Tracing Fund Flow from a Blockchain Exploit: A BFS and DFS Approach

Peter Wongsoredjo - 13523039
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: peterwongsoredo@gmail.com , 13523039@std.stei.itb.ac.id

***Abstract*—In the wake of large-scale cryptocurrency exploits, a torrent of on-chain transactions is generated as malicious perpetrators attempt to launder stolen assets. The sheer volume and complexity of this data render manual analysis inefficient and unable to reveal macro patterns. This paper introduces and demonstrates a systematic, algorithm-driven framework for blockchain forensics analysis. By applying Breadth-First Search for broad network mapping, and Depth-First Search in a comparative study of the KuCoin and the PolyNetwork hacks, this research transforms a tangled bunch of transaction data into a systematic map of laundering operations. The results demonstrate that BFS excels at macro-analysis, systematically mapping the network's scale and objectively identifying its critical interaction hubs. In contrast DFS has proven to be essential for micro-analysis, tracing deep transaction chains to provide concrete evidence of specific laundering tactics. This paper concludes that the strategic application of both BFS and DFS is fundamental to transforming chaotic on-chain data into a structured and intelligible forensic map.**

***Keywords*—Blockchain, Exploits, BFS, DFS**

I. INTRODUCTION

The evolution of digital assets has undeniably altered the financial landscape, yet its rapid ascent has been tainted by the persistent threat of large scale security breaches. When a centralized exchange is compromised, like the case of the KuCoin hack in 2020, or the PolyNetwork hack in 2021, the aftermath is not silence, but an untraceable series of on-chain activity. Perpetrators initiate a complex, high velocity campaign to launder the stolen funds, through generating thousands of interconnected transactions designed to confuse and overwhelm authorities.

This presents the core challenge addressed by this paper. For a human analyst, navigating the web manually via a block explorer is an intractable task. The process itself is usually very slow, error-prone, and while it is capable of following a single thread, it fails to grasp the overarching structure of the operation. Analysts can see the individual trees, but the forest which is the attacker's grand strategy remains hidden. This limitation underscores the critical need for a more advanced, systematic approach.

Leveraging the inherent graph structure of blockchain transactions, this paper aims to demonstrate the transformative power of fundamental traversal algorithms in on chain forensics. By applying Breadth-First Search and Depth-First Search to past study cases, this study constructs an algorithmic framework to untangle the chaos in order to systematically map the scale of the laundering network, objectively identify its critical hubs, and trace its deepest pathways to their endpoints. The primary contribution of this research is a practical, replicable model that elevates on-chain analysis from a manual, reactive process to a structured, data-driven investigation.

II. THEORETICAL FRAMEWORK

A. Blockchain as a Graph

In the context of the Ethereum blockchain, we can model the ecosystem as a graph $G = (V, E)$, where V is the set of vertices and E is the set of edges.

- **Vertices**: Every Ethereum address, whether it is an Externally Owned Account controlled by a user or a Smart Contract, represents a node in the graph.
- **Edges**: A transaction from address A to address B creates a directed edge from node A to node B. This edge can be weighted by attributes such as the transaction value.

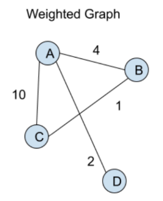


Figure 2.1 Weighted Graph
Source:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf

B. Graph Traversal Algorithms

To navigate these vast graphs, we use two fundamental traversal algorithms Breadth-First Search and Depth-First Search as they explore the graph without any prior knowledge about the distance to a goal.

1. Breadth-First Search (BFS)

   BFS is an algorithm that explores the graph layer by layer. The traversal begins at a source node v. First, node v is visited. Then, all unvisited neighbors of v are visited. Subsequently, the unvisited neighbors of those nodes are visited, and so on. To manage the order of visitation, BFS utilizes a queue data structure. Its purpose in this paper is to perform a macro-level analysis: to map the overall scale of the network and identify the most connected nodes (hubs).

