

# Pulling Off The Mask: Forensic Analysis of the Deceptive Creator Wallets Behind Smart Contract Fraud

Anonymous Authors

**Abstract**—Criminals, using crypto wallets referred to as Deceptive Creator Wallets (DCWs), have orchestrated fraudulent activities by luring victims to transfer funds to fraud smart contracts. Since it is almost impossible to reverse the transactions or pinpoint the true identity of the criminals, the industry has turned to flagging such contracts as user warnings. However, the current mitigation focuses on individual contracts, overlooking the DCWs behind the scenes. Consequently, our research found that this oversight allows fraud to thrive. To address this, we developed Cybertrack, an automated forensic analysis pipeline that processes a single fraud contract and generates evidence that the legal authorities need to mitigate the fraud. Applying Cybertrack to 157 confirmed fraud contracts, our research uncovered 1,283,198 associated contracts linked to 91 DCWs, responsible for 2,638,752 ETH (\$2,089,504,682) in illicit profits. More alarmingly, Cybertrack traces the fraudulent activities back to September 2017. In response, we are closely collaborating with Etherscan and FBI to combat the fraud identified in our study.

## 1. Introduction

Smart contracts have been exploited to orchestrate fraudulent activities, resulting in financial losses [1], [2]. In this scenario, the criminals use their crypto wallets referred to as Deceptive Creator Wallets (DCWs) to deploy fraud smart contracts. The criminals then lure victims with false promises, leading them to transfer funds to these contracts, which then illicitly divert the victims assets to criminal-designated *recipients*. The irreversible and anonymous nature of blockchain makes it impossible to reverse the transactions or to identify the identity of these criminals. To mitigate this, the industry flags fraud contracts as warnings to users, as illustrated in Figure 1 on Etherscan [3]. In fact, several works have been proposed to automate fraud contract reporting [4]–[18]. Unfortunately, the current mitigation focuses on individual contracts, overlooking the DCW orchestrating the fraud.

Compounding the issue is the low cost of smart contract deployment, as DCWs can continuously deploy new fraud contracts to avoid detection. Additionally, blockchain not only enables wallets to deploy contracts but also allows contracts to deploy others, giving DCWs a fast-track method for fraud contract creation through continuous invocation. DCWs use this to conceal the

recipients of fraud contracts by *dynamically resolving* on-chain data. This approach leads to quick deployment of complex, interconnected fraud networks, where these *associated contracts* exhibit various *capabilities*, including asset transfer and new contract deployment.

Imagine this forensic scenario: FBI agents receive a report concerning a contract suspected of being involved in fraudulent activities. Ideally, FBI agents would go beyond merely mitigating the reported fraud contract; they would proactively delve into investigating the DCW orchestrating these activities to collect critical evidence. These evidence should encompass: ① The associated contracts from the same DCW along with their recipients. ② The provenance of dynamically resolved recipients, and ③ The attribution of capabilities to the contracts. FBI agents could then submit collected evidence to the court. Upon authorization, FBI agents could utilize Evidence ① to flag additional accounts (i.e., wallets, contracts) on the blockchain and freeze assets [19], effectively disrupting the fraudulent activities. Additionally, Evidence ② sheds light on the *origins* of dynamically resolved recipients, providing agents with indicators of early recipient changes and enabling more proactive mitigation. Finally, the analysis of contract capabilities in Evidence ③ uncovers targeted mitigation strategies for specific capabilities (e.g., monitoring contracts designating fraud contract recipients).

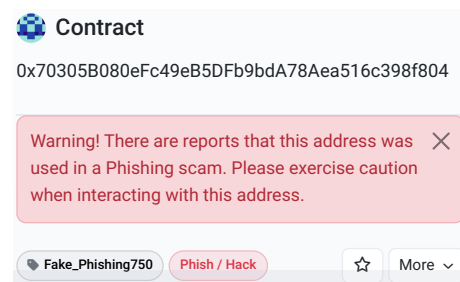


Figure 1: Flagged fraud contract.

Traditionally, FBI agents would acquire such evidence by relying on *explicit* clues, the historical transactions. In fact, prior research [4], [11] leveraged these explicit clues to *identify* fraud contracts. However, these contracts are just the starting point for forensic analysis. A smart contract can execute various transactions depending on different conditions. Consequently, the FBI agents quickly find out that a forensic approach that focuses strictly on

*explicit* clues might cause investigators to omit evidence from yet-to-be-executed transactions, thereby halting investigation until the relevant transactions occurred.

From a different perspective, contract transactions on blockchain are direct results of their implementations. Therefore, we shifted our focus to how a smart contract’s implementation could reveal *implicit* clues of fraudulent activities orchestrated by DCW, such as: (1) future transactions that DCW’s contract is programmed to execute, (2) the origin of the recipients used by DCW’s contract, and (3) the capabilities with which the contract is equipped. In fact, several works have been proposed [20]–[26] to perform program analysis on smart contracts. However, these techniques target vulnerability detection in benign contracts. The significant distinction between identifying vulnerabilities and discerning fraud capabilities means that these methods are not readily adaptable for extracting forensic clues from fraud contracts and the DCW operating behind the scenes.

Drawing inspiration from real-world forensic investigations that gather evidence from various clues at crime scenes, we propose combining explicit clues with advanced program analysis of fraud contracts for enhanced forensic analysis. Based on this insight, we developed Cybertrack, a post-detection forensics pipeline enabling the FBI agents to extract three key pieces of evidence of DCW. Given one fraud contract, Cybertrack first uses *Associated Contracts Recovery* (§3.1.1) to pinpoint associated contracts by utilizing explicit clues. This process mirrors the method of unraveling a criminal network, starting from one identified suspect and extending to the ringleader, additional suspects, and the middleman orchestrating the recruitment. Next, Cybertrack conducts symbolic analysis on each contract deployed by DCW (§3.1.2), identifying all potential recipients without depending on historical transactions. This step is comparable to identifying all possible contacts of the suspects, even those yet to engage in criminal activity. Cybertrack then combines explicit and implicit clues in *Recipient Provenance Investigation* (§3.2) to uncover the *origin* of dynamically resolved recipients. This resembles discovering the middleman’s hidden list of recruits, enabling authorities to detect future suspect recruitment early. Finally, Cybertrack applies *Capability Attribution Analysis* (§3.3) to assign specific capabilities to each contract deployed by DCW, similar to a detective deducing the roles each suspect plays in a criminal network to tailor countermeasures for each role.

Deploying Cybertrack on 157 confirmed fraud contracts, our work identified 1,283,198 associated contracts across 91 DCWs, making in total of 2,638,752 ETH (\$2,089,504,682) illicit profit. More alarmingly, Cybertrack’s analysis tracks the fraudulent activities back to September 2017. We are closely collaborating with Etherscan [3] and FBI [27] to mitigate the fraud uncovered by our study. Lastly, we have made Cybertrack available at: <redacted>

## 2. Motivation

Accounts (i.e., wallets and contracts) on Ethereum are identified by a 40-character hexadecimal strings, known as addresses. Similarly, transactions on Ethereum can be identified by 64-character hexadecimal strings, commonly referred to as transaction hashes. To enhance the readability of the paper, we have employed abbreviations of last 6 characters of addresses and transaction hashes and prefix the abbreviations with W- for wallet, C- for contract, and T- for transaction. For example, the fraud contract shown in Figure 2a can be identified by address, 0x70305b080efc49eb5dfb9bda78aea516c3 98f804. In this paper, we will refer it as C- 98f804. For the full addresses and transaction hashes used throughout this paper, readers are directed to Table 6 in §B.

### 2.1. Background

**Smart Contract.** A smart contract is a self-executing program operating on a blockchain, set to action when specific predefined conditions are satisfied. The execution of smart contract comprises three core components: (1) The contract’s bytecode, which contains the compiled instructions of the smart contract code. (2) A execution context, which includes the Program Counter (PC), available gas, stack, and memory. These components manage the execution flow and intermediate computations during the execution of a smart contract; (3) A persistent storage mechanism specific to each smart contract, providing a mapping from 256-bit words to corresponding 256-bit words. This storage is utilized by smart contracts for preserving state across transactions.

**Transaction & Trace.** A transaction represents an action initiated by an external account (i.e., a user’s wallet) that interacts with the blockchain. Transactions can encompass various actions such as transferring ETH, interacting with a smart contract, or even deploying a new contract. Traces, on the other hand, provide a step-by-step execution logs that detail all the internal calls and state changes triggered by a transaction. For example, consider a transaction where a user sends ETH to a smart contract, which then distributes this ETH to other addresses based on its logic. The transaction itself records the user’s action of sending ETH to the contract. However, the trace of this transaction would reveal the detailed sequence of events inside the contract, such as the contract calling its internal functions to distribute ETH to other addresses.

**Event Logs.** Event logs are small data amounts on the blockchain, utilizing five opcodes (LOG0 to LOG4) for log emission. Each log can include up to four 32-byte topics and a data section. The topics typically describe the event, often incorporating the event signature—a Keccak-256 [28] hash of the event name along with its parameter types. This allows for targeted searches, such as identifying logs for specific events or addresses. The data section complements the topics by providing

non-searchable, additional information. It can contain more complex details, like arrays or strings, making the logs both comprehensive and flexible. Consider an event like event Event(address indexed from, address to). For this event, the first topic is derived using a Keccak-256 hash on the event signature Event(address,address). The second topic is the indexed from address. Since the to parameter is not indexed, its value is stored in the log’s data section.

## 2.2. Preliminary Forensics Investigation

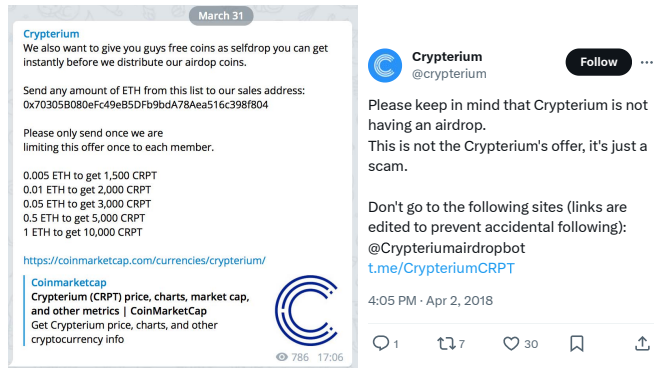


Figure 2: Fraud contract in real world.

To illustrate the method of deriving three pieces of evidence, we have devised a hypothetical forensic investigation scenario modeled on a real-world case. FBI agents received information from an individual who reported being defrauded by a deceptive message on Telegram [29]. As shown in Figure 2a, DCW propagated a deceptive message impersonating Crypterium [30], a blockchain startup. In this message, the DCW promised a giveaway, conditional upon interactions with the contract identified by C-98f804. Upon delving deeper, the agents uncovered several similar fraudulent messages disseminated across various platforms, as listed in Figure 5 and Figure 6 in §A. To ascertain the fraudulent nature of this contract, the agents located a confirmation from Crypterium on X.com [31], as evidenced in Figure 2b.

