

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

Mingxuan Yao, Runze Zhang, Haichuan Xu, Shih-Huan Chou, Varun Chowdhary Paturi,
Amit Kumar Sikder, Brendan Saltaformaggio
Georgia Institute of Technology

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, current mitigation efforts focus 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 CoCo, an automated forensic analysis pipeline that processes a single fraud contract and generates evidence that the legal authorities need to mitigate the fraud. Applying CoCo 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, CoCo traces the fraudulent activities back to September 2017. In response, we are closely collaborating with Etherscan and the 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 the blockchain makes it impossible to reverse the transactions or to identify 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, current mitigation efforts focus 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, the 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 the 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. This effort aligns with the principles of the Uniform Electronic Transactions Act (UETA) [19] and National Commerce Act (E-Sign Act) [20], which recognizes the importance of *electronic agents* (DCW in our paper) in the initiation, execution, and management of electronic contracts. Motivated by the UETA and E-Sign Act, this evidence should encompass: ① The associated contracts from the same DCW along with their recipients. This evidence reflects the concept of *managing electronic contracts*, as defined by the UETA and E-Sign Act. ② The provenance of dynamically resolved recipients, related to the act of *initiating electronic contracts* as delineated in the UETA and E-Sign Act. ③ The attribution of capabilities to the contracts. This evidence is motivated by the act of *executing electronic contracts*, as highlighted in the UETA and E-Sign Act. 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 [21], 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).

Traditionally, FBI agents would acquire such evidence by relying on *explicit* clues, the historical transactions. In fact, prior research [4], [5] leveraged these explicit clues to

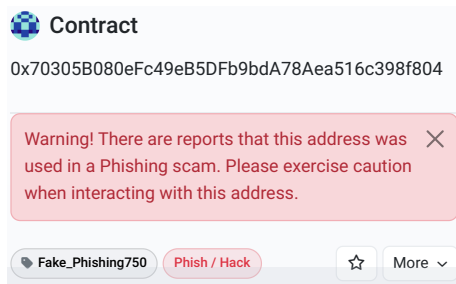


Figure 1: Flagged fraud contract.

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 discover that a forensic approach that focuses strictly on *explicit* clues might cause investigators to omit evidence from yet-to-be-executed transactions, thereby halting the investigation until the relevant transactions occur.

From a different perspective, contract transactions on the 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 the DCW, such as (1) future transactions that the DCW’s contract is programmed to execute, (2) the origin of the recipients used by the DCW’s contract, and (3) the capabilities with which the contract is equipped. In fact, several works have been proposed [22]–[28] 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 to extract 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 CoCo¹, a post-detection forensics pipeline enabling FBI agents to extract three key pieces of evidence of the DCW. Given one fraud contract, CoCo 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, CoCo conducts symbolic analysis on each contract deployed by the 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. CoCo 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, CoCo applies *Capability Attribution Analysis* (§3.3) to assign specific capabilities to each contract deployed by the DCW, similar to a detective deducing the role each suspect plays in a criminal network to tailor countermeasures for each role.

Deploying CoCo on 157 confirmed fraud contracts, our work identified 1,283,198 associated contracts across 91 DCWs, making a total of 2,638,752 ETH (\$2,089,504,682) illicit profit. More alarmingly, CoCo’s analysis tracked the fraudulent activities back to September 2017. We are closely collaborating with Etherscan [3] and the FBI [29] (§B) to mitigate the fraud uncovered by our study. Lastly, we have made CoCo available at <https://github.com/CyFI-Lab-Public/COCO>.

2. Motivation

Accounts (i.e., wallets and contracts) on Ethereum are identified by 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 this paper, we have used abbreviations of the last six characters of the addresses and transaction hashes with the prefixes W- for wallet, C- for contract, and T- for transaction. For example, the fraud contract shown in Figure 2a can be identified by address, `0x70305b080efc49eb5dfb9bda78aea516c398f804`. 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 §C.

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 a smart contract comprises three core components: (1) the contract’s bytecode, which contains the compiled instructions of the smart contract code, (2) an 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 to preserve the 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,

1. CoCo: ForensiCs on CreatOr Wallet Behind Smart COntract Fraud

on the other hand, provide step-by-step execution logs that detail all the internal calls and state changes triggered by a transaction. For example, consider a transaction in which 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 that utilize 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 [30] 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



(a) Fraud message on Telegram. (b) Confirmation on X.com.

Figure 2: Fraud contract in the real world.

To illustrate the method of deriving the three pieces of evidence, we 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 [31]. As shown in Figure 2a, the DCW propagated a deceptive message impersonating Crypterium [32], 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 [33], 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 more proactive mitigation strategies, such as flagging more fraud contracts or freezing the associated wallets or contracts (as was previously done by the FBI [21], [34]).

Associated Contracts & Recipients. Extracting Evidence ❶ consists of two steps: first, identifying the associated contracts and, second, pinpointing their recipients. Our approach begins by leveraging the transparent nature of the blockchain to reveal associated contracts. Specifically, our investigation started with an analysis of transactions to C-98f804, leading us to transaction T-8bbae5, which 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 contracts in total. Subsequent examination of transactions from these contracts unveiled a total of 251,087 contracts 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 future transactions that the contracts is programmed to execute.* Despite their diverse logic, fraud contracts 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 ❶, 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 introduce the methods to dissect this singular contract.