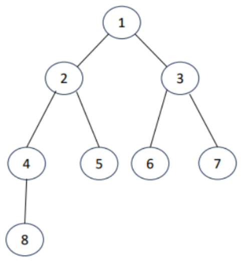


Figure 2.2 Breadth-First Graph
Source:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/13-BFS-DFS-(2025)-Bagian1.pdf

2. Depth-First Search (DFS)

   DFS explores the graph as far as possible along each branch before backtracking. The traversal begins at a source node v. First, node v is visited. Then, an unvisited neighbor w of v is visited, and a new DFS is initiated from w. When a node is reached where all its neighbors have been visited, the search backtracks to the previous node to explore other unvisited neighbors. Its purpose here is for micro-level analysis: to trace a specific money trail from a known hub to its ultimate conclusion, revealing layering tactics.

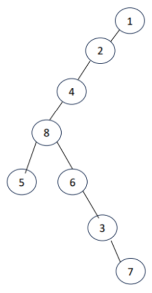


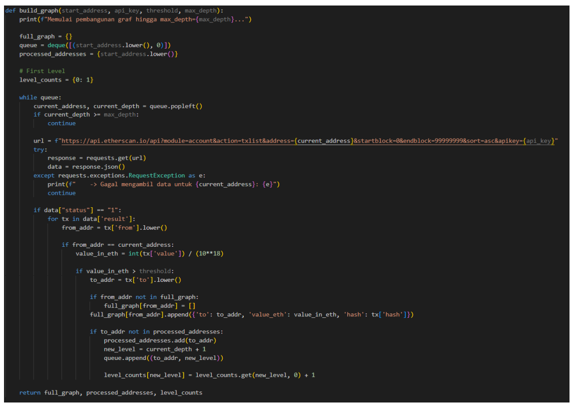
Figure 2.3 Depth-First Graph
Source:
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/13-BFS-DFS-(2025)-Bagian1.pdf

III. METHODOLOGY AND IMPLEMENTATION

The analytical framework for the implementation was constructed using the Python programming language, and centers on systematically fetching, processing, and analyzing on-chain data to build an intelligible map of fund flows from an initial source address. The entire process can be broken down into three parts: the graph construction, the analysis based on Breadth-First Search, and the analysis based on the Depth-First Search.

A. Graph Construction Implementation

1. **Iterative Data Fetching**: A queue data structure is initialized with the hacker's starting address as the first element. For each address dequeued, a targeted API call to the Etherscan public API is called to retrieve a list of all the transactions that the dequeued address has made. The graph was capped off at 3 depth level originating from 0 as the root address, in order to make sure that the addresses used are hubs used to obfuscate the analysts and not hot wallets owned by exchanges that also deal with a lot of the same funds.
2. **Data Pre-Processing**: To ensure the graph contains only meaningful data, each transaction retrieved from the API undergoes filtering. The transaction value is converted to a readable Ether unit, and then a value threshold (e.g. > 1 ETH) is applied, filtering out minor transactions such as fees, or contract interactions. Only transactions that pass this filter are considered for inclusion in the graph.
3. **Graph Data Structure**: The network is represented using an adjacency list model, using a Python dictionary. In this structure, each key corresponds to a source address (from_address). With each key is a list of dictionary objects, each represents a single outgoing transaction, such as: The destination address (to), the converted transaction value (value_eth), and the unique transaction identifier (hash).
4. **Caching Mechanism**: Since the graph construction process is time-consuming, a caching mechanism was implemented to streamline development and repeated analysis. After the graph is successfully built for the first time, the entire data structure is serialized into a JSON file (graph_cache.json). This way analysts can swap out analysis on certain addresses, or depth without having to rebuild old graphs.



