Supplementary Material

I. Artifacts

We list the anonymous artifact links below.
Dataset (LaunderNetEvm41):
https://github.com/AnonymousMFTracer/LaunderNetEvm41.
Source Code:
https://github.com/AnonymousMFTracer/codes.
Address & Tx Lists:
https://github.com/AnonymousMFTracer/findings.

II. Details of the Example in Fig. 1

A. Example Overview

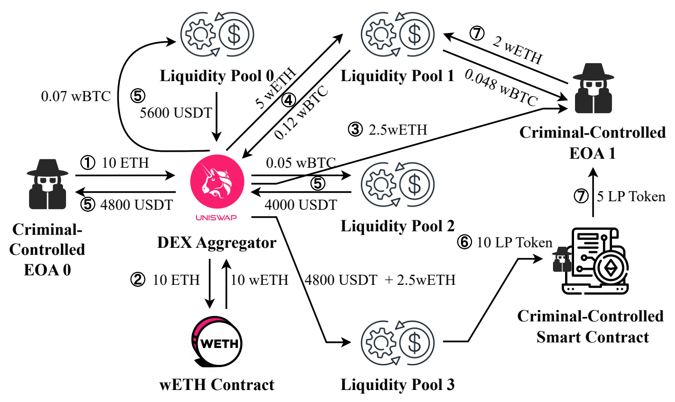


Fig. 1: An real-world example illustrates how a criminal executed a Multi-call transaction on a DEX aggregator to obscure illicit fund flows as part of their laundering strategy.

In this example, the criminal invokes multiple functions in a DEX aggregator contract within a single Multi-call transaction, triggering fund movements across multiple addresses.

The execution of this transaction can be divided into seven sub-processes. Readers may find it helpful to consult Figure 1 while reading this part. ① The criminal-controlled EOA 0 sends 10 `ETH` (Ethereum's native token) to the DEX aggregator. ② The aggregator wraps the received 10 `ETH` into the ERC-20 standard token `wETH` using the wETH contract. ③ The aggregator transfers 2.5 `wETH` to the criminal-controlled EOA 1. ④ The aggregator calls the `Swap` protocol to exchange 5 `wETH` for 0.12 `wBTC` at Liquidity Pool 1. ⑤ The aggregator calls the `Swap` protocol to exchange 0.07 `wBTC` for 5,600 `USDT` at Liquidity Pool 0 and the remaining 0.05 `wBTC` for 4,000 `USDT` at Pool 2, totaling 9,600 `USDT`, where 4800 ones are transferred back to EOA 0. ⑥ The aggregator calls the `AddLiquidity` protocol to deposit 2.5 `wETH` and 4,800 `USDT` into Pool 2 and the received 10 `LP Tokens` are transferred to a criminal-controlled smart contract. ⑦ Since the smart contract had previously obtained an allowance from EOA 1, it executes predefined code to swap 2.5 `wETH` from EOA 1 for 0.048 `wBTC` at Pool 1, and transfer 5 `LP Tokens` to EOA 1.

B. Apply Algorithm 1 to this Example

As shown in Figure 2, Algorithm 1 constructs the local money transfer graph G_{L} and the balance change table B for this transaction to decipher its intricate execution sub-processes. Upon examining B, MFTracer determines that the transaction does not result in any USD-denominated balance changes for the aggregator or liquidity pools. This is because the semantics of `Swap` and `AddLiquidity` do not inherently transfer USD value but change the composition of tokens associated with each address [1], [2]. However, EOA 0 experiences a decrease in balance, whereas EOA 1 and the criminal-controlled contract see an increase. Using this information, MFTracer concludes that the actual fund provider is the criminal-controlled EOA 0, and that the fund recipients are EOA 1 and the criminal-controlled smart contract. Then it applies subsequent computational steps on G_{L} to determine the fund flows from EOA 0 to EOA 1 and the smart contract, thereby fully uncovering the underlying fund movements of this complex transaction.