Masquerading As Inactive. The previous analysis of W-521058 shows that a constant fraud contract deployment ranged from October 2017 to January 2021. At first glance, FBI agents might conclude that the DCW ceased the fraud after January 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 invoked 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

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

Fraud Contract	C-98f804
Creator	W-521058
Evidence ①	Associated Contracts: Direct: 395,685 (🚩 11) Indirect: 65,330 (🚩 0) Recipients: W-763bc0
Evidence ②	C-48d304 → Storage → Contract
Evidence ③	<i>attr</i> ₁ : Forced Transfer (395,684) <i>attr</i> ₂ : Contract Self-Replication (1) <i>attr</i> ₃ : Fraud Event Logging (395,684) <i>attr</i> ₄ : Recipient Resolution (65,330)
Fraud Impact	Victim: 232,011 ETH: 1,240,355

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

contracts activated upon its invocation. To make things worse, the 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 the 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 the DCW from the system is almost impossible. Therefore, it becomes equally crucial for 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 a total of four 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* capability group, prompting a *reactive* strategy to flag these accounts and their recipients. Additionally, the detection of CREATE2 opcode usage for deploying asset-transferring contracts highlighted the *Contract Self-Replication* capability group, enabling FBI agents to proactively flag future deployed contracts. Furthermore, we pinpointed 395,684 contracts with the *Fraud Event Logging* capability, using the LOG opcode to generate blockchain event logs. Given the indexability of the event logs, 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 efforts. Alarming, as indicated by 🚩 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 are collaborating closely with Etherscan [3] and FBI [29] to mitigate the fraud.

Investigators With CoCo. Upon receiving a fraud smart contract report, FBI agents input the fraud smart contract address into CoCo for forensic analysis. After CoCo’s analysis, FBI agents acquire the three pieces of evidence discussed. FBI agents can then submit this evidence to the court and relate each piece of evidence to the UETA [19] and E-Sign Act [20], as demonstrated in §1. FBI agents could utilize Evidence ① to flag additional accounts involved in the fraudulent activities. Evidence ② provides FBI agents with the ability to detect and mark the new recipients designated by scammers to gather illicit profits even before these recipients become actively involved. With Evidence ③, FBI agents can take appropriate mitigation actions, as detailed in §3.3.

3. Design

CoCo equips FBI agents with the techniques to investigate the DCW orchestrating the fraudulent activities using smart contracts. Starting with a fraud contract, CoCo