At this juncture, the agent possessed only the confirmed fraud contract address, C-98f804. To proceed with legal action, the agents required three pieces of evidence, as outlined in §1, which would be necessary to present in court. This would support a more proactive mitigation strategies, such as flagging more fraud contracts or freezing the associated wallets or contracts (as was previously done by the FBI [19]).

**Associated Contracts & Recipients.** Extracting Evidence 1 consists of two steps: first, identifying the associated contracts, and second, pinpointing their recipients. Our approach begins by leveraging the transparent nature of blockchain to reveal associated contracts. Specifically, our investigation started with an

TABLE 1: RECOVERED EVIDENCE FROM CYBERTRACK’S FORENSICS INVESTIGATION.

<b>Fraud Contract</b>	C-98f804
<b>Creator</b>	W-521058
<b>Evidence 1</b>	<b>Associated Contracts:</b> Direct: 395,685 (11) Indirect: 65,330 (0) <b>Recipients: W-763bc0</b>
<b>Evidence 2</b>	C-48d304 → Storage → Contract
<b>Evidence 3</b>	attr <sub>1</sub> : Forced Transfer (105,015) attr <sub>2</sub> : Contract Self-Replication (1) attr <sub>3</sub> : Event Emission (395,684)
<b>Fraud Impact</b>	<b>Victim:</b> 232,011 <b>ETH:</b> 1,240,355

█: Number of contracts flagged on Etherscan [3].

analysis of transactions to C-98f804, leading us to transaction T-8bbae5 that deployed the contract. The sender information embedded in this transaction reveals the DCW as W-521058. Tracing back from this transaction, we further explored other contracts deployed by W-521058 directly, amounting to 395,685 in total. Subsequent examination of transactions from these contracts unveiled a total of 251,087 with historical transactions to the same recipient W-763bc0.

Another problem that emerged during this process is that 144,597 (36.5%) contracts show no historical transactions. Sole reliance on explicit clues cannot reveal the recipients of these contracts. Instead, we turned our attention to the first implicit clue: *the contract’s implementation could reveal the future transactions that the contracts is programmed to execute.* Despite the diverse logic in fraud contracts, they still need to employ CALL opcode for ETH transfers. A detailed analysis of the CALL callsites within these contracts revealed that all 144,597 contracts possess the capability to transfer victim ETH to a single recipient, W-763bc0. As shown on Row 3 of Table 1, this led to the generation of Evidence 1, showing that there are 395,684 associated contracts capable of redirecting victim assets to the recipient wallet W-763bc0. Notably, one contract deployed by W-521058 remains unattributed. The subsequent sections will introduce the methods to dissect this singular contract.

**Masquerading As Inactive.** The previous analysis of W-521058 shows a constant fraud contracts deployment ranged from Oct 2017 to Jan 2021. At first glance, FBI agents might conclude that DCW ceased the fraud post-Jan 2021 due to a halt in deploying new fraud contracts. However, they soon realized that 98.4% of transactions initiated by W-521058 after that were invoking C-48d304.

This motivated us to switch our focus to contract C-48d304. Our analysis of the implementation showed that C-48d304 uses opcode CREATE2 to deploy new contracts, activated upon its invocation. To make things worse, DCW used C-48d304 to indirectly deploy new

contracts, which were designed to transfer victim assets. This led to the discovery of an additional 65,330 associated contracts, as detailed on Row 3 of Table 1, significantly expanding the scope of the investigation.

Given the newly discovered associated contracts deployed *indirectly* by DCW, FBI agents now need to know whether these contracts share the same recipients as before. Surprisingly, FBI agents quickly realized that they could not extract recipient addresses as before by examining the address passed to the CALL opcode, because the contracts dynamically resolved the recipients. This brought us to the second implicit clue: *in-depth program analysis on the contract could reveal the origin of dynamically resolved recipients*. Equipped with this implicit clue, we discovered that the newly identified contracts retrieve values from their *storage*, which are then passed as recipient addresses to the CALL opcode. Recognizing that these indirectly deployed contracts *use* storage to determine recipients, the key question became: who *defines* this storage. To answer this question, we delve deeper into the contract deployment chain (i.e., W-521058 → C-48d304 → *Child\_Contracts*). It turned out that the parent contract C-48d304 defines the storage of the child contracts during the deployment process. Interestingly, the recipient address set by the parent contract points to the same recipient W-763bc0. By performing the provenance analysis on the contract deployment chain, we generated Evidence ② as shown on Row 4 of Table 1: The reconstructed provenance of the fraud contract deployment chain reveals that 65,330 contracts lack hardcoded recipients. Instead, these contracts determine the recipient dynamically via their own storage parameters, as defined by the parent contract C-48d304.

**Capabilities Attribution.** Evidence ① and Evidence ② present a good opportunity for the agents to flag the accounts mentioned therein, serving as a *reactive* strategy to mitigate the ongoing fraudulent activities. However, due to the immutable nature of the blockchain, completely eliminating DCW from the system is almost impossible. Therefore, it becomes equally crucial for the FBI agents to implement *proactive* strategies aimed at preventing future fraudulent activities. This led to our last implicit clue: *in-detailed program analysis could identify the capabilities of contracts, which can then be mapped to corresponding proactive mitigation strategies*. Specifically, as shown on Row 5 of Table 1, we found in total of 3 different capabilities presented in the associated contracts as Evidence ③. Our discovery of a hardcoded recipient in CALL led to the identification of the *Forced Transfer* group, prompting a *reactive* strategy of flagging these accounts and their recipients. Additionally, the detection of CREATE2 opcode usage for deploying asset-transferring contracts highlighted the *Contract Self-Replication* group, enabling FBI agents to proactively flag future deployed contracts. Furthermore, we pinpointed 395,684 contracts with the *Event Emission* capability, using the LOG opcode to generate blockchain event logs. Given the indexable,

FBI agents can actively monitor these logs for early detection of fraudulent activities.

With Evidence ①, Evidence ②, and Evidence ③ generated, we further evaluated the effectiveness of current mitigation effort. Alarming, as indicated by  $\square$  on Row 3 of Table 1, only 11 out of 461,015 associated contracts are flagged on Etherscan [3]. Compounding this issue, we analyzed historical transactions *to* these contracts, assessing the impact of fraud activities by W-521058. This led to the discovery of 232,011 unique victim addresses and an illicit profit of 1,240,355 ETH. These findings highlight the urgent need for forensic analysis focused on DCWs and we collaborating closely with Etherscan [3] and FBI [27] to mitigate the fraud.

### 3. Design

Cybertrack equips the FBI agents with the techniques to investigate the DCW orchestrating the fraudulent activities using smart contracts. Starting with a fraud contract, Cybertrack generates Evidence ① by uncovering the associated contracts from the same DCW along with the recipients they could interact with (§3.1). Subsequently, Cybertrack conducts a recipients provenance analysis on dynamically resolved recipients to produce Evidence ② (§3.2). Lastly, Cybertrack performs *Capability Attribution Analysis* to identify and attribute capabilities to associated contracts, establishing Evidence ③.

#### 3.1. Associated Contracts Recovery

Evidence ① could help FBI agents freeze the assets and flag associated contracts and recipients beyond the confirmed fraud contract. To derive Evidence ①, Cybertrack conducts *Associated Contracts Investigation* and *Recipients Forensic Investigation*.

**3.1.1. Associated Contracts Investigation.** Given a reported fraud contract  $\alpha$ , Cybertrack begins by extracting all historical transactions directed *to* this contract, represented as  $\mathcal{T} = \{t_1, t_2, \dots, t_m\}$ . For instance, when considering the fraud contract C-98f804 detailed in §2.2, Cybertrack records a total of 455 transactions directed to it. Naive FBI agents might assume the first transaction  $t_1$  to be the creation transaction. Unfortunately, as seen in real world [32] and §6.2, there are instances where victims transact with a fraud contract even before its deployment, rendering the first transaction an unreliable indicator. To address this, Cybertrack traverses through  $\mathcal{T}$  and identifies the transaction that *creates* the fraud contract, termed as *creation transaction*, denoted as  $tx_{cr}$ . For the contract C-98f804, Cybertrack identifies  $tx_{cr} = T-8bbae5$ . This identified creation transaction acts as an *explicit* clue, offering the FBI agents a crucial lead to determine the *sender* of the transaction as DCW’s wallet, denoted as  $w$ . Ideally, the agents would then monitor all transactions initiated by the  $w$  and categorize associated contracts as  $\{t_m.to | t_m \in \mathcal{T} \wedge t_m.type == CreateContract\}$ .

---

**Algorithm 1: Identify Associated Contracts**


---

```

Input: DCW  $w$ 
Output: Associated Contracts Set  $\mathcal{C}$ 
1  $\mathcal{C} \leftarrow \emptyset$ 
2 Function GenerateTxnGraph( $txn$ )
3    $G \leftarrow \emptyset$ ;
4   // Iterate over traces in  $txn$  to build graph
5   for  $trace \in txn$  do
6      $from \leftarrow trace.from$ 
7      $to \leftarrow trace.to$ 
8      $edge \leftarrow trace.type$ 
9      $G.add\_edge(from, to, edge = edge)$ 
10  end
11  return  $G$ 
12 end
13 // Extract all transactions From  $w$ 
14  $T_w \leftarrow GetAllTxnFrom(w)$ ;
15 // Generate contract deployment graph Of  $w$ 
16  $G \leftarrow \bigcup_{\forall tx_i \in T_w} GenerateTxnGraph(tx_i)$ 
17 // Update  $\mathcal{C}$  with contracts deployed directly
18  $\mathcal{C} \leftarrow \{edge.to \mid edge \in G \wedge edge.type == CreateContract\}$ 
19 // Identify contracts deployed indirectly
20 for  $s_i \in \mathcal{C}$  do
21    $T_{s_i} \leftarrow GetAllTxnFrom(s_i)$  for  $tx_j \in T_{s_i}$  do
22     // Handle different transaction type
23     switch  $tx_j.type$  do
24       // If contract deploy other contracts
25       case  $CreateContract$  do
26         // Extract deployed contract
27          $s_j \leftarrow tx_j.to$ 
28         // Get all transactions initiated by
29         // the deployed contract
30          $T_{s_j} \leftarrow GetAllTxnFrom(s_j)$ 
31         // Track contract recursively
32          $G \leftarrow G \cup \bigcup_{\forall tx_k \in T_{s_j}} GenerateTxnGraph(tx_k)$ 
33          $\mathcal{C} \leftarrow \mathcal{C} \cup s_i$ 
34       end
35     end
36   end
37 end
38 return  $\mathcal{C}$ 

```

---

Unfortunately, as highlighted in §2.1, W-521058 deployed 65,330 fraud contracts indirectly by invoking contract C-48d304. Since those transactions are simple contract call, tracking only contract creation transactions initiated by DCW could produce inaccurate Evidence ❶ easily. This nuance underscores the importance for FBI agents to discern the *deployment chain*. To overcome this challenge, Cybertrack utilizes the traces of each transaction. This solution is built upon an observation: If an invoked contract deploys a new contract during a transaction, this action will be recorded in the transaction’s traces, as it forms part of the subsequent activities of the invoked contract. Armed with  $tx_{cr}$ , Cybertrack deploys the methodology presented in Algorithm 1. As shown on Line 4 of Algorithm 1, for a given transaction, Cybertrack navigates through traces in that invocation to craft the graph  $G = (\mathcal{E}, \mathcal{V})$ . Here, each node  $e \in \mathcal{E}$  presents an account, and every edge  $v \in \mathcal{V}$  presents a trace, which could be subsequent behaviours (e.g., contract invocation, contract creation). By tracking the trace with

type *CreateContract*, Cybertrack is able to identify the contracts deployed indirectly.