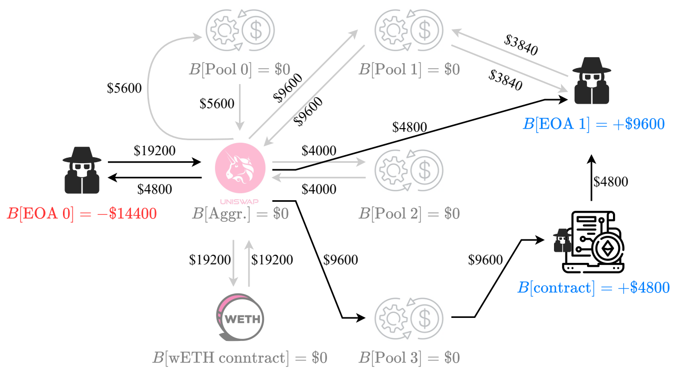


Fig. 2: The local money transfer graph G_{L} and balance change table B constructed by Algorithm 1 for the Multi-call transaction in Figure 1.

III. Proof for the Pruning Inequality of III-C

For a valid money flow path $a_1 \to a_2 \to \ldots \to a_n$, let t_i represent the timestamp of the flow between address a_i and address a_{i+1}. We have $t_i \geq t_{i-1}$ holds for all $1 < i < n$. It follows that for all $0 < j < i < n$, we have $t_i \geq t_j$. Continuing with the notation of the paper, it follows straightforwardly that a valid path from a_1 to a_n exists in G only if

$$\forall\ 0 < j < i < n,\ \exists\ t_i \in \tau_{a_i,a_{i+1}},\ \exists\ t_j \in \tau_{a_j,a_{j+1}},\ t_i \geq t_j.$$

That is,

$$\forall\ 0 < j < i < n,\ \exists\ t_i \in \tau_{a_i,a_{i+1}},\ t_i \geq \min \tau_{a_j,a_{j+1}}.$$

Equivalent to

$$\forall\ 0 < j < i < n,\ \max \tau_{a_i,a_{i+1}} \geq \min \tau_{a_j,a_{j+1}}.$$

This is equivalent to the following inequality.

$$\forall\ 1 < i < n,\ \max \tau_{a_i,a_{i+1}} \geq \max_{0<j<i} \min \tau_{a_j,a_{j+1}}.$$

Therefore, a valid path from a_1 to a_n exists in G only if the inequality above holds, and the proof is complete.

IV. Graph Search Parallelization Strategy

Algorithm 1 Pseudocode of Parallel Search on MFAs

Input: MFA slices $\left\{G_{\text{mfa}}^{(m)}, ..., G_{\text{mfa}}^{(m+k)}\right\}$ and victim address set A_{vic}.
Output: Suspicious topologies $\left\{G_{\text{s}}^{(m)}, ..., G_{\text{s}}^{(m+k)}\right\}$.

```
1: $G_{\text{s}}^{(m)}, G_{\text{s}}^{(m+1)}, ..., G_{\text{s}}^{(m+k)} \leftarrow$ new $k+1$ empty topologies
2: for $i$ such that $0 \leq i \leq k$ do
3:     $A_{\text{s}} \leftarrow G_{\text{s}}^{(m+i-1)}$.AddressSet if $i > 0$ else $A_{\text{vic}}$
4:     parallel for $j$ such that $i \leq j \leq k$ do
5:         ${}^{i}G_{\text{s}}^{(m+j)} \leftarrow$ search on $G_{\text{mfa}}^{(m+j)}$ using $A_{\text{s}}$ as the source
6:         $G_{\text{s}}^{(m+j)} \leftarrow G_{\text{s}}^{(m+j)} \bigcup {}^{i}G_{\text{s}}^{(m+j)}$
7:     end for
8: end for
```