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  $G \leftarrow \bigcup_{\forall txn_i \in T_w} GenerateTxnGraph(txn_i)$ 
17 // Update  $\mathcal{C}$  with contracts deployed directly
18  $\mathcal{C} \leftarrow \{edge.to | edge \in G \wedge edge.type == CreateContract\}$ 
19 // Identify contracts deployed indirectly
20 for  $s_i \in \mathcal{C}$  do
21    $T_{s_i} \leftarrow GetAllTxnFrom(s_i)$ ;
22   for  $trace \in GenerateTxnGraph(txn \in T_{s_i})$  do
23     // Handle different trace type
24     switch  $trace.type$  do
25       // If deploy other contracts
26       case  $CreateContract$  do
27         // Extract deployed contract
28          $s_j \leftarrow trace.to$ 
29         // Get all transactions initiated by
30         // the deployed contract
31          $T_{s_j} \leftarrow GetAllTxnFrom(s_j)$ 
32         // Track contract recursively
33          $G \cup \bigcup_{\forall txn \in T_{s_j}} GenerateTxnGraph(txn)$ 
34          $\mathcal{C} \leftarrow \mathcal{C} \cup s_j$ 
35       end
36     end
37   end
38 end
39 return  $\mathcal{C}$ 
```

generates Evidence ① by uncovering the associated contracts from the same DCW along with the recipients they could interact with (§3.1). Subsequently, CoCo conducts a recipient provenance analysis on dynamically resolved recipients to produce Evidence ② (§3.2). Lastly, CoCo 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, CoCo conducts *Associated Contracts Investigation and Recipients Forensic Investigation*.

3.1.1. Associated Contracts Investigation. Given a reported fraud contract α , CoCo 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, CoCo records a total of 455 transactions directed to it. Naive FBI agents might assume the first transaction txn_1 to be the creation transaction. Unfortunately, as seen in real-world situations [35] and in §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, CoCo traverses through \mathcal{T} and identifies the transaction that *creates* the fraud contract, called a *creation transaction*, denoted as txn_{cr} . For the contract C-98f804, CoCo identifies $txn_{cr} = T-8bbae5$. This identified creation transaction acts as an *explicit* clue, offering 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 $\{t_m.to | t_m \in \mathcal{T} \wedge t_m.type == CreateContract\}$.

Unfortunately, as highlighted in §2.1, W-521058 deployed 65,330 fraud contracts indirectly by invoking contract C-48d304. Since those transactions are a simple contract call, tracking only contract creation transactions initiated by the DCW easily could produce inaccurate Evidence ①. This nuance underscores the importance for FBI agents to discern the *deployment chain*. To overcome this challenge, CoCo 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 txn_{cr} , CoCo deploys the methodology presented in Algorithm 1. As shown on Line 4 of Algorithm 1, for a given transaction, CoCo navigates through traces in that invocation to craft the graph $G = (\mathcal{N}, \mathcal{E})$. Here, each node $n \in \mathcal{N}$ presents an account, and every edge $e \in \mathcal{E}$ presents a trace, which could be subsequent behaviors (e.g., contract invocation, contract creation). By tracking the trace with type *CreateContract*, CoCo is able to identify the contracts deployed indirectly.

Given the ability to identify contracts deployed both directly and indirectly from transactions, CoCo now can identify associated contracts by performing forensic analysis on the DCW’s wallet w . Specifically, CoCo starts by tracking all transactions initiated by w (shown on Line 12 of Algorithm 1), designated as T_w . Then, for each transaction $txn_i \in T_w$, CoCo 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 the transaction used by w to deploy contracts directly and the invocation of the contract. To delve deeper into this chain and disclose the contracts spawned by it, as shown on Line 17 of Algorithm 1, CoCo gathers traces initiated by each contract s_i , termed T_{s_i} . If a trace results in *CreateContract* (shown on Line 19 of Algorithm 1), CoCo first updates G with the transactions from this newly

deployed contract recursively and then incorporates the deployment result into associated contracts.

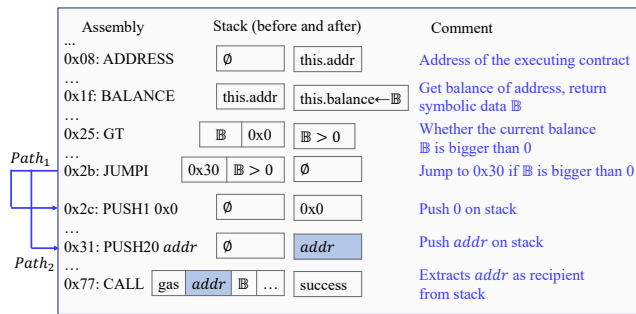


Figure 3: Recipient investigation of C-98f804.

3.1.2. Recipient Investigation. The only missing piece of Evidence ① now 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, CoCo conducts a transaction-agnostic symbolic analysis on each associated contract. Specifically, CoCo designates the input space \mathbb{I} , storage space \mathbb{S} , and account balance \mathbb{B} as symbolic. This symbolic designation allows CoCo 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 CoCo symbolically executes BALANCE opcode to retrieve the balance of the contract, CoCo marks the returned value as symbolic. Subsequently, this symbolic value is used by the GT opcode shown at address 0x25 to assess whether 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, CoCo 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 CoCo 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 CoCo. Subsequently, when the CALL opcode is invoked at address 0x77 of Figure 3, CoCo retrieves the address $addr$ from the stack, indicating its use as a recipient address. Notably, the 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 the contract, CoCo 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 in that not all contracts employ hardcoded recipients; many use dynamic resolution. As CoCo 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. CoCo employs *In Contract Resolution Analysis* and *Cross Contract Resolution Analysis* to ascertain the provenance of dynamically resolved recipients.

In Contract Resolution Analysis. CoCo determines dynamical resolution when the recipient, identified in Recipient Investigation (§3.1.2), corresponds to a symbolic value $recps$. Then, CoCo conducts a *backward slice* on $recps$ to identify the origin of $recps$. For instance, in analyzing the fraud contract deployed by C-48d304 (§2.2), CoCo uncovers that the recipient on the stack is a symbolic value. When CoCo performs backward slice on $recps$, it traces back to opcode AND at address 0x4d, indicating $recps$ is from a logical AND operation. Moving further back, CoCo encounters another AND operation at address 0x37. Continuing this trace, CoCo reaches an opcode DIV at address 0x21. This division operation implies that $recps$ is dependent on a division calculation. The backward slice eventually leads CoCo to address 0x1a with a SLOAD operation. At that moment, CoCo proves that $recps$ originates from the contract’s storage. By examining the parameter passed to SLOAD, a constant 0x00, CoCo concludes that the recipient address is located in storage location $\mathbb{S}[0]$.

