, 77]. Hu *et al.* [77] take the first step to understand state-of-the-art Bitcoin mixing services. Wang *et al.* [76] characterize ERC20 token approval transactions and their associated security issues on Ethereum. Lyu *et al.* [68] analyze private transactions and their security implications on Ethereum. Hu *et al.* [61] provide the first comprehensive analysis of cross-chain transactions, creating a large dataset, uncovering usage patterns, and developing an automated detector that identifies hundreds of abnormal transactions, highlighting security risks and misuse. Hu *et al.* [62] investigate Instant

[3] 0x48c80afd7ac03617b9f709a8eee7f2d106f80f82

Cryptocurrency Exchange transactions, revealing $12 million in illicit funds laundered due to weak KYC and limited transparency.

### 8.3 Measuring Phishing Websites

When measuring phishing websites, the targets can be classified into three categories: phishing campaigns, phishing website cloaking techniques, and toolkits. The first category involves empirical investigations into the entire lifecycle and effectiveness of specific phishing attacks [60, 65, 72, 75]. For example, Ho *et al.* [60] provide the first comprehensive analysis of lateral phishing attacks, uncovering the detailed aspects of these incidents on a large scale. The second category focuses on analyzing the interaction between phishing websites and anti-phishing detectors, as well as proposing new mechanisms to protect users [69–71, 82, 83]. For instance, Oest *et al.* [71] map out the anti-phishing ecosystem by analyzing it through the lens of phishing toolkits. The third category involves evaluating the characteristics of phishing toolkits and developing detectors specifically tailored for these tools [54, 59, 64]. For example, Kondracki *et al.* [64] conduct the first empirical study on Man-in-the-Middle (MITM) phishing toolkits, uncovering their prevalence and associated risks.

## 9 Conclusion

In this paper, we presents the first empirical study of phishing contracts on Ethereum. We begin by constructing a sample dataset containing 790 reported phishing contracts and identifying key features of them. We then propose to collect phishing contracts by analyzing suspicious functions in the bytecode and simulating transactions. In this way, we have built the first large-scale phishing contract dataset on Ethereum, consisting of 37,654 phishing contracts. Leveraging this dataset, we collect phishing transactions and conduct measurements from the perspectives of victim accounts, phishing contracts, and deployer accounts. In total, these phishing contracts have earned $190.7 million from 171,984 victim accounts. Scammers frequently deploy phishing contracts to bypass contract labeling security mechanisms. What's more, we identify a large-scale phishing group responsible for deploying 85.7% of all phishing contracts. Our work aims to serve as a guide to protect users against transaction-based phishing.

## Acknowledgments

We would like to thank the anonymous reviewers for their comments that greatly helped improve the presentation of this paper. We also want to thank Prof. Gareth Tyson for shepherding our paper. Additionally, the first author of this paper would like to thank Prof. Ting Yu, Prof. Xiaosong Ma, Qingyuan Chen, Jia He, and Mingshan Hee personally for their help with the author's study in MBZUAI. This work is partially supported by the National Key R&D Program of China (No. 2022YFE0113200), the National Natural Science Foundation of China (NSFC) under Grant U21A20464. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of funding agencies.

## References

[1] 2023. *Amazon NFTs, Losing $2m in a phishing attack, $105m payday, Is Bitcoin the best performing asset in the world this year?* Retrieved Oct 8, 2024 from https://fomofix.substack.com/p/amazon-nfts-losing-2m-in-a-phishing?utm_source=twitter&utm_campaign=auto_share&r=1duf4y

[2] 2023. *Crypto Google search ad phishing has resulted $4.16 million loss.* Retrieved Oct 8, 2024 from https://twitter.com/WuBlockchain/status/1651514902408986626

[3] 2023. *Crypto phishing scammer Monkey Drainer shuts down services.* Retrieved Oct 8, 2024 from https://cryptoslate.com/crypto-phishing-scammer-monkey-drainer-shuts-down-services

[4] 2023. *Cyber Security Firm CertiK Unmasks Scammers Linked To $4.3M Porsche NFT Phishing Scam.* Retrieved Oct 8, 2024 from https://www.business2community.com/nft-news/cyber-security-firm-certik-unmasks-scammers-linked-to-4-3m-porsche-nft-phishing-scam-02617446