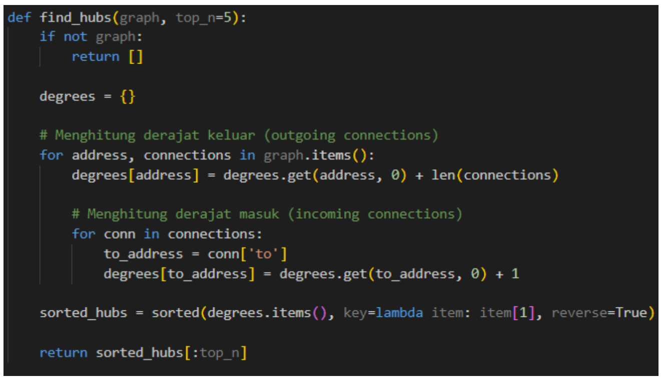
```python
def build_graph(start_address, api_key, threshold, max_depth):
    print(f"Memulai pembangunan graf hingga max_depth={max_depth}...")

    full_graph = {}
    queue = deque([(start_address.lower(), 0)])
    processed_addresses = {start_address.lower()}

    # First Level
    level_counts = {0: 1}

    while queue:
        current_address, current_depth = queue.popleft()
        if current_depth >= max_depth:
            continue

        url = f"https://api.etherscan.io/api?module=account&action=txlist&address={current_address}&startblock=0&endblock=99999999&sort=asc&apikey={api_key}"
        try:
            response = requests.get(url)
            data = response.json()
        except requests.exceptions.RequestException as e:
            print(f"    -> Gagal mengambil data untuk {current_address}: {e}")
            continue

        if data["status"] == "1":
            for tx in data['result']:
                from_addr = tx['from'].lower()

                if from_addr == current_address:
                    value_in_eth = int(tx['value']) / (10**18)

                    if value_in_eth > threshold:
                        to_addr = tx['to'].lower()

                        if from_addr not in full_graph:
                            full_graph[from_addr] = []
                        full_graph[from_addr].append({'to': to_addr, 'value_eth': value_in_eth, 'hash': tx['hash']})

                        if to_addr not in processed_addresses:
                            processed_addresses.add(to_addr)
                            new_level = current_depth + 1
                            queue.append((to_addr, new_level))

                            level_counts[new_level] = level_counts.get(new_level, 0) + 1

    return full_graph, processed_addresses, level_counts
```

Figure 3.2 Build_Graph Function

B. Graph Identification

Once the graph is constructed, it exists as a large, complex data structure. The next step is to apply analytical and visualization techniques to identify its most important structural features.

Hub Identification: To find out which addresses connect most to other nodes (addresses) on the transaction graph, a method called find_hubs is applied to iterate through the graph to calculate a "connectivity score" for every address by summing its in-degree and out-degree. By ranking addresses based on this score, the algorithm objectively highlights the most central and active nodes within the laundering operation.

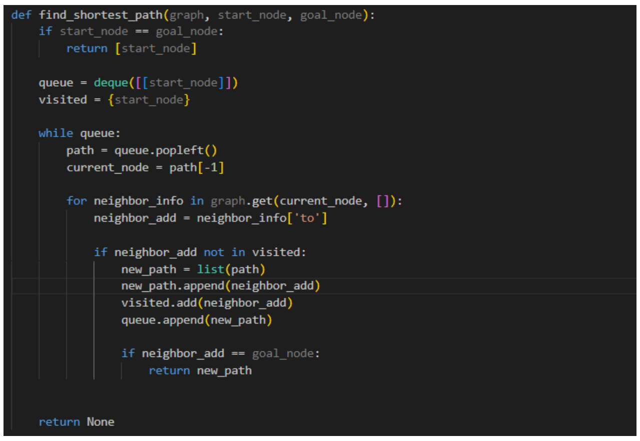


```python
def find_hubs(graph, top_n=5):
    if not graph:
        return []

    degrees = {}

    # Menghitung derajat keluar (outgoing connections)
    for address, connections in graph.items():
        degrees[address] = degrees.get(address, 0) + len(connections)

        # Menghitung derajat masuk (incoming connections)
        for conn in connections:
            to_address = conn['to']
            degrees[to_address] = degrees.get(to_address, 0) + 1

    sorted_hubs = sorted(degrees.items(), key=lambda item: item[1], reverse=True)

    return sorted_hubs[:top_n]
```

Figure 3.2 Find_Hubs Function

C. BFS-Based Analysis Implementation

To know which was the most efficient route the hacker used to reach a critical endpoint, a classic BFS search function, find_shortest_path, was implemented. This function takes the constructed graph, a start address and a target goal address as input. It terminates as soon as the first path to the goal node is found, which, due to the nature of BFS, is guaranteed to be one of the shortest paths in terms of transaction "hops".