Cross Contract Resolution Analysis. The previous analysis reveals only 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, CoCo 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 CoCo 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, CoCo 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, CoCo pinpoints its deployment transaction with hash T-d9d53e, initiated by W-521058 via invoking C-48d304. By analyzing the direct creator contract, CoCo extracts the init code from the 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 a value. Consequently, CoCo 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. CoCo uses symbolic data, introduced in §3.1.2, effectively highlighting the information flow to track the capabilities of associated contracts. An example in §3.2 demonstrates this: When CoCo 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 CoCo, along with semantic models and mitigation strategies, are outlined in Table 2. As seen on Column 4 of Table 2, CoCo proposes three different mitigation strategies. The function $F(\theta)$ represents flagging and freezing the account denoted by θ . $S(\theta)$ means scanning the blockchain for contracts sharing the same code as θ if θ is a contract address, or for identical data to θ if θ represents raw byte content. $M(\theta)$ indicates monitoring the account specified by θ . Notably, CoCo's approach to symbolic analysis, independent of semantic model knowledge, requires only a one-time effort and offers ease of extension.

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 the 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 the DCW

to alter the recipient address. Reporting this capability allows FBI agents not only to flag the current recipient address, but also to monitor the location where the contract load recipient address, in order to detect a change in the recipient's address before fraud occurs.

4. Address Allocation. The SSTORE opcode, coupled with a non-symbolic first stack element that is an address of an 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 the 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 the 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 the 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 the 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 the stack that points to an account, and a key exists 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 CoCo encounters the LOGX opcode ($X \in [0, 4]$), which is used to log events. This capability suggests that the contract logs event that could be used to track the contract's activities and identify fraudulent transactions.

CoCo is now equipped with the technique to produce Evidence ③ by attributing capabilities to each associated contract and proposing corresponding mitigation strategies.

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

Fraud Capability	Opcode	Semantic Models	Mitigation ¹
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.

This categorization enables FBI agents to implement bulk mitigation actions effectively.

4. Validation

We implemented a prototype of CoCo in Python leveraging Mythril [36], 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 with different DCWs. The distribution of ground truth dataset aligns with the overall distribution of the 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.

Ground Truth. Generating ground truth for the validation dataset required us to produce both on-chain transaction ground truth and off-chain contract implementation ground truth. For transaction ground truth, we deployed an Ethereum archive node utilizing Reth [37] to access the on-chain data. For the contract implementation ground truth, we employed Panoramix [38] and Dedaub [39] to decompile each contract, followed by a manual inspection

of the decompiled code to ascertain its provenance, logic, and capabilities.

4.1. Associated Contracts & Recipients

Table 3 presents CoCo’s validation result. As shown in the False Positive (FP) and False Negative (FN) columns, CoCo generated seven FPs and one FN. To prevent overstating performance, we assign one TP in both Associated Contracts (Columns 3-4 of Table 3) and Deploy Length (Columns 7-8 of Table 3) if CoCo correctly identifies all associated contracts and deployment lengths, respectively, otherwise assigning zero in each case. Given this, we can see that CoCo achieves an overall accuracy of 90.91% , demonstrating its efficiency in generating precise forensic evidence. Delving further into the performance of CoCo, each piece of 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 CoCo and the uncovered DCW, respectively. Interested readers can find the full address of the abbreviation in Table 6. Columns 3-6 show the validation results for Evidence ①.

As shown in the Total row, CoCo 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 two FPs can be attributed to the distinct methods employed by DCWs (Row 6 and Row

TABLE 3: CoCo’S VALIDATION.

Contract	DCW	Evidence ①				Evidence ②				Evidence ③		FP	FN
		Associated Contracts		Recipients		Deploy Length ²		Origin		GT	C		
		GT	C ¹	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
Total	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 CoCo.

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.

12 of Table 3) in deploying the fraud contracts. Specifically, CoCo 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, CoCo 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, CoCo 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 that FBI agents in this case would have no obvious evidence of the fraud intention, we consider this as FP. Notably, CoCo still reported these two contracts since they give FBI agents an important lead on potential contract abuse. We confirmed that these are rare cases in our dataset. Overall, CoCo was 90.91% accurate in generating evidence, making it robust for our evaluation.

4.2. Dynamically Resolved Recipients

Columns 7-10 of Table 3 present the validation result of CoCo generating Evidence ②. As illustrated in Column 8 and Column 10 of Table 3, CoCo identified 3,162,963 Deploy Lengths (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, CoCo generated FP when analyzing DCW W-3CD202 and W-5FFDdc. This FP is the direct result of the FP in the Evidence ① generation, as discussed in §4.1. We also see one False Negative (FN) shown on Row 6 of Table 3 (W-3CD202) during the track of the 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 CoCo’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. A 90.91% accuracy in generating evidence makes CoCo robust for our evaluation.