Given the ability to identify contracts deployed both directly and indirectly from transactions, Cybertrack now could identify associated contracts by performing forensic analysis on DCW’s wallet  $w$ . Specifically, Cybertrack starts by tracking all transactions initiated by  $w$  (shown on Line 12 of Algorithm 1), designated as  $T_w$ . Then, for each transaction  $tx_i \in T_w$ , Cybertrack uses GenerateTxnGraph introduced on Line 2 in Algorithm 1 to distill the internal trace graph, subsequently consolidating them to form the transaction graph  $G$  for wallet  $w$ . Essentially,  $G$  encapsulates transaction used by  $w$  to deploy contracts directly and the invocation of the contract. To delve deeper into this chain and disclose contracts spawned via it, as shown on Line 16 of Algorithm 1, Cybertrack gathers traces initiated by each contract  $s_i$ , termed  $T_{s_i}$ . If a trace,  $tx_j$ , results in *CreateContract* (shown on Line 18 of Algorithm 1), Cybertrack first updates  $T_{s_i}$  with the transactions from this newly deployed contracts recursively and then incorporates the deployment result into associated contracts.

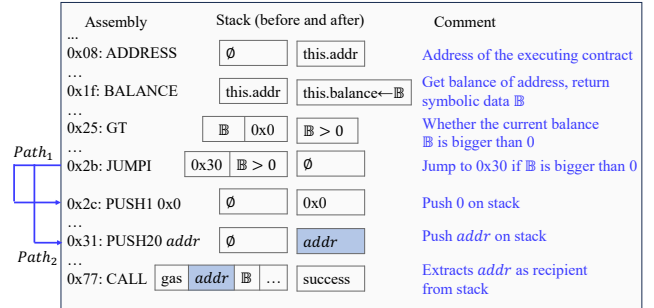


Figure 3: Recipient investigation of C-98f804.

**3.1.2. Recipient Investigation.** Now the only missing piece of Evidence ❶ is the recipients along with the associated contracts identified. This identification enables FBI agents to flag accounts and freeze ETH assets more proactively. However, as illustrated in §2.2, our initial investigation indicates that 36.5% of the fraudulent contracts deployed by W-521058 lack any historical transactions. Relying solely on, or waiting for, these explicit clues (i.e., transactions) could significantly hinder the progress of the investigation.

To overcome this challenge, Cybertrack conducts a transaction-agnostic symbolic analysis on each associated contract. Specifically, Cybertrack designates the input space  $\mathbb{I}$ , storage space  $\mathbb{S}$ , and account balance  $\mathbb{B}$  as symbolic. This symbolic designation allows Cybertrack to conduct a multi-path exploration based on the contract’s logic. Figure 3 shows a segment of the opcode in fraud contract C-98f804, as discussed in §2.2. As shown at address 0x1f in Figure 3, when Cybertrack symbolically executes BALANCE opcode to retrieve balance of contract, Cybertrack marks the returned value as symbolic.

Subsequently, this symbolic value is used by GT opcode shown at address `0x25` to assess if the balance exceeds 0. Upon reaching the JUMPI opcode at address `0x2b` in Figure 3, which performs a conditional jump based on the preceding comparison, Cybertrack forks the execution state into two paths. One path,  $Path_1$  in Figure 3, operates under the condition  $\mathbb{B} > 0$ , while the other,  $Path_2$  in Figure 3, assumes  $\mathbb{B} \leq 0$ , allowing Cybertrack to explore behaviors in both contexts. As depicted at address `0x31` in Figure 3, the fraud contract deploys PUSH20 opcode to push an address  $addr$  onto the stack. However, the specific purpose of  $addr$  remains unclear to Cybertrack. Subsequently, when the CALL opcode is invoked at address `0x77` of Figure 3, Cybertrack retrieves the address  $addr$  from the stack, indicating its use as a recipient address. Notably, smart contract permits the invocation of code from other contracts via DELEGATECALL or CALLCODE. In fraud contracts, such delegation can obscure potential transaction logic in other contracts. To uncover all possible recipients used by contract, Cybertrack recursively delves into contracts invoked with DELEGATECALL or CALLCODE, inheriting constraints from the caller contract. This transaction-agnostic analytical method equips FBI agents with the capabilities to analyze associated contracts and pinpoint potential recipients, independent of historical transaction data.

### 3.2. Recipient Provenance Analysis

Following the derivation of Evidence ①, a subsequent challenge arises that not all contracts employ hardcoded recipients; many use dynamic resolution. As Cybertrack tracks recipients from the stack during multi-path exploration (§3.1.2), dynamically resolved recipients present a challenge, lacking hardcoded addresses on the stack. This situation motivates Evidence ②, which focuses on the recipient’s provenance, including both the resolution logic and the recipient’s origin. With Evidence ②, FBI agents gain the ability to flag accounts that define recipients and monitor recipient changes at upstream, enabling proactive mitigation of fraudulent activities. Cybertrack employs *In Contract Resolution Analysis* and *Cross Contract Resolution Analysis* to ascertain the provenance of dynamically resolved recipients.

**In Contract Resolution Analysis.** Cybertrack determines dynamical resolution when the recipient, identified in Recipient Investigation (§3.1.2), corresponds to a symbolic value  $recp_s$ . Then, Cybertrack conducts a *backward slice* on  $recp_s$  to identify the origin of  $recp_s$ . For instance, in analyzing the fraud contract deployed by C-48d304 (§2.2), Cybertrack uncovers that the recipient on stack is a symbolic value. When Cybertrack performs backward slice on  $recp_s$ , it traces back to opcode AND at address `0x4d`, indicating  $recp_s$  is from a logical AND operation. Moving further back, Cybertrack encounters another AND operation at address `0x37`. Continuing this trace, Cybertrack reaches an opcode DIV at address `0x21`. This division operation implies that  $recp_s$  is depend on a division calculation. The

backward slice eventually leads Cybertrack to address `0x1a` with SLOAD operation. At that moment, Cybertrack proves that  $recp_s$  originates from the contract’s storage. By examining the parameter passed to SLOAD, a constant `0x00`, Cybertrack concludes that the recipient address is located in storage location  $\mathbb{S}[0]$ .

**Cross Contract Resolution Analysis.** The previous analysis only reveals where the fraud contract loads the recipient from. However, there is no evidence of which account defines the recipient address (i.e., the value stored in  $\mathbb{S}[0]$ ). To address this and draw a full picture of Evidence ②, Cybertrack conducts Cross Contract Resolution Analysis. Notably, smart contracts rely on the CREATE or CREATE2 opcodes to deploy other contracts. Executing these opcodes results in a contract deployment transaction, and eventually, both opcodes need init code as input. Motivated by this, once Cybertrack determines the in-contract origin of a dynamically resolved recipient, such as  $\mathbb{S}[0]$ , it identifies the contract’s deployment transaction (as discussed in §3.1.1). From this transaction, Cybertrack extracts the init code used in the contract’s creation. It then conducts symbolic analysis on this init code to track changes to the in-contract recipient origin. For instance, in the case of associated contract C-99bcb3, Cybertrack pinpoints its deployment transaction with hash T-d9d53e, initiated by W-521058 via invoking C-48d304. By analyzing the direct creator contract, Cybertrack extracts the init code from parameter passed to CREATE2. Subsequent symbolic analysis on this init code reveals the use of SSTORE opcode, pointing to key 0 and assigning a hardcoded address as value. Consequently, Cybertrack attributes the parent contract as the origin of the recipient.

### 3.3. Capability Attribution Analysis

Evidence ③ attributes specific capabilities to contracts associated with a confirmed fraud. These capabilities guide FBI agents in implementing corresponding mitigation strategies for ongoing and future fraud prevention. Cybertrack uses symbolic data, introduced in §3.1.2, effectively highlights the information flow, to track the capabilities of associated contracts. An example in §3.2 demonstrates this: when Cybertrack detects symbolic data propagation from SLOAD to the recipient in a CALL opcode, it’s interpreted as *Dynamic Recipient Resolution*, triggering the *Monitor* mitigation strategy. The 10 capabilities identified by Cybertrack, along with semantic models and mitigation strategies, are outlined in Table 2. As seen on Column 4 of Table 2, Cybertrack proposes three different mitigation strategies. The function  $F(\theta)$  represents flagging and freezing the account denoted by  $\theta$ .  $S(\theta)$  means scanning the blockchain for contracts sharing the same code as  $\theta$  if  $\theta$  is a contract address, or for identical data to  $\theta$  if  $\theta$  represents raw byte content.  $M(\theta)$  indicates monitoring the account specified by  $\theta$ . Notably, Cybertrack’s approach to symbolic analysis, independent of semantic model knowledge, requires only a one-time effort and offers ease of extension.

TABLE 2: FRAUD CONTRACT CAPABILITIES, OPCODE, SEMANTIC MODELS, AND MITIGATION STRATEGIES.