The details of graph search operations on a single MFA have been discussed in the Suspicious Topology Construction part of the paper's Section III-C. Here, we present the parallelization method for the search mentioned in the paper's Section III-D. The parallelization strategy can be depicted using the above algorithm. Let the relevant MFA slices be temporally ordered as $\left\{G_{\text{mfa}}^{(m)}, ..., G_{\text{mfa}}^{(m+k)}\right\}$. The search begins with using A_{vic} as the source to perform parallel searches across all MFA slices, yielding the suspicious sub-topologies $\left\{{}^{0}G_{\text{s}}^{(m)}, ..., {}^{0}G_{\text{s}}^{(m+k)}\right\}$, where the ${}^{0}G_{\text{s}}^{(m+i)}$ denotes the search result on $G_{\text{mfa}}^{(m+i)}$. Then, for $0 \leq i < k$, the system repeats the process: it takes all addresses from the union $\bigcup_{0 \leq j \leq i} {}^{j}G_{\text{s}}^{(m+k)}$ as the source and runs parallel searches on MFAs $\left\{G_{\text{mfa}}^{(m+i+1)}, ..., G_{\text{mfa}}^{(m+k)}\right\}$ to obtain suspicious sub-topologies $\left\{{}^{i+1}G_{\text{s}}^{(m+i+1)}, ..., {}^{i+1}G_{\text{s}}^{(m+i+k)}\right\}$. Finally, for $0 \leq i \leq k$, MFTracer merges suspicious sub-topologies $\left\{{}^{0}G_{\text{s}}^{(m+i)}, ..., {}^{i}G_{\text{s}}^{(m+i)}\right\}$ to obtain the suspicious topology $G_{\text{s}}^{(m+i)}$ as the search result on the MFA $G_{\text{mfa}}^{(m+i)}$. We now demonstrate that the parallel strategy yields identical results to the serial execution, proving its correctness. For ease of explanation, we use the notation $\mathcal{S}(G_{\text{mfa}}; G_{\text{s}})$ to refer to the suspicious sub-topology derived from searches on the MFA G using the addresses in the topology G_{mfa} as the source. Then for all $0 \leq i \leq k$, we have

$$\begin{aligned} G_{\text{s}}^{(m+i)} &= \bigcup_{0 \leq j \leq i} {}^{j}G_{\text{s}}^{(m+i)} \\ &= \bigcup_{0 < j \leq i} {}^{j}G_{\text{s}}^{(m+i)} \cup {}^{0}G_{\text{s}}^{(m+i)} \\ &= \bigcup_{0 < j \leq i} \mathcal{S}(G_{\text{mfa}}^{(m+i)}; G_{\text{s}}^{(m+j-1)}) \cup \mathcal{S}(G_{\text{mfa}}^{(m+i)}; A_{\text{vic}}) \\ &= \mathcal{S}\left(G_{\text{mfa}}^{(m+i)}; A_{\text{vic}} \cup \bigcup_{0 \leq j < i} G_{\text{s}}^{(m+j)}\right). \end{aligned}$$

The first equality is based on line 6 of algorithm 1. The third equality follows from line 3 and line 5. This equation demonstrates that $G_{\text{s}}^{(m+i)}$ is obtained by searching on $G_{\text{mfa}}^{(m+i)}$ using addresses from A_{vic} and all the suspicious topologies with earlier timestamps as the source, which completes the proof. We discuss the performance of this parallelization based on an empirical assumption: the subtasks in line 5 of algorithm 1 have similar time costs of $O(C)$. Then we derive that the total time cost for a serial algorithm is $O(C) \cdot (k+1)(k+2)/2 = O(Ck^2)$, while the parallel time cost is $O(Ck)$ from the loop in line 2.

V. Reserve Ratio ϵ Theoretical Analysis

The reserve requirement, also known as the cash reserve ratio (CRR) [3], [4], is a regulation set by central banks that mandates commercial banks to hold a certain percentage of their deposit liabilities in reserves. By adjusting the reserve requirement, central banks can influence the liquidity in the economy system.

Drawing inspiration from this, we apply a cap on the amount of funds that can flow out from each address to control liquidity distribution in the downstream network. Here, we discuss how the reserve ratio $\epsilon \in [0, 1)$ affects the breadth and depth of the downstream topology. For this, we estimate the depth and breadth that a specific amount of money can travel through under a small threshold T by assuming that each outflow for each address reaches its upper limit controlled by ϵ.