[5] 2023. *Fake_Phishing8210 on etherscan.com has taken $1.24M in USDC from a victim.* Retrieved Oct 8, 2024 from https://twitter.com/CertiKAlert/status/1623131855661805571

[6] 2023. *Robinhood and NFT project Azukis Twitter hacked; 122 NFTs worth 484.99 ETH stolen from the latter.* Retrieved Oct 8, 2024 from https://gemhodlers.com/robinhood-and-nft-project-azukis-twitter-hacked-122-nfts-worth-484-99-eth-stolen-from-the-latter

[7] 2023. *Venom Drainer has Drained $27M from 15k victims.* Retrieved Oct 8, 2024 from https://twitter.com/realScamSniffer/status/1642813130454765568

[8] 2024. *alchemy.* https://www.alchemy.com/.

[9] 2024. *Chainabuse.* https://www.chainabuse.com.

[10] 2024. *Contract ABI Specification.* https://docs.soliditylang.org/en/latest/abi-spec.html.

[11] 2024. *Dedaub.* https://app.dedaub.com/decompile.

[12] 2024. *DeFiLlama.* https://defillama.com/chains.

[13] 2024. *Display in MetaMask.* https://docs.metamask.io/wallet/how-to/display/method-names/.

[14] 2024. *ERC20 token standard.* https://eips.ethereum.org/EIPS/eip-20.

[15] 2024. *Ethereum Signature Database.* https://www.4byte.directory.

[16] 2024. *Etherscan.* https://etherscan.io.

[17] 2024. *How Phishing Websites Bypass Wallet Security Alerts: Strategies Unveiled.* https://blocksec.com/blog/how-phishing-websites-bypass-wallet-security-alerts-strategies-unveiled.

[18] 2024. *How to Avoid Being a Web3 Phishing Victim.* https://blocksec.com/blog/how-to-avoid-being-a-web3-phishing-victim.

[19] 2024. *Protecting Your Assets: Safeguarding Against Phishing Scams in Web3.* https://blocksec.com/blog/protecting-your-assets-safeguarding-against-phishing-scams-in-web3.

[20] 2024. *Solidity.* https://soliditylang.org.

[21] 2024. *Spin up your own Ethereum node.* https://ethereum.org/en/developers/docs/nodes-and-clients/run-a-node/.

[22] 2024. *Uniswap.* https://app.uniswap.org.

[23] 2024. *Uniswap V3: Multicall 2.* https://etherscan.io/address/0x5ba1e12693dc8f9c48aad8770482f4739beed696.

[24] 2024. *USDC Banned Addresses.* https://dune.com/phabc/usdc-banned-addresses.

[25] 2024. *Vyper.* https://docs.vyperlang.org/en/latest/index.html.

[26] 2024. *Zero-dollar Purchases in Blur.* https://x.com/MetaSleuth/status/1633318417938939905.

[27] 2025. *Angel Drainer upgraded, deploying 300+ malicious DApps in 4 days.* Retrieved Jan 14, 2025 from https://cointelegraph.com/news/angel-wallet-drainer-is-back-with-300-malicious-d-apps-and-counting-block-aid

[28] 2025. *Binance deposit account 0x18c4.* Retrieved Jan 13, 2025 from https://etherscan.io/address/0x18c4bf7c470069b9d18b6a5670e457de3983c299

[29] 2025. *Crypto Threats Grow as Angel Drainer Takes Control of Inferno's Draining Tools Report 1.* Retrieved Jan 14, 2025 from https://www.bitget.com/news/detail/12560604288409

[30] 2025. *Crypto Threats Grow as Angel Drainer Takes Control of Inferno's Draining Tools Report 2.* Retrieved Jan 14, 2025 from https://x.com/blockaid_/status/1849122383585571313

[31] 2025. *Fake Phishing 182233.* Retrieved Jan 14, 2025 from https://etherscan.io/address/0x000011387eb24f199e875b1325e4805efd3b0000

[32] 2025. *Fake Phishing 187019.* Retrieved Jan 16, 2025 from https://etherscan.io/address/0x0000db5c8b030ae20308ac975898e09741e70000

[33] 2025. *Fake Phishing 188615.* Retrieved Jan 16, 2025 from https://etherscan.io/address/0xed0e416e0feea5b484ba5c95d375545ac2b60572

[34] 2025. *Fake Phishing 228344.* Retrieved Jan 16, 2025 from https://etherscan.io/address/0x0000d38a234679f88dd6343d34e26dcb50c30000