Fraud Capability	Opcode	Semantic Models	Mitigation <sup>1</sup>
Forced Transfer	CALL	$\neg \text{Symb}(\text{Stack}[1]) \wedge \text{Address}(\text{Stack}[1]) \wedge \text{Stack}[2] > 0$	$F(\text{Stack}[1])$
User-Defined Transfer	CALL	$\text{Symb}(\text{Stack}[1]) \wedge \text{Stack}[1] \in \{\text{CALLDATA}, \text{MLOAD}\} \wedge \text{Address}(\text{Stack}[1]) \wedge \text{Stack}[2] > 0$	$F(\text{Stack}[1])$
Recipient Resolution	CALL	$\text{Symb}(\text{Stack}[1]) \wedge \text{Stack}[1] \in \text{SSLOAD} \wedge \text{Stack}[2] > 0$	$F(\text{Stack}[1])$ $M(\text{SSLOAD})$
Address Allocation	SSTORE	$\neg \text{Symb}(\text{Stack}[1]) \wedge \text{Address}(\text{Stack}[1]) \wedge \exists k. (\text{SLOAD}(k) == \text{Stack}[1]) \wedge (\text{CALL}(k) \vee \text{CALLDATA}(k))$	$F(\text{Stack}[1])$ $M(\text{this})$
Contract Self-Replication	CREATE CREATE2	$\neg \text{Symb}(\text{Stack}[0]) \wedge \text{Code}(\text{Stack}[0])$	$F(\text{return})$ $S(\text{Stack}[0])$
Dynamic Contract Deployment	CREATE CREATE2	$\text{Symb}(\text{Stack}[0]) \wedge \text{Stack}[0] \in \text{SSLOAD} \wedge \text{Code}(\text{Stack}[0])$	$F(\text{return})$ $S(\text{Stack}[0])$ $M(\text{SSLOAD})$
Predefined Contract Redirection	DELEGATECALL CALLCODE	$\neg \text{Symb}(\text{Stack}[0]) \wedge \text{Address}(\text{Stack}[0])$	$F(\text{Stack}[0])$ $S(\text{Stack}[0])$
Dynamic Execution Delegation	DELEGATECALL CALLCODE	$\text{Symb}(\text{Stack}[0]) \wedge \text{Stack}[0] \in \{\text{SSLOAD}\}$	$F(\text{Stack}[0])$ $M(\text{this})$ $S(\text{Stack}[0])$
Address Forwarding Strategy	SSTORE	$\neg \text{Symb}(\text{Stack}[1]) \wedge \text{Address}(\text{Stack}[1]) \wedge \exists k. (\text{SLOAD}(k) == \text{Stack}[1]) \wedge (\text{DELEGATECALL}(k) \vee \text{CALLCODE}(k))$	$F(\text{Stack}[1])$ $M(\text{this})$ $S(\text{Stack}[1])$
Fraud Event Logging	LOGX	$(\text{Stack}[0] \neq 0 \wedge \text{Stack}[1] \neq 0) \vee (\text{Stack}[2], \dots, \text{Stack}[X-1]) \neq 0$	$S(\text{Stack}[2 : X-1])$ $S(\text{Mem}[\text{Stack}[0] : \text{Stack}[1]])$

1:  $F$  flags and freezes the specified account;  $S$  scan the content across on-chain data;  $M$  monitors the designated account.

**1. Forced Transfer.** When the opcode is CALL and the first element on the stack is not symbolic (hardcoded) and the second element is greater than zero, this indicates a Forced Transfer. This capability involves transferring ETH to a hardcoded recipient address, suggesting a transfer to a specific address. Identifying and reporting these hardcoded recipients allows FBI agents to proactively flag and freeze the extracted recipient accounts.

**2. User-Defined Transfer.** This occurs when the CALL opcode is used with a symbolic recipient, which is specified by the user. The recipient address is usually derived from user input, typically through CALLDATA or MLOAD. Reporting this capability allows FBI agents to proactively flag recipients specified in the user transaction.

**3. Dynamical Recipient Resolution.** In this scenario, the CALL opcode uses a symbolic first stack element where the recipient address is dynamically determined. This is identified when the first stack element depends on the return of SLOAD. This dynamic resolution allows DCW to alter the recipient address. Reporting this capability allows FBI agents not only to flag the current recipient address, but also monitor the place that contract load recipient address to detect recipients change before fraud occur.

**4. Address Allocation.** The SSTORE opcode, coupled with a non-symbolic first stack element that is an address of account, indicates Address Allocation capability. Here, a specific address is hardcoded into storage and subsequently used in transaction operations, which can be identified if SLOAD with a particular key equals the non-symbolic data

identified previously and is followed by CALL or CALLDATA operations. Reporting this allows FBI agents to proactively flag the account and monitor the transactions of this contract since it could update the address.

**5. Contract Self-Replication.** This capability is identified when either the CREATE or CREATE2 opcode is used with a non-symbolic first element on stack. It indicates Contract Self-Replication, where a contract replicates itself using a predefined template. This is often used in fraudulent activities to propagate fraud contracts.

**6. Dynamic Contract Deployment.** Identified by the use of CREATE or CREATE2 opcodes with a symbolic first element on stack, this capability suggests Dynamic Contract Deployment. It is marked by the dynamic deployment of new contracts, which can vary in nature and functionality based on the input or state, allowing for varied fraudulent activities.

**7. Predefined Contract Redirection.** This occurs when either the DELEGATECALL or CALLCODE opcode is used with a non-symbolic first element on stack. It indicates a redirection of execution to another contract with a hardcoded address, typically for executing specific fraudulent actions predefined in the target contract.

**8. Dynamic Execution Delegation.** Identified by the use of DELEGATECALL or CALLCODE opcodes with a symbolic first element on stack, this capability suggests the delegation of execution to different contracts based on dynamic conditions or inputs. It allows a contract to adapt its execution and fraud strategy based on real-time data or states.

**9. Address Forwarding Strategy.** This is observed when the SSTORE opcode is used with a non-symbolic second element on stack that points to an account, and there exists a key such that SLOAD with this key equals the value previously stored. The loaded value would be passed to DELEGATECALL or CALLCODE. It indicates a strategic storing of an address for subsequent use in delegation operations, implying a planned redirection of execution.

**10. Fraud Event Logging.** This is observed when Cybertrack encounters the LOGX opcode ( $X \in [0, 4]$ ), which is used to log events. This capability suggests that the contract logs events, which could be used to track the contract’s activities and identify fraudulent transactions.

Cybertrack is now equipped with the technique to produce Evidence ③ by attributing capabilities to each associated contract and proposing corresponding mitigation strategies. This categorization enables FBI agents to implement bulk mitigation actions effectively.

## 4. Validation

We have implemented prototype of Cybertrack in Python leveraging Mythril [33], with customized module (10K lines) to perform program analysis, and on-chain transaction analysis (10K lines). We conducted our experiments on an Ubuntu 20.04 LTS system equipped with 12 CPUs and 64GB of memory. For validation purposes, we randomly selected fraud contracts from our dataset until we found 12 ones with different DCWs. The distribution of ground truth dataset aligns with the overall distribution of DCW and the number of fraud contracts they deployed, shown in Figure 4 in §5.1. Notably, since contract flagging serves as a user warning and removing these accounts from the blockchain is almost impossible, these flagged contracts might still be active. To ensure the reproducibility of our study, we based our analysis on the Ethereum state as of June 26, 2023. Operating within this state, we derived the ground truth by conducting a manual investigation on each fraud contract in validation dataset.

### 4.1. Associated Contracts & Recipients

Table 3 presents Cybertrack’s validation result. As shown in False Positive (FP) and False Negative (FN) columns, Cybertrack has generated 7 FPs and 1 FN. To prevent overstating performance, we assign 1 TP in both Associated Contracts (Columns 3-4 of Table 3) and Deploy Length (Columns 7-8 of Table 3) if Cybertrack correctly identifies all associated contracts and deployment lengths, respectively, otherwise assigning 0 in each case. Given this, we can see that Cybertrack achieves an overall accuracy of 90.91% , demonstrating its efficiency in generating precise forensic evidence. Delving further into the performance of Cybertrack, each evidence (Evidence ① , Evidence ② , and Evidence ③ ) is evaluated. The first two columns of Table 3 list the abbreviation of each reported fraud contract processed by Cybertrack and the

uncovered DCW, respectively. Interested readers could find the full address of the abbreviation in Table 6. Columns 3-6 show the validation results for Evidence ① .

As shown in Total row, Cybertrack pinpointed 1,050,705 associated contracts, along with 17 recipients. Our ground truth data indicates the presence of 1,050,703 associated contracts with 17 recipients. Upon a thorough investigation, we found that 2 FPs can be attributed to the distinct methods employed by DCWs (Row 6 and Row 12 of Table 3) in deploying the fraud contracts. Specifically, Cybertrack discovered that W-3CD202 and W-5FFDdc invoked contract C-36FFaE and C-1e3e4e to introduce new fraud contracts respectively. In the course of analyzing C-36FFaE, Cybertrack identified that, rather than employing the standard CREATE or CREATE2 operations for contract deployment, C-36FFaE instead delegated this task to an alternative contract, C-2C4691. Consequently, Cybertrack flagged C-2C4691 as associated contracts. However, a manual review of C-1e3e4e and C-2C4691 revealed that they were in fact deployed by different accounts, W-53C9dA and W-13666c respectively. Considering the FBI agent in this case would have no obvious evidence of the fraud intention, we consider this as FP. Notably, Cybertrack still reported these two contracts since they give FBI agents an important lead of potential contract abuse. We confirmed these are rare cases in our dataset. Overall, Cybertrack was 90.91% accurate in generating evidence, making it robust for our evaluation.

### 4.2. Dynamically Resolved Recipients

Columns 7-10 of Table 3 presents the validation result in Cybertrack generating Evidence ② . As illustrated in Column 8 and Column 10 of Table 3, Cybertrack has identified 3,162,963 Deploy Length (number of accounts involved in the deployment of a contract) and 17 origins of recipients. As shown on Row 6 and Row 12 of Table 3, Cybertrack generated FP when analyzing DCW W-3CD202 and W-5FFDdc. This FP is the direct result of the FP in Evidence ① generation, as discussed in §4.1. We also see 1 False Negative (FN) shown on Row 6 of Table 3 (W-3CD202) during the track of recipient’s origin. As discussed in §4.1, when W-3CD202 invokes the intermediate contract C-36FFaE, it delegates the responsibility of deploying fraud contracts to C-2C4691. By manually dissecting the implementation of C-36FFaE, we found that the function it will trigger in C-2C4691 is not hard-coded. Instead, it is resolved based on the transaction invoking C-36FFaE. Consequently, the absence of a static entry point hindered Cybertrack’s capacity to pinpoint the contract deployment process used by the DCW, W-3CD202, thereby impeding the tracing of recipient origins in contracts deployed by C-2C4691. Nevertheless, as discussed earlier, such instances are rare cases. 90.91% accurate in generating evidence makes Cybertrack robust for our evaluation.



TABLE 3: CYBERTRACK’S VALIDATION.