4.3. Capability Attribution

Columns 11-12 illustrate the performance of CoCo regarding the generation of Evidence ③. As presented in the Total row, CoCo 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 (shown on Row 6 of Table 3), CoCo 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, CoCo misclassified it as a separate attributed group. It’s important to note that the FP in the Evidence ③ generation is intrinsically linked to the FP in the Evidence ① generation. FBI agents, by identifying and excluding C-2C4691 from the forensic analysis of the 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%, CoCo is validated as an effective tool for generating the three pieces of evidence of the DCW that FBI agents require to mitigate fraudulent activities.

5. Evaluation

In this section, we demonstrate the effectiveness of CoCo in uncovering the impact of DCWs. We obtained contracts marked as fraudulent from Etherscan [3]. To avoid false positives and potential overclaiming, we manually verified each marked contract by checking for either public reports or reported victims in Etherscan’s comment section. This resulted in 157 flagged contracts. We utilized an open-source project [40] that pulled the Etherscan labeled data.

5.1. Post Deployment Dataset Highlights

Deploying CoCo on our dataset revealed an unnerving trend in fraudulent activities perpetrated through smart contracts. Given 157 flagged smart contracts on Etherscan [3], CoCo uncovered a total of 1,283,198 associated contracts from 91 DCWs behind these contracts. The distribution of these associated contracts is presented in Figure 4. Furthermore, as highlighted under *Profit (ETH)* in Figure 4, CoCo assessed the impact of these fraudulent activities by calculating the ETH equivalent of the illicit profits stolen by DCWs. Specifically, CoCo 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.

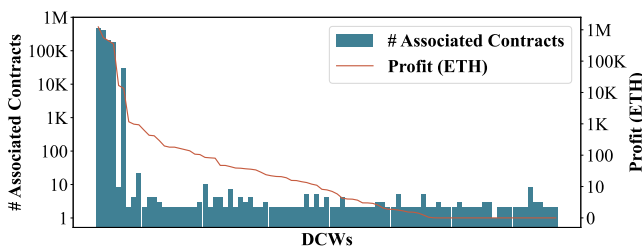


Figure 4: Distribution of associated contracts and illicit ETH profits across DCWs.

5.2. Impact Of DCW

Table 4 presents the top 10 DCWs with the highest fraud profit identified by CoCo. Column 1 shows the abbreviation of each DCW. Columns 2 and 3 show the start and end dates of each DCW’s 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 that the objective of the DCW is to

generate real-world currency profits, CoCo 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 the severity the fraud based on the number of affected individuals. In support of this, CoCo 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 the DCW detailed in Column 1 on a daily basis. Column 10 provides a total victim count. To underscore the significance of CoCo, we compared the efficacy in victim identification by FBI agents solely from reported fraud contracts against the enhanced forensics capabilities equipped by using CoCo.

Table 4 provides insight into the operations of fraudulent activities conducted via smart contracts. Column 2 reveals that, out of 10 contracts analyzed by CoCo, the inception of such fraudulent activities dates back five years to September 2017, which is only two years subsequent to Ethereum’s introduction [41]. 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 is apparent that four out of 10 DCWs (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 five years. Considering the observed efforts by government agencies (e.g., the FBI) to combat general cryptocurrency fraud [21], it is clear that the stealth and complexity of smart-contract-based fraud have outpaced the investigative capabilities of legal authorities.

The enduring presence of DCWs, coupled with inadequate investigative approaches, has led to the rampant presence of these fraudulent activities. An examination of Column 4 of Table 4, which details the daily transaction volume of the DCW listed in Column 1, yields several notable observations: (1) As evidenced by Rows 1-4 of Table 4, substantial DCWs 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 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

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 CoCo
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%
Summary	Sep 2017	Jun 2023		2,002,533	1,741,081,326	1	5,756	232	493,068	445,556%

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 DCWs.

It is clear that fraudulent activities facilitated by smart contracts present a critical issue, especially considering the substantial illicit profits they have accrued. 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 shows that the total victim count from these fraudulent operations could reach 493,068. Focusing on Row 1 of Table 4, we find that the fraudulent activities conducted by W-521058 have affected 232,011 individuals, which represents 47.05% of all identified victims. Comparing Columns 5-6 with Column 10, it is 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 using CoCo. The Summary row highlights an average gain of 445,556% additional victims, reinforcing the importance of integrating CoCo into DCW fraud forensics.
















5.3. Drill Down Into Fraud Contracts

As detailed in §5.2, we have established that the illicit profits collected by the fraudulent activities can reach \$1,741,081,326 and the number of involved victims can easily reach 493,068. However, it is unclear whether the smart contracts driving these frauds exhibit a similar transaction pattern (e.g., volume) or profit margin. 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 CoCo. Column 1 shows the abbreviation for each contract; 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 each 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, CoCo has converted the ETH amounts into USD based on the historical exchange rates at the time of each transaction. The corresponding USD profits are shown 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 shows that the contracts on Rows 4, 8-10, and 14-15 of Table 5 (40.00% of the top 15 contracts) undergo a significant period of dormancy before experiencing a

TABLE 5: TOP 15 FRAUD CONTRACTS IDENTIFIED BY CoCo 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
Summary	Oct 2017	Jun 2023		0.004	14,592	325	673,221	7.383	23,519,179	360,436	745,743,608

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 is 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, shows 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 CoCo, 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 CoCo for