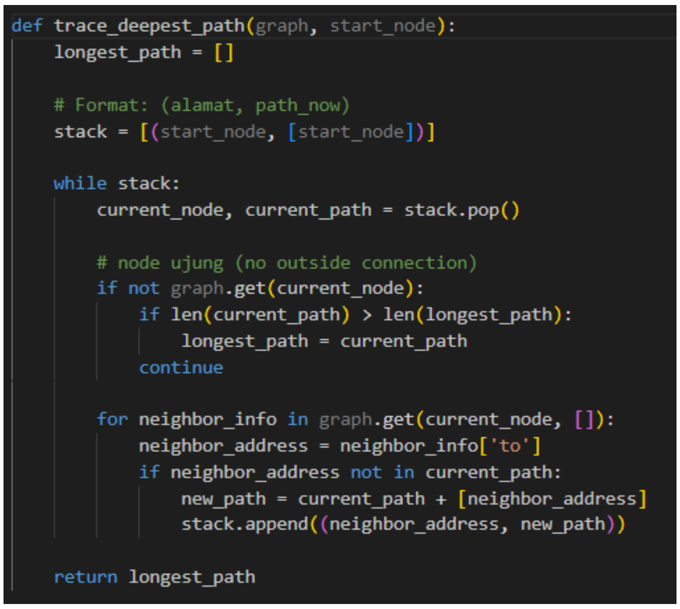
```python
def find_shortest_path(graph, start_node, goal_node):
    if start_node == goal_node:
        return [start_node]

    queue = deque([[start_node]])
    visited = {start_node}

    while queue:
        path = queue.popleft()
        current_node = path[-1]

        for neighbor_info in graph.get(current_node, []):
            neighbor_add = neighbor_info['to']

            if neighbor_add not in visited:
                new_path = list(path)
                new_path.append(neighbor_add)
                visited.add(neighbor_add)
                queue.append(new_path)

                if neighbor_add == goal_node:
                    return new_path


    return None
```

Figure 3.3 Find_Shortest_Path Function

D. DFS-Based Analysis Implementation

To conduct a micro-level analysis and uncover specific laundering tactics like layering, Depth-First Search (DFS) was implemented in the function trace_deepest_path_from_hub.

1. **Deep-Path Traversal:** This function utilizes a stack data structure to perform its search. Starting from an identified hub, it follows a single branch of transactions as far as it can go, pushing each new address onto the stack.
2. **Backtracking and Longest Path Discovery:** When a path reaches a terminal node (an address with no outgoing transactions in the graph), the function backtracks. The length of this completed path is then compared against the longest path found so far. If the current path is longer, it is stored as the new "deepest trail." This process continues until all possible paths from the starting hub have been explored.
3. **Providing Qualitative Evidence:** The output of this function is not just a number, but a concrete list of addresses that form a single, deep transaction chain. This chain serves as powerful, qualitative evidence of a deliberate layering strategy, demonstrating the attacker's effort to obfuscate the funds.



```python
def trace_deepest_path(graph, start_node):
    longest_path = []

    # Format: (alamat, path_now)
    stack = [(start_node, [start_node])]

    while stack:
        current_node, current_path = stack.pop()

        # node ujung (no outside connection)
        if not graph.get(current_node):
            if len(current_path) > len(longest_path):
                longest_path = current_path
            continue

        for neighbor_info in graph.get(current_node, []):
            neighbor_address = neighbor_info['to']
            if neighbor_address not in current_path:
                new_path = current_path + [neighbor_address]
                stack.append((neighbor_address, new_path))

    return longest_path
```

Figure 3.4 Trace_Deepest_Path Function

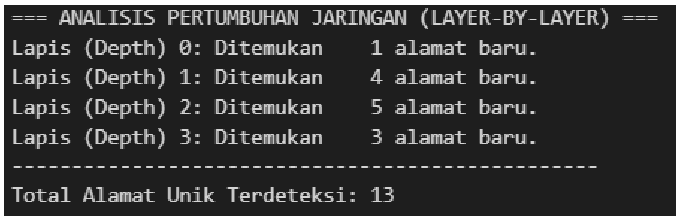
IV. RESULTS AND ANALYSIS

The implemented algorithm framework is applied into two famous case studies from the Blockchain world, through the cases of the KuCoin and the PolyNetwork hacks. This chapter presents the analysis of each case study individually, followed by a comparative discussion on the specific roles and strength each BFS and DFS algorithms have in the context of blockchain forensics.