Contract	DCW	Evidence ①				Evidence ②				Evidence ③		FP	FN
		Associated Contracts		Recipients		Deploy Length <sup>2</sup>		Origin		GT	C		
		GT	C <sup>1</sup>	GT	C	GT	C	GT	C				
C-98f804	W-521058	461,015	461,015	1	1	987,360	987,360	1	1	2	2	0	0
C-6113fB	W-bceE76	1	1	1	1	2	2	1	1	1	1	0	0
C-Fd562c	W-82f98B	3	3	1	1	6	6	1	1	1	1	0	0
C-2e14CC	W-2499E7	3	3	1	1	6	6	1	1	1	1	0	0
C-8f428e	W-b10f4B	2	2	1	1	4	4	1	1	1	1	0	0
C-18B228	W-3CD202	406,559	406,560	1	1	1,626,234	1,626,236	1	0	1	2	4	1
C-805a29	W-6629BA	2	2	1	1	4	4	1	1	1	1	0	0
C-078228	W-2b0878	2	2	1	1	4	4	1	1	1	1	0	0
C-0E712A	W-09739F	1	1	3	3	2	2	3	3	1	1	0	0
C-db7aFF	W-187307	3	3	3	3	7	7	3	3	3	3	0	0
C-70aae3	W-1AB876	4	4	2	2	10	10	2	2	4	4	0	0
C-e18895	W-5FFDdc	183,108	183,109	1	1	549,324	549,326	1	1	1	1	3	0
<b>Total</b>	12	1,050,703	1,050,705	17	17	3,162,963	3,162,967	17	16	18	19	7	1

1: C is short for Cybertrack.  
 2: Deploy Length refers to the number of accounts involved in the deployment of a contract.  
 This column shows sum of Deploy Length of all associated contracts.

### 4.3. Capability Attribution

Columns 11-12 illustrate the performance of Cybertrack regarding to the generation of Evidence ③. As presented in Total row, Cybertrack has successfully grouped 18 associated contracts into attributed capabilities from the 12 fraud contracts. Nonetheless, our verified ground truth indicates 19 such groups. Further scrutiny, as discussed in relation to the earlier FP incident detailed in §4.1, reveals that during the analysis of the transactions originating from the fraud contract C-36FFaE (show on Row 6 of Table 3) Cybertrack included contract C-2C4691 because it was designated by C-36FFaE to carry out the deployment of fraud contracts. Since C-2C4691 operates as a multi-signature wallet contract, Cybertrack mis-classified it as a separate attributed group. It’s important to note that the FP in Evidence ③ generation is intrinsically linked to the FP in Evidence ① generation. The FBI agents, by identifying and excluding C-2C4691 from the forensic analysis of DCW, can eliminate this error in the generation of Evidence ③ as well. Considering the minimal occurrence of FPs and FNs, coupled with an accuracy rate of 90.91% , Cybertrack is validated as an effective tool for generating the three pieces of evidence of DCW that FBI agents require to mitigate the fraudulent activities.

## 5. Evaluation

In this section, we will demonstrate the effectiveness of Cybertrack in uncovering the impact of DCWs. Specifically, We applied Cybertrack to analyze 157 verified fraud contracts using labeled contracts from Etherscan [3], to emulate the investigative processes faced by FBI agents.

### 5.1. Post Deployment Dataset Highlights

Deploying Cybertrack on our dataset revealed an unnerving trend in fraudulent activities perpetrated through smart contracts. Given 157 flagged smart contracts on Etherscan [3], Cybertrack uncovered a total of 1,283,198 associated contracts from 91 behind these contracts. The distribution of these associated contracts is presented in Figure 4. Furthermore, as highlighted under Profit (ETH) in Figure 4, Cybertrack assessed the impact of these fraudulent activities by calculating the ETH equivalent of the illicit profits garnered by DCWs. Specifically, Cybertrack determined that DCWs amassed a total of 2,638,752 ETH in illicit profits, averaging 2.06 ETH per contract. Interestingly, Figure 4 also reveals a strong correlation between the number of associated contracts and the profits accruing to DCWs. This suggests that rather than relying on a small number of contracts, DCWs are more inclined to distribute risk across various fraud contracts. The tactics of DCWs reflect the effectiveness of current mitigation strategies, such as flagging, in influencing DCWs’ operations.

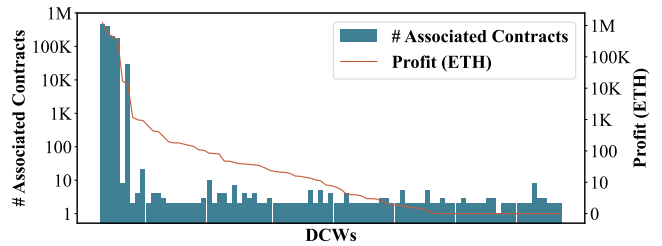








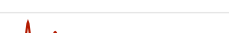



Figure 4: Distribution of associated contracts and illicit ETH profits across DCWs.

TABLE 4: TOP 10 DCWS THAT MADE MOST ILLICIT PROFIT THROUGH FRAUD CONTRACTS.

DCW	Start Time	End Time	Fraud Transaction Volume	ETH	USD	Victim				
						Min	Max	Avg	Total	w Cybertrack
W-521058	Oct 2017	Jun 2023		1,240,355	1,384,080,119	1	2,978	471	232,011	209,926%
W-3cd202	Feb 2018	Jun 2023		280,053	192,665,735	1	2,842	303	173,469	113,519%
W-3a3250	Sep 2017	Jun 2023		447,891	141,686,276	2	5,756	295	74,074	116,699%
W-4b2c9b	Nov 2017	Jun 2023		13,605	9,639,322	1	99	16	5,833	5,382%
W-9989f8	Apr 2018	Jul 2018		16,706	7,976,355	1	5	0	43	1%
W-ec1ef1	Aug 2022	Feb 2023		976	1,374,083	1	714	67	14,489	0%
W-82f98b	Apr 2022	May 2022		416	1,224,971	1	14	5	93	1%
W-e058d6	Dec 2020	Feb 2021		931	1,138,740	1	11	2	130	17%
W-2499e7	May 2022	May 2022		432	901,275	1	8	4	51	2%
W-f09117	Sep 2017	Sep 2017		1,163	394,444	1	19	3	26	0%
<b>Summary</b>	Sep 2017	Jun 2023		2,002,533	1,741,081,326	1	5,756	232	493,068	445,556%

## 5.2. Impact Of DCW

Table 4 presents the top 10 DCWs with the highest fraud profit that have been identified by Cybertrack. Column 1 shows the abbreviation of each DCW. Columns 2 and 3 display the start and end dates of its fraudulent activities, respectively. Column 4 details the number of fraudulent transactions associated with each DCW outlined in Column 1, arranged temporally. Column 5 indicates the aggregate quantity of ETH illicitly appropriated by each DCW. Considering the objective of DCW is to generate real-world currency profits, Cybertrack converts the stolen ETH into USD, based on the exchange rate at the time of each transaction; this converted amount is reflected in Column 6. Beyond assessing the economic impact to indicate the extent of the fraudulent activities, FBI agents should also consider its severity based on the number of affected individuals. In support of this, Cybertrack identifies the associated contracts and their corresponding transactions, subsequently determining the *origin* of each transaction. Such origins are used as the estimation of victim accounts. Columns 7 through 9 show the minimum, maximum, and average number of *distinct* victims engaged with DCW detailed in Column 1 on a daily basis. Column 10 provides a total victim count. To underscore the significance of Cybertrack, we compared the efficacy in victim identification by FBI agents solely from reported fraud contracts against the enhanced forensics capabilities equipped by using Cybertrack.

Table 4 provides insights into the operations of fraudulent activities conducted via smart contracts. Column 2 reveals that, out of 10 analyzed by Cybertrack, the inception of such fraudulent activities dates back 5 years ago to Sep 2017, which is merely 2 years subsequent to Ethereum’s introduction [34]. This highlights the long-standing presence of fraudulent undertakings within the Ethereum network. It was anticipated that this prolonged history of fraudulent acts

would have garnered the attention of law enforcement and government agencies, prompting them to initiate countermeasures. Unfortunately, when we examine Column 3 of Table 4, it becomes apparent that 4 out of 10 (40.00%) remain active up to the date of this study (shown as Jun 2023 in Column 3 of Table 4). For example, Row 1 of Table 4 illustrates that the fraud led by W-521058 has persisted for 5 years. Considering the observed efforts by government agencies (e.g., the FBI) to combat general cryptocurrency fraud [19], it is clear that the stealth and complexity of smart contract-based fraud have out paced the investigative capabilities of legal authorities.

The enduring presence of DCW, coupled with inadequate investigative approaches, has led to the rampant of these fraudulent activities. An examination of Column 4 of Table 4, which details the daily transaction volume of DCW listed in Column 1, yields several notable observations: (1) As evidenced by Rows 1-4 of Table 4, substantial DCW do not opt for discretion; instead, they exhibit consistently high volumes of fraudulent transactions on a daily basis. (2) Although the transaction volume for less aggressive DCWs may vary, as depicted by Rows 5-10 of Table 4, their relatively brief duration of activity still demonstrates a significant level of traffic.

The high volume of fraudulent transactions directly correlates with the substantial profits these DCWs yield. Columns 5 and 6 of Table 4 demonstrate that the foremost 10 DCWs have collectively garnered 2,002,533 ETH (\$1,741,081,326) in illicit profits. In reviewing Row 1 of Table 4, it becomes evident that the most profitable scheme, conducted by W-521058, has amassed 1,240,355 ETH (\$1,384,080,119) in unlawful earnings, accounting for 61.94% of the total ETH and 79.50% of the total USD accumulated by these 10 leading DCW.

Till now, it is clear that fraudulent activities facilitated by smart contracts represent a critical issue, especially considering the substantial illicit profits they have accrued.

TABLE 5: TOP 15 FRAUD CONTRACTS IDENTIFIED BY CYBERTRACK THAT MAKE HIGHEST ILLICIT PROFITS.

Contract	Start Time	End Time	Daily Revenue	ETH				USD			
				Min	Max	Avg	Total	Min	Max	Avg	Total
C-cd154b	Oct 2017	Jun 2023		0.005	8,364	147	304,772	8.690	5,847,875	99,874	206,240,342
C-cc1d68	Feb 2018	Jun 2023		0.060	1,238	53	105,061	52.057	2,402,960	49,762	97,036,980
C-2f9ffe	Oct 2019	Jun 2023		0.004	666	42	57,385	7.383	1,029,244	60,650	81,635,779
C-e00b88	May 2021	Jun 2023		0.040	609	37	28,453	99.679	1,267,433	90,000	68,040,043
C-e2985f	May 2021	Nov 2022		0.010	14,344	47	25,599	29.309	23,519,179	84,345	45,546,477
C-f03cef	Mar 2022	Jun 2023		1.864	613	54	25,596	2236.746	955,632	82,739	39,053,042
C-6f5365	Dec 2019	Jun 2023		4.060	559	15	19,762	593.342	716,962	15,858	20,346,288
C-cedd45	Nov 2021	Jun 2023		0.095	397	32	19,610	410.665	1,161,698	64,659	38,795,418
C-5e0679	Dec 2017	Nov 2022		17.958	450	10	18,689	10,977.451	756,184	11,387	20,543,033
C-3c2551	Nov 2017	Nov 2022		0.630	192	9	16,629	640.500	606,352	22,067	40,074,910
C-af45d2	Nov 2022	Nov 2022		14,592	14,592	14,592	14,592	17,733,852	17,733,852	17,733,852	17,733,852
C-3c9b11	May 2021	Jun 2023		4.992	199	13	10,420	9,148.109	690,314	27,946	21,798,422
C-49c546	Apr 2021	Dec 2022		0.545	268	15	9,697	2,421.490	675,311	41,793	25,828,472
C-47961f	Oct 2021	Jun 2023		1.000	500	14	8,627	3,971.950	937,348	27,819	16,858,621
C-b65ed7	Nov 2017	Jun 2023		0.295	232	4	8,322	94.204	177,506	3,027	6,211,922
<b>Summary</b>	Oct 2017	Jun 2023		0.004	14,592	325	673,221	7.383	23,519,179	360,436	745,743,608

Alarming, the situation may escalate when considering the number of victims impacted by these DCWs. Columns 9-10 of Table 4 detail the minimum, maximum, and average number of daily victims for the top 10 DCWs. A closer look at Column 10 of Table 4 reveals that the total victim count ensnared by these fraudulent operations could stand at 493,068. Focusing on Row 1 of Table 4, we find that the fraudulent activities conducted by W-521058 has affected 232,011 individuals, which represents 47.05% of all identified victims. When comparing Columns 5-6 with Column 10, it becomes apparent that, on average, each victim has lost 4.06 ETH (\$3531.12) to these DCWs. We have thus demonstrated that fraudulent activities executed via smart contracts are a great concern, both in terms of the illicit profits and the number of victims affected. To further underscore the need for more proactive forensic analysis, Column 11 of Table 4 shows the potential increase in identified victims by using Cybertrack. Specifically, the difference between victims from reported contracts and the total number involved in the fraudulent activities. The Summary row highlights an average gain of 445,556% additional victims, reinforcing the importance of integrating Cybertrack into DCW fraud forensics.