comprehensive forensic analysis.

6. Case Studies

6.1. Case Study 1: Contracts Shared By DCWs

Deploying CoCo on the dataset surprisingly revealed contract reuse across different DCWs. Specifically, CoCo identified a total of three different bytecodes used by seven DCWs involving 16 contracts. The FBI agents may incorrectly assume that contracts sharing the same bytecode would execute the 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, CoCo reported that these 16 contracts are all equipped with the *Dynamic Recipient Resolution* capability. For example, CoCo observed that the contracts could execute SLOAD to load an address from the sixth 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, CoCo discovered that the sixth storage slot consistently hosted the address of the *creator*. Consequently, even though these 16 contracts share three 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 CoCo also

enables a detailed examination of their technical evolution. During the analysis of W-521058, who has collected around 1,250,355 of illicit ETH profit, CoCo observed an evolution from manual 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, CoCo discovered that instead of using the CREATE opcode, the DCW employed CREATE2. The CREATE opcode computes the address of the deployed contract using Keccak256 [30] on *sender* and *nonce* where *sender* is the address of the 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 [35]. 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 that 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. [5] improved on the LightGBM algorithm for better detection efficacy. Extracting bytecode features [6] and leveraging text convolutional neural networks [7] have also been proposed to identify Ponzi-like characteristics in smart contracts. There are also works using opcode compression for vulnerability detection [8] and identifying static features within smart contracts for Ponzi scheme classification [9]. Techniques using opcode sequences [10], code analysis [11], and enhanced convolutional neural networks [12], [13] have also been implemented to identify Ponzi schemes.

Fraud And Phishing Detection. Beyond Ponzi schemes, broader fraud detection on the blockchain is tackled through various innovative approaches. Liu et al. [15] 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. [16] to classify Ethereum transactions and detect anomalies like double-spending and Sybil attacks. Furthermore, Hu et al. [17] explored deep learning models

to identify scams on a large scale by examining the N-gram bytecode patterns of a smart contract. Notably, the identification of fraud contracts acts as the input for CoCo, enabling it to generate evidence that assists FBI agents in obtaining legal authorization and implementing mitigation strategies. The research community has also studied a wide range of Phishing transactions [42]–[45]. Unlike these existing studies, which treat the fraud contracts like a blackbox and focus on examining/triggering transactions, CoCo bridges the gap between the contract implementation and on-chain transactions to produce forensic evidence.

7.2. Smart Contract Vulnerability

Luu et al. [22] were among the first to address the security risks in smart contracts by identifying security pitfalls. Symbolic analysis has been used frequently to detect such vulnerabilities; Krupp and Rossow [23] developed techniques to automatically exploit smart contracts by analyzing bytecode. Similarly, Zeus [24], a framework for the symbolic verification of smart contracts, uses Constrained Horn Clauses to identify vulnerabilities, an approach also adopted by others [25], [27]. With the rising complexity of smart contracts, especially those involving multiple contracts, Ma et al. [26] proposed an inter-contract analysis tool, while others have focused on state inconsistency bugs [28]. Online detection methods have also been investigated, such as SODA [46], an online detection framework for smart contracts.

Recent works have shifted toward leveraging machine learning to enhance the detection process. Sendner et al. [47] introduced ESCORT, a method employing deep learning to identify different types of smart contract vulnerabilities. There are also works utilizing dynamic analysis [48], deep learning [49], and pre-training techniques [50] to identify smart contract vulnerabilities. Others have focused on static analysis rules tailored to specific blockchain platforms, like VRust for Solana smart contracts [51], or on defining critical paths to address fund transfer vulnerabilities [52]. Moreover, combining expert knowledge with graph neural networks has shown promise in enhancing detection capabilities [53]. Another notable approach is the analysis of confused deputy vulnerabilities in Ethereum smart contracts [54], which highlights the importance of understanding the security challenges in contract interactions. Contributing to a different domain, CoCo conducts forensic analysis on fraud contracts, producing the evidence for FBI agents to obtain legal authorization and execute mitigation strategies.

8. Discussion

Ethical Concerns. The data utilized in this analysis is sourced exclusively from the public domain, specifically Ethereum. This research centers on conducting forensic investigation of fraud contracts, and it is important to clarify that we *do not* engage in any exploitative activities.

Challenges of Symbolic Analysis. CoCo leverages symbolic analysis, a technique adopted by top-tier

research [55]–[60]. While symbolic analysis often faces the challenge of path explosion, our evaluation in §5 shows 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. CoCo primarily focuses on the forensic analysis of fraudulent activities involving the scam of ETH. ERC-20 [61] and Non-Fungible Tokens (NFTs) [62] have emerged as extensions built upon 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 contracts, 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.

Generalizability of CoCo. The design principles of CoCo are not tied to specific features of any blockchain (e.g., Ethereum). This makes CoCo portable to other blockchain platforms. Specifically, since blockchain technology mandates the public disclosure of all transactions, the transaction forensics components of CoCo can be applied across different blockchains. Additionally, the program analysis components of CoCo can be easily extended for use with any blockchain that supports smart contracts. For instance, BNB Smart Chain (BSC) [63] is compatible with EVM bytecode, enabling direct application of CoCo. Similarly, Solana [64] employs Berkeley Packet Filter (BPF) bytecode for its smart contracts. Porting CoCo for Solana only requires a one-time effort to map the current bytecode set used by CoCo to Solana’s bytecode set.