First, let's assume an upstream flow of $M \gg T$ reaches a downstream address, which is the flow we're to analyze. We call this address as the starting address and mark its depth as 1. Then $M(1-e)^n$ indicates the maximum outflow from an address at depth n. The following can be determined, where d indicates the maximum depth.

$$M(1-\epsilon)^d \geq T, M(1-\epsilon)^{d+1} < T.$$

We can then derive that

$$d = \left\lfloor \frac{\log(T/M)}{\log(1-\epsilon)} \right\rfloor.$$

The equation above demonstrates that d decreases as ϵ increases.

Since the starting address is randomly selected from the downstream set, the number of addresses at depth 2 indicates the breadth b, which satisfies the inequality below.

$$M\epsilon^{b-1}(1-\epsilon) \geq T, M\epsilon^{b}(1-\epsilon) < T.$$

Then we obtain that

$$b = \left\lceil \frac{\log(T/M) - \log(1-\epsilon)}{\log \epsilon} \right\rceil .$$

Given that $M \gg T$, then $\log(T/M)$ is a negative number with a significantly large absolute value. Since that ϵ is usually chosen to be less than 0.5, then we have $|\log(1-\epsilon)| \ll |\log(T/M)|$. Thus, we can arrive at the following approximation.

$$b \approx \left\lceil \frac{\log(T/M)}{\log \epsilon} \right\rceil .$$

This demonstrates that the breadth increases as ϵ increases.

Altogether, a lower reserve ratio leads to more liquidity among farther[1] addresses, increasing the downstream topology's depth, while a higher distributes liquidity to closer addresses and thus increasing the downstream topology's breadth. This affects the final coverage rates and precision. For criminals using extremely deep laundering paths, a lower ϵ increasing both the coverage rates and precision. For criminals using wider laundering paths, a higher ϵ is a better choice. In real-world use, multiple values can be selected to ensure a better coverage of illicit flows.

VI. Evaluation Results on ϵ

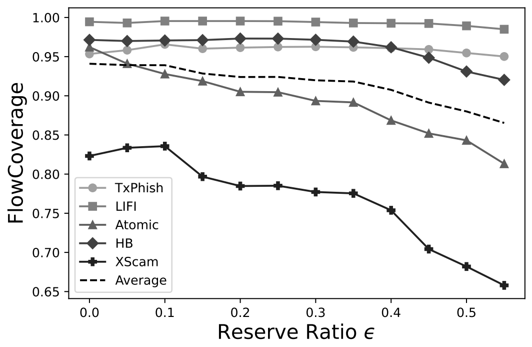
The reserve ratio ϵ is a configuration parameter for the money flow simulation algorithm described in the paper's Section III-E. Setting an appropriate ϵ value allows MFTracer to perform better in tracing tasks. Apart from theoretical analysis above, we also conduct experiments to examine the effect of different values of ϵ on the results. Figure 3 illustrates the curves of precision and two types of coverage rates as ϵ is set from 0% to 55%. For multi-perspective presentation, evaluation results are disaggregated by metrics and top-level incidents.

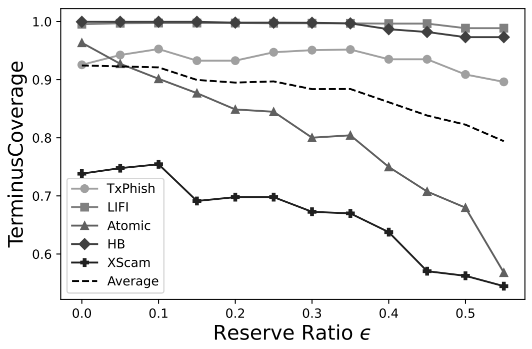
In all incidents, especially for Atomic and XScam, too large values of ϵ result in low coverage rates. Conversely, setting ϵ to 0% yields satisfactory coverage in all cases. Sometimes, such as the laundering cases of incidents TxPhish and XScam, reach their highest coverage rates when ϵ is set to a small positive value. Except for LIFI, the precision shows relatively small fluctuation. Overall, based on the empirical analysis, setting ϵ between 0% and 10% allows MFTracer to achieve the optimal effectiveness, balancing both two types of coverage rates and precision. For real-world tracing applications, multiple close values of ϵ can be selected to compare the differences in the respective resulting topologies.