### 5.3. Drill Down Into Fraud Contracts

We have established that the illicit profits and the number of victims involved in the fraudulent activities can easily reach \$1,741,081,326 and 493,068, respectively, as detailed in §5.2. However, it remains unclear that on the finer granularity, whether the smart contracts driving these

frauds exhibit similar transaction pattern (e.g., volume) or profit margin analogous. To address this gap, we delve deeper into the contracts in the following section. Table 5 lists the top 15 fraudulent contracts, ranked by the highest illicit profits, as identified by Cybertrack. Column 1 displays the abbreviation for each contract, and the readers interested in the full address can refer to Table 6. Columns 2-3 outline the start time and end time of the fraudulent activities for each contract, respectively. Column 4 presents the daily fraudulent transaction volume timeline for contract in Column 1. Columns 5-7 show the daily illicit ETH profits for each contract in Column 1, formatted as Minimum (Min), Maximum (Max), and Average (Avg). Column 8 tallies the total amount of ETH each contract has accumulated from the transactions of victims. To further quantify the impact of these fraud contracts, Cybertrack has converted the ETH amounts into USD based on the historical exchange rates at the time of each transaction. The corresponding USD profits are exhibited in Columns 9-12. Specifically, Columns 9-11 detail the daily USD profits, while Column 12 aggregates the total USD revenue accrued by each contract throughout its period of activity.

Table 5 provides interesting insights into the fraud contracts that direct fraudulent activities orchestrated by DCWs. As discussed in §5.2, we observed that fraudulent activities tend to have an extended duration of activity. Meanwhile, an examination of Columns 2-3 in Table 5 indicates that 14 out of 15 contracts (93.33%) have sustained fraudulent activities for over a year. This

observation could be attributed to the immutable characteristic of blockchain technology. Specifically, once fraud contracts are exposed, their perpetrators might not be concerned about the contracts being terminated. Intuitively, one might assume that these fraud contracts would exhibit a consistently high volume of fraudulent activity since the orchestrators do not have to worry about the contracts being dismantled. Surprisingly, Column 3 of [Table 5](#) reveals that the contracts on Rows 4, 8-10, 14-15 of [Table 5](#) (40.00% of the top 15 contracts) undergo a significant period of dormancy before experiencing a resurgence of fraudulent activities. This pattern suggests that, regardless of their lack of concern over contract removal, orchestrators still take measures to prevent the fraud contracts from being detected.

Upon reviewing Columns 8 and 12 of [Table 5](#), it becomes evident that the top 15 contracts have collectively accumulated illicit gains of 673,221 ETH (\$745,743,608). It is important to note that the USD profits are calculated using the historical ETH-to-USD exchange rate corresponding to the dates of the fraudulent transactions. Considering the extended duration of the fraudulent activities and the rising value of ETH, it is conceivable that the perpetrators might realize even greater profits if they were to liquidate their ETH holdings at current market prices. Delving deeper into the profit analysis, a focus on Columns 7 and 11 of [Table 5](#), which outline the average daily ETH and USD profits generated by the contracts listed in Column 1, reveals that periods of dormancy have not impeded the substantial accumulation of profits by the orchestrators. Referencing [Table 5](#), it is evident that with the aid of Cybertrack, FBI agents can ascertain that the average daily profits of the contracts amount to 325 ETH and \$360,436. These figures constitute 2.23% and 1.53% of the maximum daily profits, respectively. The data highlighted here underscore the orchestrators' proficiency in sustaining a regular flow of illicit revenue, navigating even through sporadic phases of dormancy. This revelation amplifies the gravity of fraudulent activities facilitated by smart contracts and underscores the urgency to deploy Cybertrack for comprehensive forensic analysis.

## 6. Case Studies

### 6.1. Case Study 1: Contracts Shared By DCWs

Instead of writing customized fraud contracts for themselves, deploying Cybertrack on the dataset surprisingly revealed contract reuse across different DCWs. Specifically, Cybertrack identified a total of 3 different bytecodes used by 7 DCWs, involving 16 contracts. The naive FBI agents would assume that contracts sharing the same bytecode would execute same transactions. This assumption would be valid if these contracts shared the *Forced Transfer* capability, which means that hardcoded recipients are in the code. Interestingly, Cybertrack reported that these 16 contracts are all equipped with the

*Dynamic Recipient Resolution* capability. For example, Cybertrack observed that the contracts could execute SLOAD to load an address from the 6-th storage slot. Then, the loaded result would be utilized as the recipient in the fraudulent transactions. By performing *Recipient Provenance Analysis*, as detailed in [§3.2](#), Cybertrack discovered that the 6-th storage slot consistently hosted the address of the *creator*. Consequently, even though these 16 contracts share 3 unique bytecodes, they still engaged in different transactions.

### 6.2. Case Study 2: Abuse Of CREATE2

Beyond merely scrutinizing the overarching fraudulent activities orchestrated by DCWs, deploying Cybertrack also enables a detailed examination of their technical evolution. During the analysis of W-521058, who has collected around 1,250,355 illicit ETH profit, Cybertrack observed an evolution from manually fraud contract deployment to an automated fraud contract deployment given C-48d304 with *Contract Self-Replication*. Compounding this issue, during the analysis of C-48d304, Cybertrack discovered that instead of using the CREATE opcode, the DCW employed CREATE2. The CREATE opcode computes the address of the deployed contract using Keccak256 [28] on *sender* and *nonce* where *sender* is the address of sender and *nonce* represents the number of transactions originating from the sender's address. Although CREATE's functionality is deterministic, predicting the address of the deployed contract remains a challenge since it relies on the on-chain data, *nonce*. In contrast, instead of using *nonce*, the CREATE2 opcode facilitates the creation of contracts with a deterministic address by using *salt*, which could be arbitrarily defined in the transaction. This feature allows DCWs to predict the address of the yet-to-be-deployed fraud contract without even accessing the blockchain. Consequently, DCWs could lure victims into transacting with a 'non-existent' fraud contract [32]. At such a juncture, discerning the malicious nature of the contract becomes exceedingly challenging for the victims, as it is yet to be deployed. To substantiate this hypothesis, we observed that all 65,530 fraud contracts had victim transactions occurred prior to the actual deployment of the fraud contract.

## 7. Related Work

### 7.1. Fraud Detection On Blockchain

**Ponzi Scheme.** Yu et al. [4] utilized graph convolutional networks to detect Ponzi schemes using transactions, whereas Zhang et al. [11] improved on the LightGBM algorithm for better detection efficacy. Extracting bytecode features [12] and leveraging text convolutional neural networks [13] have also been proposed to identify Ponzi-like characteristics in smart contracts. There are also works using opcode compression for vulnerability

detection [14], and identifying static features within smart contracts for Ponzi scheme classification [15]. Techniques using opcode sequences [16], code analysis [17], and enhanced convolutional neural networks [5], [18] have also been implemented to identify Ponzi scheme.

**Fraud In General.** Beyond Ponzi schemes, the broader fraud detection on blockchain is tackled through various innovative approaches. Liu et al. [7] utilized a Heterogeneous Information Network to model smart contracts and apply graph Transformer networks to detect abnormalities. Machine learning algorithms, such as XGBoost and Random Forest, have been employed by Ashfaq et al. [8] to classify Ethereum transactions and detect anomalies like double-spending and Sybil attacks. Furthermore, Hu et al. [9] explored deep learning models to identify scams on a large scale by examining the n-gram features of byte. Notably, the identification of fraud contracts acts as the input for Cybertrack, enabling it to generate evidence that assists FBI agents in obtaining legal authorization and implementing mitigation strategies.

## 7.2. Smart Contract Vulnerability

Luu et al. [20] were among the first to address the security risks in smart contracts by identifying security pitfalls. Symbolic analysis has been frequently used for detecting such vulnerabilities; Krupp and Rossow [21] developed techniques for automatically exploiting smart contracts by analyzing bytecode. Similarly, Zeus [22], a framework for the symbolic verification of smart contracts, uses Constrained Horn Clauses to identify vulnerabilities, an approach also adopted by others [23], [25]. With the rising complexity of smart contracts, especially those involving multiple contracts, Ma et al. [24] proposed an inter-contract analysis tool, while others have focused on state inconsistency bugs [26]. Online detection methods have also been investigated, such as SODA [35], an online detection framework for smart contracts.

Recent works have shifted towards leveraging machine learning to enhance the detection process. Sendner et al. [36] introduced ESCORT, a method employing deep learning to identify different types of smart contract vulnerabilities. There are also works utilizing dynamic analysis [37], deep learning [38], and pre-training techniques [39] to identify the smart contract vulnerabilities. Others have focused on static analysis rules tailored to specific blockchain platforms, like VRust for Solana smart contracts [40], or on defining critical paths to address fund transfer vulnerabilities [41]. Moreover, combining expert knowledge with graph neural networks has shown promise in enhancing detection capabilities [42]. Another notable approach is the analysis of confused deputy vulnerabilities in Ethereum smart contracts [43], which highlights the importance of understanding security challenges in contract interactions. Contributing to a different domain, Cybertrack conducts forensic analysis on fraud contracts, producing the evidence for FBI agents to

obtain legal authorization and execute mitigation strategies.