9. Conclusion

Applying CoCo 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. CoCo 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 [29] to take actions based on our findings.

10. Acknowledgement

We thank the anonymous reviewers for their constructive comments and feedback. In particular, we

thank our shepherd for the guidance throughout the revision process. We also thank our collaborators at the FBI and Etherscan for their support, insights, and suggestions throughout this research. This material was supported in part by the Office of Naval Research (ONR) under grants N00014-19-1-2179 and N00014-23-1-2073; the National Science Foundation (NSF) under grant 2143689; and Cisco Systems under an unrestricted gift. Any opinions, findings, and conclusions in this paper are those of the authors and do not necessarily reflect the views of our sponsors and collaborators.

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-scams-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. 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.
- [6] 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.
- [7] 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.
- [8] 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.
- [9] 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, pp. 1–28, 2023.
- [10] 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.
- [11] 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.
- [12] 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.
- [13] Y. Lou, Y. Zhang, and S. Chen, "Ponzi contracts detection based on improved convolutional neural network," in *Proceedings of the 2020 IEEE International Conference on Services Computing (SCC)*, Beijing, China, Mar. 2020.
- [14] 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.
- [15] 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.
- [16] 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.
- [17] 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.
- [18] 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.
- [19] *Wikipedia. Uniform Electronic Transactions Act*, https://en.wikipedia.org/wiki/Uniform_Electronic_Transactions_Act, [Accessed: 2023-11-13].
- [20] *Wikipedia. Electronic Signatures in Global and National Commerce Act*, https://en.wikipedia.org/wiki/Electronic_Signatures_in_Global_and_National_Commerce_Act, [Accessed: 2023-11-13].
- [21] *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].
- [22] 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.
- [23] 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.
- [24] 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.

- [25] 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.
- [26] 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.
- [27] 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.
- [28] 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.
- [29] FBI - Federal Bureau of Investigation, <https://www.fbi.gov/>, [Accessed: 2023-04-06].
- [30] Wikipedia. SHA-3, <https://en.wikipedia.org/wiki/SHA-3>, [Accessed: 2023-04-06].
- [31] Telegram Messenger, <https://t.me/>, [Accessed: 2023-04-06].
- [32] Crypterium - BitcoinWiki, <https://en.bitcoinwiki.org/wiki/Crypterium>, [Accessed: 2023-04-06].
- [33] Twitter, <https://twitter.com/>, [Accessed: 2023-04-06].
- [34] Tether assists us government in seizing \$1.4 million from unhosted wallet, <https://news.bitcoin.com/tether-assists-us-government-in-seizing-1-4-million-from-unhosted-wallet/>, [Accessed: 2024-03-14].
- [35] 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].
- [36] Consensus - Mythril, <https://github.com/Consensus/mythril>, [Accessed: 2023-04-06].
- [37] GitHub. reth, <https://github.com/paradigmxyz/reth>, [Accessed: 2023-11-13].
- [38] GitHub. Panoramix, <https://github.com/palkeo/panoramix>, [Accessed: 2023-11-13].
- [39] Dedaub. Smart Contract Security Analysis, <https://dedaub.com/>, [Accessed: 2023-11-13].
- [40] GitHub. evm-labels, <https://github.com/dawsbot/evm-labels>, [Accessed: 2023-11-13].
- [41] Wikipedia. ethereum. [Online]. Available: <https://en.wikipedia.org/wiki/Ethereum>.
- [42] B. He, Y. Chen, Z. Chen, X. Hu, Y. Hu, L. Wu, R. Chang, H. Wang, and Y. Zhou, "TxPhishScope: Towards detecting and understanding transaction-based phishing on ethereum," in *Proceedings of the 30th ACM Conference on Computer and Communications Security (CCS)*, Copenhagen, Denmark, Nov. 2023.
- [43] W. Chen, X. Guo, Z. Chen, Z. Zheng, and Y. Lu, "Phishing scam detection on ethereum: Towards financial security for blockchain ecosystem.," in *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI)*, Yokohama, Japan, Jan. 2021.
- [44] S. Li, G. Gou, C. Liu, C. Hou, Z. Li, and G. Xiong, "TTAGN: Temporal transaction aggregation graph network for ethereum phishing scams detection," in *Proceedings of the 31st International World Wide Web Conference (WWW)*, Virtual Conference, Apr. 2022.
- [45] P. Xia, H. Wang, B. Gao, W. Su, Z. Yu, X. Luo, C. Zhang, X. Xiao, and G. Xu, "Trade or trick? detecting and characterizing scam tokens on uniswap decentralized exchange," in *Proceedings of the ACM on Measurement and Analysis of Computing Systems (POMACS)*, vol. 5, Dec. 2021, pp. 1–26.
- [46] 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.
- [47] 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 *Proceedings of the 2023 Annual Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, Feb. 2023.
- [48] 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.
- [49] 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.
- [50] 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.
- [51] 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.
- [52] 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.

- [53] 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.
- [54] 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.
- [55] 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.
- [56] 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.
- [57] 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.
- [58] 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.
- [59] 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.