VII. Dataset Construction Methodology

To rigorously evaluate the performance of illicit fund tracing systems, we constructed a ground-truth dataset LaunderNetEvm41 that reflects a diverse and realistic set of money laundering techniques. The dataset was collaboratively curated by a team of seven experts, including two members from our author group, three professionals from leading blockchain security firms, and two domain specialists from the broader

[1]These terms, closer and farther, are defined by the distance between addresses in the flow topology, i.e. the length of reachable path in the graph.





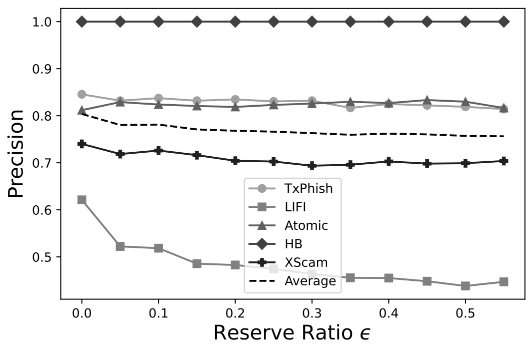


Fig. 3: FlowCoverage, TerminusCoverage and Precision as ϵ is set from 0% to 55%.

security community. Each expert has at least one year of hands-on experience in tracking crypto-related cybercrime.

In total, LaunderNetEvm41 contains 1,939 accounts involved in money laundering activities and 6,701 detailed illicit fund flow records—an order of magnitude larger than any previously disclosed (Bitcoin) illicit flow dataset we are aware of, collectively covering over US \$125 million in stolen assets. The laundering cases were carefully selected to cover a broad range of laundering behaviors and adversarial techniques employed by different attacker groups, allowing to test the generalizability, robustness and real-world applicability of tracing systems.

The construction process began with the collection of victim addresses and attack transaction hashes, primarily sourced from publicly available incident reports published by affected parties and security communities. These sources served as the origin points for tracing the downstream movement of stolen assets.

Experts utilized a combination of tools and platforms—such as Etherscan [5], Phalcon [6], Otterscan [7], and self-hosted full nodes—to gather on-chain transactional data and token transfer events.

The tracing process covered all types of fungible tokens, including native tokens and ERC-20 tokens, ensuring comprehensive coverage of the illicit asset flow. Experts analyzed transactions on a per-transaction basis. For each address identified as holding illicit funds, all subsequent outbound transactions were examined to determine how the assets were redistributed. These transactions were categorized into three general patterns: ① *Transactions to EOAs*: If the `to` address was an externally owned account (EOA), resulting in a native token transfer, the receiving address was marked as a potential downstream recipient and added to the pending analysis queue. ② *Transactions to ERC-20 Token Contracts*: In such cases, the expert team inspected the transaction's payload and analyzed the emitted token transfer events to determine fund movement: (i) If the transaction invoked the `transfer` function, the recipient of the token was added to the queue. (ii) If the transaction called `permit`, `approve`, or similar functions that delegated token spending rights to another address, the spender was recorded for further analysis. ③ *Transactions to Other Smart Contracts:* For these, a multi-tiered semantic analysis was conducted: (i) If the address matched a known public service address of a decentralized protocol, its function was analyzed using protocol documentation and public interfaces. (ii) If the contract was not recognized but its source code was available on platforms like Etherscan, experts manually audited the code using similarity detection [8], library usage analysis [9], and protocol-specific knowledge to interpret the contract's logic. (iii) If only bytecode was available (e.g., attacker-deployed contracts), experts used function selector databases [10] and dynamic analysis techniques such as reverse engineering [11], execution tracing [12], testnet deployment and black-box fuzzing [13] to infer contract behavior. Based on their domain-specific knowledge, experts used assistive tools to decipher smart contract semantics and identify the underlying fund movements resulting from the contract execution. The actual receivers of illicit money are marked as laundering addresses.