A. Case Study 1: The KuCoin Hack (2020)

1. Incident Overview

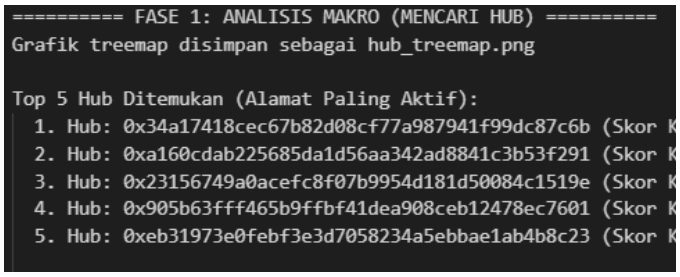
In September 2020, the KuCoin centralized exchange suffered a massive security breach resulting from the leakage of private keys to its hot wallets. The attackers drained approximately $280 million in various cryptocurrencies. What followed was a classic, large-scale money laundering operation using brute-force tactics to obfuscate the stolen assets. A key part of their strategy involved moving funds to anonymity protocols like Tornado.Cash to permanently break the transaction trail. In such a scenario, the speed at which funds can be traced to these critical endpoints is paramount for any potential intervention, creating a direct race against time between the attacker and the security teams. Here is what the framework found:



```
=== ANALISIS PERTUMBUHAN JARINGAN (LAYER-BY-LAYER) ===
Lapis (Depth) 0: Ditemukan    1 alamat baru.
Lapis (Depth) 1: Ditemukan    4 alamat baru.
Lapis (Depth) 2: Ditemukan    5 alamat baru.
Lapis (Depth) 3: Ditemukan    3 alamat baru.
--------------------------------------------------
Total Alamat Unik Terdeteksi: 13
```

Figure 4.1 Network Growth Analysis

The KuCoin hack serves as a textbook example of a classic laundering typology. Through the graph that was built from the primary hacker address of (0xeb31…), the algorithm is then able to reveal the scale and complexity of the exploit. The layer-by-layer growth was steady and expansive, indicating layering during the process of laundering the stolen funds.

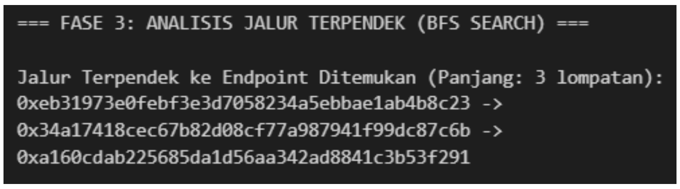


```
========== FASE 1: ANALISIS MAKRO (MENCARI HUB) ==========
Grafik treemap disimpan sebagai hub_treemap.png

Top 5 Hub Ditemukan (Alamat Paling Aktif):
  1. Hub: 0x34a17418cec67b82d08cf77a987941f99dc87c6b (Skor K
  2. Hub: 0xa160cdab225685da1d56aa342ad8841c3b53f291 (Skor K
  3. Hub: 0x23156749a0acefc8f07b9954d181d50084c1519e (Skor K
  4. Hub: 0x905b63fff465b9ffbf41dea908ceb12478ec7601 (Skor K
  5. Hub: 0xeb31973e0febf3e3d7058234a5ebbae1ab4b8c23 (Skor K
```

Figure 4.2 Macro Analysis on the KuCoin Hack

The find_hubs function was then applied to this sprawling graph, and crucially found out the top 5 hubs were in fact protocol contracts through the Tornado.Cash service, not a series of Externally Owned Accounts. With each being distributed a fair share of ETH as the attacker tries and disperse the funds to confuse analysts.

2. Algorithmic Application: Determining a Time-Critical Strategy



```
=== FASE 3: ANALISIS JALUR TERPENDEK (BFS SEARCH) ===

Jalur Terpendek ke Endpoint Ditemukan (Panjang: 3 lompatan):
0xeb31973e0febf3e3d7058234a5ebbae1ab4b8c23 ->
0x34a17418cec67b82d08cf77a987941f99dc87c6b ->
0xa160cdab225685da1d56aa342ad8841c3b53f291
```

Figure 4.3 BFS Search on the KuCoin Hack

In the immediate aftermath of the attack, the race against time had started, with analysts having to face a chaotic flurry of outgoing transactions, clear through all the "noise" that the attacker was making, and find the hub wallet directing all these movements. The core analytical challenge is thus one of speed and priority: to find the most direct path to a cash-out or anonymizing service before the attacker succeeds.