[35] 2025. *Fake Phishing 268895.* Retrieved Jan 16, 2025 from https://etherscan.io/address/0x00006bbe73c2c7878dc9dc19e0d947e4c4270000

[36] 2025. *Fake Phishing 270928.* Retrieved Jan 16, 2025 from https://etherscan.io/address/0x000022e528968de17f8ec16a3128dd403a8f0000

[37] 2025. *Fake Phishing 276019.* Retrieved Jan 16, 2025 from https://etherscan.io/address/0x0000f0d34d51d23b0dd75a6b720c7f4f10430000

[38] 2025. *Fake Phishing 291040.* Retrieved Jan 16, 2025 from https://etherscan.io/address/0x0000e53a54a68e047b73cffdea45f44368a60000

[39] 2025. *Fake Phishing 322778.* Retrieved Jan 16, 2025 from https://etherscan.io/address/0x00000952c5165e391ed0f8adffcfaf45f3b80000
[40] 2025. *Fake Phishing 324041.* Retrieved Jan 16, 2025 from https://etherscan.io/address/0x000099b4a4d3ceb370d3a8a6235d24e07a8c0000
[41] 2025. *Fake Phishing 328458.* Retrieved Jan 14, 2025 from https://etherscan.io/address/0x29fb39f4bf464fabb96d9fe49e97d69b05d05bc3
[42] 2025. *Fake Phishing 532355.* Retrieved Jan 14, 2025 from https://etherscan.io/address/0x0001be0b7008d9273edbac98e6c841972d458740
[43] 2025. *Inferno Drainer Phishing Incident.* Retrieved Jan 13, 2025 from https://etherscan.io/tx/0xb5f03259859c10243a85f7a9ca27d8b45e67ce6463800193634d27718ba162a0
[44] 2025. *Phishing Contract 0x0005.* Retrieved Jan 14, 2025 from https://etherscan.io/address/0x00055c007cbcef4d548c922a50dffe57f0699276
[45] 2025. *Phishing contract 0xacbd.* Retrieved Jan 14, 2025 from https://etherscan.io/address/0xacbd7c3357687be445985fcab1ff4551c88aa375
[46] 2025. *Phishing transaction 0x829c.* Retrieved Jan 14, 2025 from https://etherscan.io/tx/0x829c393b2b296e56089a4062f7da245ad6b64bd2550e68b934e5cfd3e9c0f4d4
[47] 2025. *Phishing transaction 0xa915.* Retrieved Jan 14, 2025 from https://etherscan.io/tx/0xa9156965f8f7fe584a1bdad08edb134a5db10c46ecffb7660fc20dd9ec4d8b18
[48] 2025. *Phishing transaction 0xa915.* Retrieved Jan 14, 2025 from https://etherscan.io/tx/0x842fe65219146cd3056d5a3dfb52533014da6d2a03e9dfccb33f10f749afa4d0
[49] 2025. *Transaction 0x59c3.* Retrieved Jan 14, 2025 from https://etherscan.io/tx/0x59c3116671b77cf3f9de44c6f38f676036ea8d3f19d0881860b4518f11c5f78b
[50] 2025. *Transaction 0xaa4b.* Retrieved Jan 14, 2025 from https://etherscan.io/tx/0xaa4bb33c140949719f95717a3860ffa03f99dd2673d5370143be735e8b78ec42
[51] 2025. *Victim account 0x285c.* Retrieved Jan 13, 2025 from https://etherscan.io/address/0x285c58205e903f6bcbd5d2d0b91f06dc7c75aefb
[52] 2025. *Victim account 0x4a4e.* Retrieved Jan 14, 2025 from https://www.coingecko.com/