To ensure data accuracy and reduce bias, each tracing step was independently conducted by at least two experts. When interpretations differed, the findings were debated in peer-review sessions involving all seven members. Final decisions were made by majority vote. In cases of significant disagreement or ambiguity, deeper analysis was conducted using additional on-chain evidence, behavioral heuristics, and, where necessary, the recreation of suspicious transactions in controlled environments.

Once the full trace paths were established, the resulting dataset underwent a final round of verification. The two domain experts from the security community performed an extensive cross-validation process against publicly disclosed intelligence—including independent reports by other researchers, statements from victim entities, investigative reports from regulatory bodies, and analyses by external security firms. We list some of the references here [14], [15], [16], [17]. Any contradictions or discrepancies between our dataset and these sources triggered a re-examination of the relevant tracing paths. For especially complex or conflicting cases, we proactively reached out to the authors of the referenced reports for clarification or technical discussion.

This rigorous, expert-driven process ensures that the dataset reflects a realistic, high-fidelity mapping of illicit fund flows across various money laundering strategies. It serves not only as a benchmark for evaluating tracing systems, but also as a resource for further research in blockchain forensics and threat intelligence.

VIII. False Positives and Mitigation Strategies

While MFTracer demonstrates strong performance in flow coverage and precision, a limited number of false positives were observed in certain real-world tasks. To better understand these, we conducted a detailed comparison between MFTracer's tracing outputs and the ground truth dataset. Our analysis reveals two primary sources of false positives: *parameter sensitivity* and *token volatility*.

Parameter Sensitivity: Depth and Breadth of Topology. As noted in V and VI, the reserve ratio parameter directly affects the depth and breadth of the generated fund flow topology. An overly conservative or aggressive setting can inflate the scope of simulated flows, resulting in false positives. This issue is readily identifiable: by running MFTracer under multiple reserve ratio settings and comparing the topological differences, investigators can isolate spurious branches and quickly discard them. This strategy offers a fast and low-effort diagnostic mechanism for tuning the system's sensitivity.

Token Volatility and Temporal Value Drift. A more subtle source of false positives stems from *temporal valuation errors* due to fluctuations in token prices. MFTracer's transaction-level fund flow analysis relies on historical USD-denominated token prices to infer monetary value at the moment of each transaction. The valuation performs well for short-term flows, but may exhibit slight deviations in the context of long-term laundering strategies.

For example, suppose an attacker transfers \$100,000 worth of ETH to address A in December 2024. These assets remain idle for over two months before being moved entirely to address B in February 2025. If ETH depreciates during that time, and is only worth \$80,000 at the time of the outbound transaction, MFTracer's simulation will record a residual balance of \$20,000 at address A. As a result, subsequent unrelated transactions from A may be falsely flagged as laundering activity—since the system assumes part of the illicit funds remain there. This phenomenon arises from valuation drift—a mismatch between the token's market price at time t_1 (deposit) and t_2 (withdrawal). Similar distortions occur in the presence of price slippage. For instance, during panic-induced sell-offs, DeFi swaps may execute at unfavorable rates, causing a discrepancy between the input and output values. From a balance-sheet perspective, part of the "missing" value is effectively transferred to public liquidity pools. Without address-level contextual information, MFTRACER may incorrectly interpret such residual value flows as criminal transfers, leading to false labeling of common service addresses as laundering participants.

Manual Review and Mitigation Potential. As discussed in Section 4.2, such false positives can often be filtered out manually within a few dozen minutes by inspecting token types and public address tags. However, to minimize human review time, we propose two practical improvements: ① *Token-Aware Asset Balances.* Currently, MFTRACER maintains per-address USD-denominated balance sheets. We propose extending this to token-specific multi-asset ledgers. By appending token type metadata to the transaction-level fund flow outputs, MFTRACER can construct *asset-based* rather than value-based balance states. This approach would eliminate valuation drift caused by market fluctuations. Although this enhancement may marginally increase storage overhead, MFTRACER already achieves a 3.7×—9.4× improvement in storage efficiency, making this trade-off acceptable. ② *Integration with Address Labeling Systems.* To address errors stemming from price slippage and transfers to public infrastructure, we suggest integrating MFTRACER with blockchain address labeling services such as Etherscan [5], Moralis [18], MetaSleuth [19], or Chainlabs [20]. These platforms maintain curated labels for exchanges, bridges, liquidity pools, and other non-adversarial service addresses. Incorporating such data into MFTRACER 's inference process would allow the system to avoid flagging known public infrastructure as part of laundering trails—particularly in cases where residual value ”leaks” to public endpoints during volatile market conditions.