- [60] J. Fuller, R. P. Kasturi, A. K. Sikder, H. Xu, B. Arik, V. Verma, E. Asdar, and B. Saltaformaggio, “C3PO: Large-scale study of covert monitoring of C&C servers via over-permissioned protocol infiltration,” in *Proceedings of the 28th ACM Conference on Computer and Communications Security (CCS)*, Seoul, South Korea, Nov. 2021.
- [61] *Bitcoinwiki*. ERC20, <https://en.bitcoinwiki.org/wiki/ERC20>, [Accessed: 2023-11-13].
- [62] *Wikipedia*. Non-fungible token, https://en.wikipedia.org/wiki/Non-fungible_token, [Accessed: 2023-11-13].
- [63] *Binance Smart Chain*, <https://www.bnbchain.org/en/bnb-smart-chain>, [Accessed: 2023-11-13].
- [64] *Solana*, <https://solana.com/>, [Accessed: 2023-11-13].

Appendix A. 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 [31]. 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.

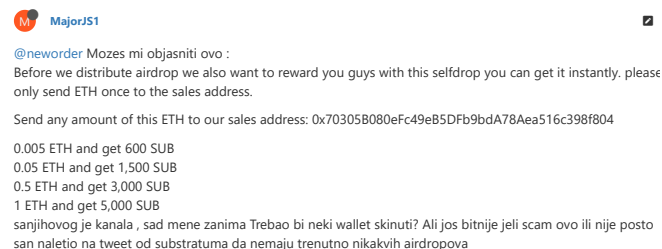


Figure 5: Scammers post the fraudulent message on forum.

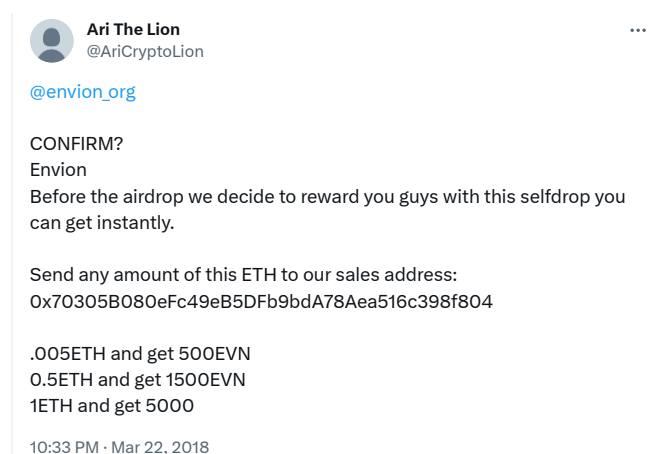


Figure 6: Scammers post the fraudulent message on X.com.

Appendix B. Collaboration With Investigators

Figure 7 presents our communication with the FBI [29]. We redacted the name of the agent but kept the domain name for security and ethical considerations.

Appendix C. Full Forms Of Abbreviations

Accounts on Ethereum (i.e., wallets and contracts) are identified by 40-character hexadecimal strings known as addresses, while transactions are distinguished by 64-character hexadecimal strings, called transaction hashes. To improve the readability, we use abbreviations comprising the last six characters of addresses and transaction hashes in the paper. The full forms of these abbreviations are presented in Table 6.

From: [REDACTED] <[REDACTED]@FBI.GOV>
Sent: Tuesday, December 5, 2023 10:50 AM
To: Saltaformaggio, Brendan D <brendan@ece.gatech.edu>
Subject: FBI Referral

Good Morning Brendan,

I received your contact information from a complaint you submitted involving fraudulent Ethereum smart contracts. I would love to set up a meeting to discuss the information you and your team identified during your investigation. Feel free to give me a call to discuss in detail and we can schedule a day in the next week to come to your office.

Thanks

Special Agent [REDACTED]
 Complex Financial Crimes
 FBI Atlanta
 3000 Flowers Road South
 Atlanta
 Cell #: [REDACTED]
 Desk #: [REDACTED]

From: Saltaformaggio, Brendan D <brendan@ece.gatech.edu>
Sent: Tuesday, December 5, 2023 1:18 PM
To: [REDACTED] (AT) (FBI) <[REDACTED]@FBI.GOV>
Cc: Yao, Mingxuan <mingxuanyao@gatech.edu>
Subject: [EXTERNAL EMAIL] - Re: FBI Referral

Hi [REDACTED],

Nice to talk to you. I am also looping in Mingxuan Yao (cc'ed). He is the lead researcher on this project.

We are free next Tuesday or Wednesday. Let me know if that works for your team.

We are in the coda building (<https://maps.app.goo.gl/LMwZGqgrSekk8Z9>) in Midtown Atlanta.

Feel free to call my cell any time: [REDACTED]

Best,
 Brendan

RE: FBI Referral

[REDACTED]@FBI.GOV>
 Wed 12/6/2023 2:25 PM
 To: Saltaformaggio, Brendan D <brendan@ece.gatech.edu>
 Cc: Yao, Mingxuan <mingxuanyao@gatech.edu>

Good Afternoon Brendan,

Wednesday afternoon next week works for us. Let me know if that works for you all.

Thanks

Special Agent [REDACTED]
 Complex Financial Crimes
 FBI Atlanta
 3000 Flowers Road South
 Atlanta
 Cell #: [REDACTED]
 Desk #: [REDACTED]

Figure 7: Collaborating with FBI.

TABLE 6: MAP ABBREVIATION TO FULL FORMS.

Abbr	Full Forms
Contract	
C-98f804	0x70305b080efc49eb5dfb9bda78aea516c398f804
C-48d304	0x5BE1De8021cc883456FD11DC5CD3806dBc48D304
C-6113fB	0xf97Bd29b8eE6E246Eb57eEcf5D0E8486366113fB
C-Fd562c	0xefef14C36C1F2de2ca3772Ba9539B6A58cFd562c
C-2e14CC	0xcB3315A42E76b70D2f3e8E595a5d13855c2e14CC
C-789332	0xcF50193c27DF08423BFe813676541B2268789332
C-D986ae	0x8014FB4882b1f99a3E60AEce1d39400560D986ae
C-6786ae	0x8014ae6574CAcE1f2435a86d4ea0472f466786ae
C-2D3B2f	0x65a8135596AE13C0Dd5c17bA1059C61Bc42D3B2f
C-D3c82b	0xDD499857c8539bEF04477B52782bE6A9FbD3c82b
C-159624	0xCC326C1D41f64c5331bc7Ba555d75306C3159624
C-8f428e	0xA77db707916aDEff81042ca57656931CcD8f428e
C-805a29	0x5F856630adBC27c0F5bC1DE1961D4f0fB1805a29
C-078228	0x1bd913BBADE46bF5AD8b1e5d117701fEB078228
C-0E712A	0xc25ab34E7F3a1eb2C6a3a23DF851F351df0E712A
C-a5f260	0xEb411D5Df13AC7020992306e78955f67CBA5f260
C-db7aFF	0xBCC6C0feF89b87a12773Db7a9a8ECBCCcDb7aFF
C-18B228	0x95115419B09E8Cea70a9bDbCA3fEe8C5e118B228
C-57e2e8	0x890bcE348BAE449Df3783ba0E1C7eB82C557e2e8
C-4F99A8	0x6032D639E634E788FcE323B316E06d18194f99A8
C-a5CDE9	0x6574C0bF7F3D144F5837acC160773c8f2a5CDE9
C-3D7D37	0x197e45d45F4DA0C3f15002222BcADDd9D3D7D37
C-70aae3	0x4a96e9b57a229d94c0c28950355A72Fa9e70aae3
C-CbEB9E	0xfdd46E0ea17622d70AdaE6535948776160CbEB9E
C-e18895	0x3cD6ef508c1c448e293075f1dE2ae96a49e18895
C-2C4691	0x5B9E8728E316bBEB692d22daaAB74F6cBF2C4691
C-36FFaE	0x131A99859a8bfa3251D899F0675607766736FFaE
C-1e3e4e	0xbf0c5d82748ed81b5794e59055725579911e3e4e
Wallet	
W-521058	0x2E05A304d3040f1399c8C20D2a9F659AE7521058
W-bceE76	0x29203118cCbBF5277C1CEB49aF1333A91CbceE76
W-82f98B	0xd6E56a65f795Fd136406e668c0eB69360F82f98B
W-2499E7	0xA40b913D654D803b9833e9a699D5830f262499E7
W-b10f4B	0xF52426340e0548a8d58b970f2283e22c1bb10f4B
W-6629BA	0xD0E680D5141f6E61E953903736E3637a6E6629BA
W-2b0878	0xF7c855E3BB2986729eC47c1B6f64e36aA2b0878
W-09739F	0x848a757656650c9950fb1Aed03eaC8A92209739F
W-223DcF	0xc2221f38dE2eB19125A5b77b5D82d5bFc7223DcF
W-187307	0x4005de995109895BE7Eac74346a62Db28b187307
W-3CD202	0x38c7eA86c8235b0CfCfB91153259e85353CD202
W-297f83	0x7d3Bdf1b728386efDb9a3A0328a95D94b0297f83
W-15467A	0x7734aA368Df7bd09D2AbCBf925C92314A15467A
W-64EE46	0xfa32e18Cf5e9E96eDBa979f40DC55E465864EE46
W-1AB876	0xe3172Dc735B44893303e2fd22D1d3647271AB876
W-93abb5	0x9D31e30003f253563Ff108BC60B16Fdf2c93abb5
W-5FFDdc	0xd5e015739a8BEffF075C4eAA2013D27Df35FFDdc
W-13666c	0x058251232C086247cA91998472245D8Ae213666c
W-53C9dA	0x604Df452158e7ddF3E44338308EdC079a953C9dA
Transaction	
T-8bbae5	0x58f6fb1d2440eb4d6d5ac64a152aa156d3850eff6b3-56ab86904ac28758bbae5

Appendix D. Meta-Review

The following meta-review was prepared by the program committee for the 2024 IEEE Symposium on Security and Privacy (S&P) as part of the review process, as detailed in the call for papers.

D.1. Summary

This paper presents CoCo, an automated approach to identify deceptive creator wallets (DCWs) through fraud contracts to aid in identifying the stakeholders in a fraud scheme rather than just individual wallets.

D.2. Scientific Contributions

- Creates a new tool to enable future science
- Provides a Valuable Step Forward in an Established Field

D.3. Reasons for Acceptance

- 1) This paper is well-written.
- 2) This paper presents an in-depth, real-world analysis of experimental results with some interesting findings.