## 8. Discussion

**Ethical Concerns.** In this paper, we address the ethical considerations associated with the forensic analysis executed by Cybertrack. The data utilized in this analysis is sourced exclusively from the public domain, specifically Ethereum. Subsequent to the forensic investigation, remedial actions are undertaken through a concerted effort with service providers (for instance, Etherscan [3]) and relevant governmental agencies. Throughout the process of remediation, we adhere to the established guidelines and the terms of service stipulated by these service providers as well as government agencies. We *do not* attempt to take any exploitative actions on our own.

**Challenges of Symbolic Analysis.** Cybertrack, leverages symbolic analysis, a technique adopted by top-tier research [44]–[48]. While symbolic analysis often faces the challenge of path explosion, our evaluation in §5 reveals this to be a rare issue in smart contract analysis. This is likely because smart contract execution involves a cost, known as ‘gas’, leading malicious actors to prefer simpler contract designs.

**ERC-20 & NFT.** Cybertrack primarily focuses on the forensic analysis of fraudulent activities involving the scam of ETH. ERC-20 [49] and Non-Fungible Tokens (NFTs) [50] have emerged as extensions built upon the Ethereum. ERC-20 tokens provide a standardized protocol for fungible tokens, ensuring consistency in token interactions. Non-Fungible Tokens (NFTs), in contrast, represent unique digital assets, each characterized by distinct metadata and individual ownership records. Even though both token standards are implemented through smart contract, forensic analysis of ERC-20 and NFT requires distinct approaches. In particular, both transaction and program analysis offer limited insight into the complex behaviors associated with these tokens. While ETH transfers are directly recorded as transactions, exchanges of ERC-20 tokens or activities like NFT minting and ownership transfer are typically reflected as internal state changes within the contracts themselves. Additionally, the value of NFTs is often influenced by external, off-chain factors, further complicating the analysis.

## 9. Conclusion

Applying Cybertrack to 157 Etherscan-flagged contracts [3], our research identified 1,283,198 associated contracts across 91 DCWs. These frauds have accrued 2,638,752 ETH (\$2,089,504,682) in illicit profits, averaging 2.06 ETH (\$1628.36) per contract. Alarmingly, Cybertrack revealed that these frauds date back to September 2017. Notably, our research found that scammers tend to employ multiple fraud contracts to distribute the risk, suggesting the efficacy of current flagging mitigation strategies upon scammers. In response, we are actively collaborating with Etherscan [3] and the FBI [27] to take actions based on our findings.

## References

- [1] *A crypto scam is born every four minutes, report finds*, <https://www.wsj.com/livecoverage/stock-market-news-today-10-27-2022-us-economy-gdp-q3/card-a-crypto-scam-is-born-every-four-minutes-report-finds-EXvZpYofaSx3mgA0QFf1>, [Accessed: 2023-12-03].
- [2] *Most crypto scams on bnb chain, solidus labs says*, <https://www.coindesk.com/business/2022/10/27/most-crypto-scams-on-bnb-chain-solidus-labs-says/>, [Accessed: 2023-12-03].
- [3] *Ethereum (eth) blockchain explorer - etherscan*, <https://etherscan.io/>, [Accessed: 2023-12-03].
- [4] S. Yu, J. Jin, Y. Xie, J. Shen, and Q. Xuan, "Ponzi scheme detection in ethereum transaction network," in *Proceedings of the 3rd International Conference on Blockchain and Trustworthy Systems (BlockSys)*, Guangzhou, China, Aug. 2021.
- [5] Y. Lou, Y. Zhang, and S. Chen, "Ponzi contracts detection based on improved convolutional neural network," in *Proceedings of 2020 IEEE International Conference on Services Computing (SCC)*, Beijing, China, Mar. 2020.
- [6] W. Sun, G. Xu, Z. Yang, and Z. Chen, "Early detection of smart Ponzi scheme contracts based on behavior forest similarity," in *Proceedings of the 20th International Conference on Software Quality, Reliability and Security (QRS)*, Macau, China, Dec. 2020.
- [7] L. Liu, W.-T. Tsai, M. Z. A. Bhuiyan, H. Peng, and M. Liu, "Blockchain-enabled fraud discovery through abnormal smart contract detection on ethereum," *Future Generation Computer Systems*, vol. 128, pp. 158–166, Mar. 2022.
- [8] T. Ashfaq, R. Khalid, A. S. Yahaya, S. Aslam, A. T. Azar, S. Alsafari, and I. A. Hameed, "A machine learning and blockchain based efficient fraud detection mechanism," *Sensors*, vol. 22, no. 19, p. 7162, Sep. 2022.
- [9] H. Hu, Q. Bai, and Y. Xu, "SCSGuard: Deep scam detection for ethereum smart contracts," in *Proceedings of IEEE INFOCOM 2022-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Virtual Conference, May 2022.
- [10] R. M. Aziz, R. Mahto, K. Goel, A. Das, P. Kumar, and A. Saxena, "Modified genetic algorithm with deep learning for fraud transactions of ethereum smart contract," *Applied Sciences*, vol. 13, no. 2, p. 697, Jan. 2023.
- [11] Y. Zhang, W. Yu, Z. Li, S. Raza, and H. Cao, "Detecting ethereum Ponzi schemes based on improved lightgbm algorithm," *IEEE Transactions on Computational Social Systems*, vol. 9, no. 2, pp. 624–637, Jul. 2021.
- [12] X. Shen, S. Jiang, and L. Zhang, "Mining bytecode features of smart contracts to detect Ponzi scheme on blockchain," *Computer Modeling in Engineering & Sciences*, vol. 127, no. 3, pp. 1069–1085, Jan. 2021.
- [13] Y. Chen, H. Dai, X. Yu, W. Hu, Z. Xie, and C. Tan, "Improving Ponzi scheme contract detection using multi-channel textcnn and transformer," *Sensors*, vol. 21, no. 19, p. 6417, Sep. 2021.
- [14] H. Zhang, J. Yu, B. Yan, M. Jing, and J. Zhao, "Security on ethereum: Ponzi scheme detection in smart contract," in *Proceedings of the 16th International Conference on Algorithmic Applications in Management (AAIM)*, Guangzhou, China, Aug. 2022.
- [15] Z. Zheng, W. Chen, Z. Zhong, Z. Chen, and Y. Lu, "Securing the ethereum from smart Ponzi schemes: Identification using static features," *ACM Transactions on Software Engineering and Methodology*, vol. 32, no. 5, 130:1–130:28, 2023.
- [16] J. Peng and G. Xiao, "Detection of smart Ponzi schemes using opcode," in *Proceedings of the 2nd International Conference on Blockchain and Trustworthy Systems (BlockSys)*, Dali, China, Aug. 2020.
- [17] Y. Zhang, S. Kang, W. Dai, S. Chen, and J. Zhu, "Code will speak: Early detection of Ponzi smart contracts on ethereum," in *Proceedings of 2021 IEEE International Conference on Services Computing (SCC)*, Chicago, IL, Sep. 2021.
- [18] W. Chen, Z. Zheng, J. Cui, E. Ngai, P. Zheng, and Y. Zhou, "Detecting Ponzi schemes on ethereum: Towards healthier blockchain technology," in *Proceedings of the 27th International World Wide Web Conference (WWW)*, Lyon, France, Apr. 2018.
- [19] *Fbi identifies cryptocurrency funds stolen by dprk*, <https://www.fbi.gov/news/press-releases/fbi-identifies-cryptocurrency-funds-stolen-by-dprk>, [Accessed: 2023-12-03].
- [20] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, and A. Hobor, "Making smart contracts smarter," in *Proceedings of the 23rd ACM Conference on Computer and Communications Security (CCS)*, Vienna, Austria, Oct. 2016.
- [21] J. Krupp and C. Rossow, "teEther: Gnawing at ethereum to automatically exploit smart contracts," in *Proceedings of the 27th USENIX Security Symposium (Security)*, Baltimore, MD, Aug. 2018.
- [22] S. Kalra, S. Goel, M. Dhawan, and S. Sharma, "Zeus: Analyzing safety of smart contracts.," in *Proceedings of the 2018 Annual Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, Feb. 2018.
- [23] I. Nikolić, A. Kolluri, I. Sergey, P. Saxena, and A. Hobor, "Finding the greedy, prodigal, and suicidal contracts at scale," in *Proceedings of the 34th Annual Computer Security Applications Conference (ACSAC)*, San Juan, PR, Dec. 2018.
- [24] F. Ma, Z. Xu, M. Ren, Z. Yin, Y. Chen, L. Qiao, B. Gu, H. Li, Y. Jiang, and J. Sun, "Pluto: Exposing vulnerabilities in inter-contract scenarios," *IEEE*