[53] 2025. *Victim account 0x4a4e.* Retrieved Jan 13, 2025 from https://etherscan.io/address/0x4a4e08aefa0ab8d9225ca9ff7c1d7f9bc0a0a979
[54] Hugo Bijmans, Tim Booij, Anneke Schwedersky, Aria Nedgabat, and Rolf van Wegberg. 2021. Catching phishers by their bait: Investigating the dutch phishing landscape through phishing kit detection. In *30th USENIX security symposium (USENIX security 21)*. 3757–3774.
[55] Liang Chen, Jiaying Peng, Yang Liu, Jintang Li, Fenfang Xie, and Zibin Zheng. 2020. Phishing scams detection in ethereum transaction network. *ACM Transactions on Internet Technology (TOIT)* 21, 1 (2020), 1–16.
[56] Ting Chen, Zihao Li, Xiapu Luo, Xiaofeng Wang, Ting Wang, Zheyuan He, Kezhao Fang, Yufei Zhang, Hang Zhu, Hongwei Li, et al. 2021. Sigrec: Automatic recovery of function signatures in smart contracts. *IEEE Transactions on Software Engineering* 48, 8 (2021), 3066–3086.
[57] Weili Chen, Xiongfeng Guo, Zhiguang Chen, Zibin Zheng, and Yutong Lu. 2020. Phishing Scam Detection on Ethereum: Towards Financial Security for Blockchain Ecosystem.. In *IJCAI*, Vol. 7. 4456–4462.
[58] Zhuo Chen, Yufeng Hu, Bowen He, Dong Luo, Lei Wu, and Yajin Zhou. 2024. Dissecting Payload-based Transaction Phishing on Ethereum. *arXiv preprint arXiv:2409.02386* (2024).
[59] Bowen He, Yuan Chen, Zhuo Chen, Xiaohui Hu, Yufeng Hu, Lei Wu, Rui Chang, Haoyu Wang, and Yajin Zhou. 2023. TxPhishScope: Towards Detecting and Understanding Transaction-based Phishing on Ethereum. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*. 120–134.
[60] Grant Ho, Asaf Cidon, Lior Gavish, Marco Schweighauser, Vern Paxson, Stefan Savage, Geoffrey M Voelker, and David Wagner. 2019. Detecting and characterizing lateral phishing at scale. In *28th USENIX Security Symposium (USENIX Security 19)*. 1273–1290.
[61] Xiaohui Hu, Hang Feng, Pengcheng Xia, Gareth Tyson, Lei Wu, Yajin Zhou, and Haoyu Wang. 2024. Piecing Together the Jigsaw Puzzle of Transactions on Heterogeneous Blockchain Networks. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 8, 3 (2024), 1–27.
[62] Yufeng Hu, Yingshi Sun, Lei Wu, Yajin Zhou, and Rui Chang. 2024. Towards Understanding and Analyzing Instant Cryptocurrency Exchanges. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 8, 3 (2024), 1–24.
[63] Jintao Huang, Ningyu He, Kai Ma, Jiang Xiao, and Haoyu Wang. 2023. A Deep Dive into NFT Rug Pulls. *arXiv preprint arXiv:2305.06108* (2023).
[64] Brian Kondracki, Babak Amin Azad, Oleksii Starov, and Nick Nikiforakis. 2021. Catching transparent phish: analyzing and detecting MITM phishing toolkits. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and*