REFERENCES

[1] (2025) Adding liquidity vs. taking liquidity in trading. Referenced April 2, 2025. [Online]. Available: https://www.zeiierman.com/blog/adding-liquidity-vs-taking-liquidity-in-trading

[2] (2025) Providing liquidity. Referenced April 2, 2025. [Online]. Available: https://docs.uniswap.org/contracts/v2/guides/smart-contract-integration/providing-liquidity

[3] J. N. Feinman, “Reserve requirements: history, current practice, and potential reform,” *Fed. Res. Bull.*, vol. 79, p. 569, 1993.

[4] M. S. Gray, *Central bank balances and reserve requirements*. International Monetary Fund, 2011.

[5] (2025) Etherscan. Referenced April 2, 2025. [Online]. Available: https://etherscan.io/

[6] (2025) Blocksec phalcon. Referenced April 2, 2025. [Online]. Available: https://blocksec.com/phalcon

[7] (2025) Otterscan. Referenced April 2, 2025. [Online]. Available: https://github.com/otterscan/otterscan

[8] L. Chen, H. Wang, Y. Zhou, T. Wong, J. Wang, and C. Zhang, “Smarttrans: Advanced similarity analysis for detecting vulnerabilities in ethereum smart contracts,” *IEEE Transactions on Dependable and Secure Computing*, 2025.

[9] M. Huang, J. Chen, Z. Jiang, and Z. Zheng, “Revealing hidden threats: An empirical study of library misuse in smart contracts,” in *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering*, 2024, pp. 1–12.

[10] (2025) Welcome to the ethereum signature database. Referenced April 2, 2025. [Online]. Available: https://www.4byte.directory/

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

[12] S. S. Kushwaha, S. Joshi, D. Singh, M. Kaur, and H.-N. Lee, “Ethereum smart contract analysis tools: A systematic review,” *Ieee Access*, vol. 10, pp. 57 037–57 062, 2022.

[13] S. Wu, Z. Li, L. Yan, W. Chen, M. Jiang, C. Wang, X. Luo, and H. Zhou, “Are we there yet? unraveling the state-of-the-art smart contract fuzzers,” in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, 2024, pp. 1–13.

[14] zachxbt. (2022) Harmony bridge hack address report. Referenced April 2, 2025. [Online]. Available: https://www.chainabuse.com/report/0a2e8e00-00e2-4749-9b00-ceb1c6202d33

[15] ——. (2022) Tracking down discord & twitter phishing scammers. Referenced April 2, 2025. [Online]. Available: https://zachxbt.mirror.xyz/svL1N4xPLX5nXHr6Cw4KLsjRtaYHxm4MAqmFy6zx3cw

[16] tayvano. (2025) Atomic wallet hack report. Referenced April 2, 2025. [Online]. Available: https://dune.com/tayvano/atomic-wallet-hack

[17] (2024) How lazarus group laundered $200m from 25+ crypto hacks to fiat from 2020–2023. Referenced April 2, 2025. [Online]. Available: https://zachxbt.mirror.xyz/B0-UJtxN41cJhpPtKv0v2LZ8u-0PwZ4ecMPEdX4l8vE

[18] M. Team. (2025) Check wallet activity and get crypto address labels. Referenced April 2, 2025. [Online]. Available: https://developers.moralis.com/check-wallet-activity-and-get-crypto-address-labels/

[19] (2025) Metasleuth. Referenced April 2, 2025. [Online]. Available: https://metasleuth.io/

[20] (2025) Labelled datasets for your own analysis. Referenced April 2, 2025. [Online]. Available: https://chainlabs.ai/wallet-labels/