- Transactions on Software Engineering*, vol. 48, no. 11, pp. 4380–4396, Oct. 2021.
- [25] J. Frank, C. Aschermann, and T. Holz, “ETHBMC: A bounded model checker for smart contracts,” in *Proceedings of the 29th USENIX Security Symposium (Security)*, Virtual Conference, Aug. 2020.
- [26] P. Bose, D. Das, Y. Chen, Y. Feng, C. Kruegel, and G. Vigna, “SAILFISH: Vetting smart contract state-inconsistency bugs in seconds,” in *Proceedings of the 43rd IEEE Symposium on Security and Privacy (S&P)*, San Francisco, CA, May 2022.
- [27] FBI - Federal Bureau of Investigation, <https://www.fbi.gov/>, [Accessed: 2023-04-06].
- [28] Wikipedia. SHA-3, <https://en.wikipedia.org/wiki/SHA-3>, [Accessed: 2023-04-06].
- [29] Telegram Messenger, <https://telegram.org/>, [Accessed: 2023-04-06].
- [30] Crypterium - BitcoinWiki, <https://en.bitcoinwiki.org/wiki/Crypterium>, [Accessed: 2023-04-06].
- [31] Twitter, <https://twitter.com/>, [Accessed: 2023-04-06].
- [32] Bleeping Computer. Ethereum feature abused to steal \$60 million from 99K victims, <https://www.bleepingcomputer.com/news/security/ethereum-feature-abuse-d-to-steal-60-million-from-99k-victims/>, [Accessed: 2023-11-13].
- [33] Consensys - Mythril, <https://github.com/Consensys/mythril>, [Accessed: 2023-04-06].
- [34] Wikipedia. ethereum. [Online]. Available: <https://en.wikipedia.org/wiki/Ethereum>.
- [35] T. Chen, R. Cao, T. Li, X. Luo, G. Gu, Y. Zhang, Z. Liao, H. Zhu, G. Chen, Z. He, *et al.*, “SODA: A generic online detection framework for smart contracts,” in *Proceedings of the 2020 Annual Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, Feb. 2020.
- [36] C. Sendner, H. Chen, H. Fereidooni, L. Petzi, J. König, J. Stang, A. Dmitrienko, A.-R. Sadeghi, and F. Koushanfar, “Smarter contracts: Detecting vulnerabilities in smart contracts with deep transfer learning,” in *NDSS*, 2023.
- [37] M. Eshghie, C. Artho, and D. Gurov, “Dynamic vulnerability detection on smart contracts using machine learning,” in *Proceedings of the 25th International Conference on Evaluation and Assessment in software engineering (EASE)*, Trondheim, Norway, Jun. 2021.
- [38] S. Gopali, Z. A. Khan, B. Chhetri, B. Karki, and A. S. Namin, “Vulnerability detection in smart contracts using deep learning,” in *Proceedings of 2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC)*, Los Alamitos, CA, Jun. 2022.
- [39] H. Wu, Z. Zhang, S. Wang, Y. Lei, B. Lin, Y. Qin, H. Zhang, and X. Mao, “Peculiar: Smart contract vulnerability detection based on crucial data flow graph and pre-training techniques,” in *Proceedings of 2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE)*, Wuhan, China, Oct. 2021.
- [40] S. Cui, G. Zhao, Y. Gao, T. Tavu, and J. Huang, “VRust: Automated vulnerability detection for solana smart contracts,” in *Proceedings of the 29th ACM Conference on Computer and Communications Security (CCS)*, Los Angeles, CA, Nov. 2022.
- [41] Y. Wang, J. Zhao, Y. Zhang, X. Hei, and L. Zhu, “Smart contract symbol execution vulnerability detection method based on CFG path pruning,” in *Proceedings of the 5th ACM International Symposium on Blockchain and Secure Critical Infrastructure (BSCI)*, Melbourne, VIC, Australia, Sep. 2023.
- [42] Z. Liu, P. Qian, X. Wang, Y. Zhuang, L. Qiu, and X. Wang, “Combining graph neural networks with expert knowledge for smart contract vulnerability detection,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 2, pp. 1296–1310, Feb. 2021.
- [43] F. Gritti, N. Ruaro, R. McLaughlin, P. Bose, D. Das, I. Grishchenko, C. Kruegel, and G. Vigna, “Confusum contractum: Confused deputy vulnerabilities in ethereum smart contracts,” in *Proceedings of the 32nd USENIX Security Symposium (Security)*, Anaheim, CA, Aug. 2023.
- [44] D. Brumley, C. Hartwig, M. G. Kang, Z. Liang, J. Newsome, P. Poosankam, D. Song, and H. Yin, “BitScope: Automatically dissecting malicious binaries,” CS-07-133, School of Computer Science, Carnegie Mellon University, Tech. Rep., 2007.
- [45] Y. Shoshitaishvili, R. Wang, C. Salls, N. Stephens, M. Polino, A. Dutcher, J. Grosen, S. Feng, C. Hauser, C. Kruegel, *et al.*, “SOK: (State of) the art of war: Offensive techniques in binary analysis,” in *Proceedings of the 37th IEEE Symposium on Security and Privacy (S&P)*, San Jose, CA, May 2016.
- [46] Y. Li, Z. Su, L. Wang, and X. Li, “Steering symbolic execution to less traveled paths,” in *Proceedings of the 2013 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, & Applications (OOPSLA)*, Indianapolis, IN, Oct. 2013.
- [47] V. Kuznetsov, J. Kinder, S. Bucur, and G. Candea, “Efficient state merging in symbolic execution,” *Acm Sigplan Notices*, vol. 47, no. 6, pp. 193–204, Jun. 2012.
- [48] M. Yao, J. Fuller, R. P. Kasturi, S. Agarwal, A. K. Sikder, and B. Saltaformaggio, “Hiding in plain sight: An empirical study of web application abuse in malware,” in *Proceedings of the 32nd USENIX Security Symposium (Security)*, Anaheim, CA, Aug. 2023.
- [49] Bitcoinwiki. ERC20, <https://en.bitcoinwiki.org/wiki/ERC20>, [Accessed: 2023-11-13].

[50] *Wikipedia. Non-fungible token*, [https://en.wikipedia.org/wiki/Non-fungible\\_token](https://en.wikipedia.org/wiki/Non-fungible_token), [Accessed: 2023-11-13].

## Appendix

### 1. Start Of Fraudulent Campaign

As highlighted in §2.2, FBI agents were notified with a fraud contract C-98f804 associated with a scam message on Telegram [29]. Further investigation into this message led agents to uncover similar content on other platforms, as shown in Figure 5 and Figure 6, reinforcing the message's fraudulent nature.

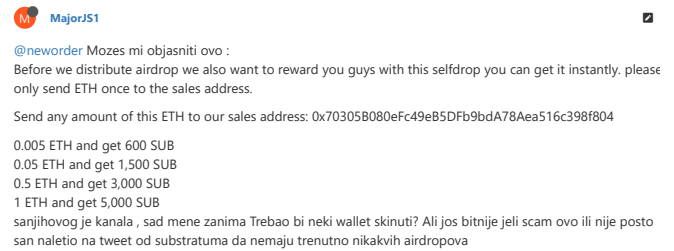


Figure 5: Scammers post the fraudulent message on forum.

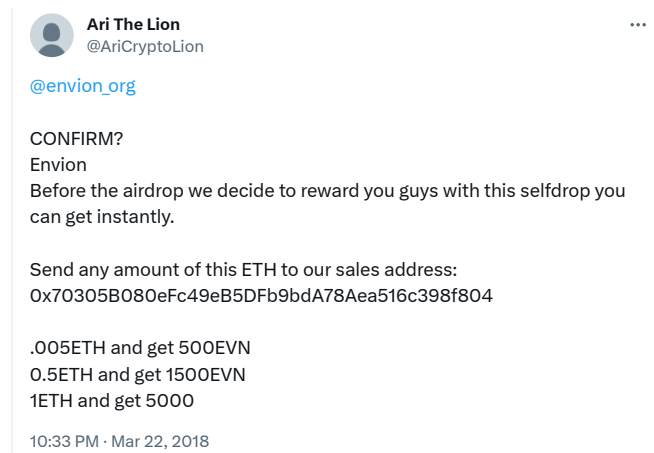


Figure 6: Scammers post the fraudulent message on X.com.



## 2. Full Forms Of Abbreviations

On Ethereum, accounts (i.e., wallets and contracts) are identified by 40-character hexadecimal strings known as addresses, while transactions are distinguished by 64-character hexadecimal strings, termed transaction hashes. To improve the readability, we use abbreviations comprising the last 6 characters of addresses and transaction hashes in the paper. Full forms of these abbreviations are presented in [Table 6](#).

TABLE 6: MAP ABBREVIATION TO FULL FORMS.

Abbr	Full Forms
<b>Contract</b>	
C-98f804	0x70305b080efc49eb5dfb9bda78aea516c398f804
C-48d304	0x5BE1De8021cc883456FD11DC5CD3806dBc48D304
C-6113fB	0xf97Bd29b8eE6E246Eb57eEcf5D0E8486366113fB
C-Fd562c	0xfef14C36C1F2de2ca3772Ba9539B6A58cFd562c
C-2e14CC	0xcB3315A42E76b70D2f3e8E595a5d13855c2e14CC
C-789332	0xcf50193c27DF08423BFe813676541B2268789332
C-D986ae	0x8014FB4882b1f99a3E60AEced39400560D986ae
C-6786ae	0x8014ae6574CAcE1f2435a86d4ea0472f466786ae
C-2D3B2f	0x65a8135596AE13C0Dd5c17bA1059C61Bc42D3B2f
C-D3c82b	0xDD499857c8539bEF04477B52782bE6A9FbD3c82b
C-159624	0xCC326C1D41f64c5331bc7Ba555d75306C3159624
C-8f428e	0xA77db707916aDEff81042ca57656931CcD8f428e
C-805a29	0x5F856630adBC27c0F5bC1DE1961D4f0fB1805a29
C-078228	0x1bd913BBaDE46bF5AD8b1e5d117701fBEb078228
C-0E712A	0xc25ab34E7F3a1eb2C6a3a23DF851F351df0E712A
C-a5f260	0xEb411D5Df13AC7020992306e78955fb7CBa5f260
C-db7aFF	0xBCC6C0feF89b87a12773Db7a9a8ECBCCcDb7aFF
C-18B228	0x95115419B09E8Cea70a9bdbCA3fEe8C5e118B228
C-57e2e8	0x890bcE348BAE449Df3783ba0E1C7eB82C557e2e8
C-4f99A8	0x6032D639E634E788FcE323B316E06d18194f99A8
C-a5CDE9	0x6574C0bf7F3D144F5837acC160773cC8f2a5CDE9
C-3D7D37	0x197e45d545F4DA0C3f15002222BcADDd9D3D7D37
C-70aae3	0x4a96e9b57a229d94c0c28950355A72Fa9e70aae3
C-CbEB9E	0xfdd46E0ea17622d70AdaE6535948776160CbEB9E
C-e18895	0x3cD6ef508c1c448e293075f1dE2ae96a49e18895
C-2C4691	0x5B9E8728E316bBEB692d22daaAB74F6cBF2C4691
C-36FFaE	0x131A99859a8bfa3251D899F0675607766736FFaE
C-1e3e4e	0xbf0c5d82748ed81b5794e59055725579911e3e4e
<b>Wallet</b>	
W-521058	0x2E05A304d3040f1399c8C20D2a9F659AE7521058
W-bceE76	0x29203118cCbBF5277C1CEB49aF1333A91CbceE76
W-82f98B	0xd6E56a65f795Fd136406e668c0eB69360F82f98B
W-2499E7	0xA40b913D654D803b9833e9a699D5830f262499E7
W-b10f4B	0xF52426340e0548a8d58b970f2283e22c1bb10f4B
W-6629BA	0xD0E680D5141f6E61E953903736E3637a6E6629BA
W-2b0878	0xF7cC855E3BB2986729eC47c1B6f64e36aA2b0878
W-09739F	0x848a757656650c9950fb1Aed03eaC8A92209739F
W-223DcF	0xc2221f38dE2eB19125A5b77b5D82d5bFc7223DcF
W-187307	0x4005de995109895BE7Eac74346a62Db28b187307
W-3CD202	0x38c7eA86c8235b0CfCfB91153259e85353CD202
W-297f83	0x7d3Bdf1b728386efDb9a3A0328a95D94b0297f83
W-15467A	0x7734aA368Df7bd09D2AbCBf925Cc92314A15467A
W-64EE46	0xfa32e18Cf5e9E96eDBa979f40DC55E465864EE46
W-1AB876	0xe3172Dc735B44893303e2fd22D1d3647271AB876
W-93abb5	0x9D31e30003f253563Ff108BC60B16Fdf2c93abb5
W-5FFDdc	0xd5e015739a8BEffF075C4eAA2013D27DF35FFDdc
W-13666c	0x058251232C086247cA91998472245D8Ae213666c
W-53C9dA	0x604Df452158e7ddf3E44338308EdC079a953C9dA
<b>Transaction</b>	
T-8bbae5	0x58f6fb1d2440eb4d6d5ac64a152aa156d3850eff6b3-56ab86904ac28758bbae5