*Communications Security*. 36–50.
[65] Daniele Lain, Kari Kostiainen, and Srdjan Čapkun. 2022. Phishing in organizations: Findings from a large-scale and long-term study. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 842–859.
[66] Sijia Li, Gaopeng Gou, Chang Liu, Chengshang Hou, Zhenzhen Li, and Gang Xiong. 2022. TTAGN: Temporal transaction aggregation graph network for ethereum phishing scams detection. In *Proceedings of the ACM Web Conference 2022*. 661–669.
[67] Xigao Li, Anurag Yepuri, and Nick Nikiforakis. 2023. Double and Nothing: Understanding and Detecting Cryptocurrency Giveaway Scams. In *Network and Distributed Systems Security (NDSS) Symposium*.
[68] Xingyu Lyu, Mengya Zhang, Xiaokuan Zhang, Jianyu Niu, Yinqian Zhang, and Zhiqiang Lin. 2022. An empirical study on ethereum private transactions and the security implications. *arXiv preprint arXiv:2208.02858* (2022).
[69] Adam Oest, Yeganeh Safaei, Adam Doupé, Gail-Joon Ahn, Brad Wardman, and Kevin Tyers. 2019. Phishfarm: A scalable framework for measuring the effectiveness of evasion techniques against browser phishing blacklists. In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1344–1361.
[70] Adam Oest, Yeganeh Safaei, Penghui Zhang, Brad Wardman, Kevin Tyers, Yan Shoshitaishvili, Adam Doupé, and Gail-Joon Ahn. 2020. Phishtime: Continuous longitudinal measurement of the effectiveness of anti-phishing blacklists. In *Proceedings of the 29th USENIX Conference on Security Symposium*. 379–396.
[71] Adam Oest, Yeganeh Safei, Adam Doupé, Gail-Joon Ahn, Brad Wardman, and Gary Warner. 2018. Inside a phisher's mind: Understanding the anti-phishing ecosystem through phishing kit analysis. In *2018 APWG Symposium on Electronic Crime Research (eCrime)*. IEEE, 1–12.
[72] Adam Oest, Penghui Zhang, Brad Wardman, Eric Nunes, Jakub Burgis, Ali Zand, Kurt Thomas, Adam Doupé, and Gail-Joon Ahn. 2020. Sunrise to sunset: Analyzing the end-to-end life cycle and effectiveness of phishing attacks at scale. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*.
[73] Sayak Saha Roy, Dipanjan Das, Priyanka Bose, Christopher Kruegel, Giovanni Vigna, and Shirin Nilizadeh. 2023. Demystifying NFT Promotion and Phishing Scams. *arXiv preprint arXiv:2301.09806* (2023).
[74] Nicola Ruaro, Fabio Gritti, Robert McLaughlin, Ilya Grishchenko, Christopher Kruegel, and Giovanni Vigna. 2024. Not your type! Detecting storage collision vulnerabilities in ethereum smart contracts. In *Proc. Netw. Distrib. Syst. Secur. Symp.* 1–17.
[75] Amber Van Der Heijden and Luca Allodi. 2019. Cognitive triaging of phishing attacks. In *28th USENIX Security Symposium (USENIX Security 19)*. 1309–1326.
[76] Dabao Wang, Hang Feng, Siwei Wu, Yajin Zhou, Lei Wu, and Xingliang Yuan. 2022. Penny wise and pound foolish: quantifying the risk of unlimited approval of ERC20 tokens on ethereum. In *Proceedings of the 25th International Symposium on Research in Attacks, Intrusions and Defenses*. 99–114.
[77] Lei Wu, Yufeng Hu, Yajin Zhou, Haoyu Wang, Xiapu Luo, Zhi Wang, Fan Zhang, and Kui Ren. 2021. Towards understanding and demystifying bitcoin mixing services. In *Proceedings of the Web Conference 2021*. 33–44.
[78] Pengcheng Xia, Haoyu Wang, Bingyu Gao, Weihang Su, Zhou Yu, Xiapu Luo, Chao Zhang, Xusheng Xiao, and Guoai Xu. 2021. Trade or trick? detecting and characterizing scam tokens on uniswap decentralized exchange. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 5, 3 (2021), 1–26.
[79] Yijun Xia, Jieli Liu, and Jiajing Wu. 2022. Phishing detection on ethereum via attributed ego-graph embedding. *IEEE Transactions on Circuits and Systems II: Express Briefs* 69, 5 (2022), 2538–2542.
[80] Guoyi Ye, Geng Hong, Yuan Zhang, and Min Yang. 2024. Interface Illusions: Uncovering the Rise of Visual Scams in Cryptocurrency Wallets. In *Proceedings of the ACM on Web Conference 2024*. 1585–1595.
[81] Qi Yuan, Baoying Huang, Jie Zhang, Jiajing Wu, Haonan Zhang, and Xi Zhang. 2020. Detecting phishing scams on ethereum based on transaction records. In *2020 IEEE international symposium on circuits and systems (ISCAS)*. IEEE, 1–5.
[82] Penghui Zhang, Adam Oest, Haehyun Cho, Zhibo Sun, RC Johnson, Brad Wardman, Shaown Sarker, Alexandros Kapravelos, Tiffany Bao, Ruoyu Wang, et al. 2021. Crawlphish: Large-scale analysis of client-side cloaking techniques in phishing. In *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1109–1124.
[83] Penghui Zhang, Zhibo Sun, Sukwha Kyung, Hans Walter Behrens, Zion Leonahenahe Basque, Haehyun Cho, Adam Oest, Ruoyu Wang, Tiffany Bao, Yan Shoshitaishvili, et al. 2022. I'm SPARTACUS, No, I'm SPARTACUS: Proactively Protecting Users from Phishing by Intentionally Triggering Cloaking Behavior. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. 3165–3179.

Received January 2025; revised April 2025; accepted April 2025