Through the BFS function, the algorithm was then able to pan out directly the perpetrator's strategy in moving their funds straight to an anonymity service. After building the initial transaction graph from the hacker's address, the find_shortest_path function was executed with the Tornado.Cash contract address (0xa160cd…) as the goal_node. The algorithm cuts through the complexity of the wider network, ignoring deeper, more convoluted paths, and focuses solely on finding the most direct route through the hub address of (0x34a17…) as a waypoint. This result demonstrates that despite the overall complexity of their operation, the attackers had a pre-planned, highly efficient route for rapidly anonymizing a portion of the stolen funds. BFS, therefore, allows us to not only trace the funds but to diagnose the attacker's high-priority, time-sensitive strategy.

3. Actionable Intelligence & Corporate Alignment

The output of the BFS Search is a critical piece of time-sensitive intelligence that directly informs real-world incident response and aligns with the actions taken by KuCoin and its partners.

- Forensics: In a live incident, analysts that are usually faced with hundreds of potential leads, can now prioritize the hub address 0x34a…. The shortest path by BFS immediately elevates the handful of addresses in that specific chain, enabling the team to immediately begin a deep dive investigation on these 2 - 3 wallets, saving time and resources.

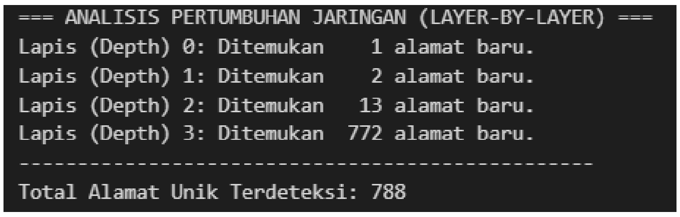
- Incident Response Teams: The information that was found, is what is needed for KuCoin to directly alert the wider ecosystem with an identified direct route to the mixer.

- Ecosystem Partners (Tether): This level of specificity is what enables decisive action. With the identification of a key hub in 0x34a… on a direct path to an endpoint is the exact evidence partners like Tether need to act upon. With the information given, Tether was then able to use its smart contract capability to freeze roughly $33 million in USDT held in the attackers' identified addresses. The BFS search directly addresses the race against time by finding the quickest path that authorities and partners must intercept.

B. Case Study 2: The Poly Network Hack (2021)

1. Incident Overview

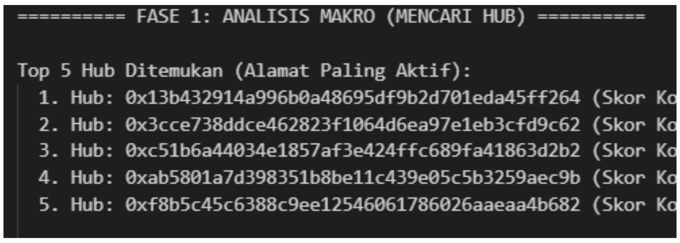
The Poly Network exploit of August 2021 was, at the time, the largest DeFi hack in history, involving over $600 million in assets. This was not a simple theft caused by compromised keys, but a sophisticated attack that exploited a logical vulnerability between smart contracts. The attacker, initiating the exploit on the Ethereum network from address 0xc8a6..., was able to bypass security verifications and arbitrarily withdraw funds. The subsequent on-chain activity revealed a modern and efficient laundering strategy, one that leveraged the existing financial infrastructure of the blockchain itself as a primary tool. Here is what the framework found:



```
=== ANALISIS PERTUMBUHAN JARINGAN (LAYER-BY-LAYER) ===
Lapis (Depth) 0: Ditemukan    1 alamat baru.
Lapis (Depth) 1: Ditemukan    2 alamat baru.
Lapis (Depth) 2: Ditemukan   13 alamat baru.
Lapis (Depth) 3: Ditemukan  772 alamat baru.
-----------------------------------------------
Total Alamat Unik Terdeteksi: 788
```

Figure 4.4 Network Growth Analysis

The initial graph construction immediately revealed a unique network topology. The layer-by-layer growth analysis showed a dramatic explosive growth between Depth 2 and 3, where the network ballooned from 13 to 772 new addresses. This pattern is not indicative of manual fund dispersal but is a clear signature of a single transaction interacting with a complex DeFi protocol or exchange hot wallet, which in turn interacts with hundreds of liquidity addresses in the background.

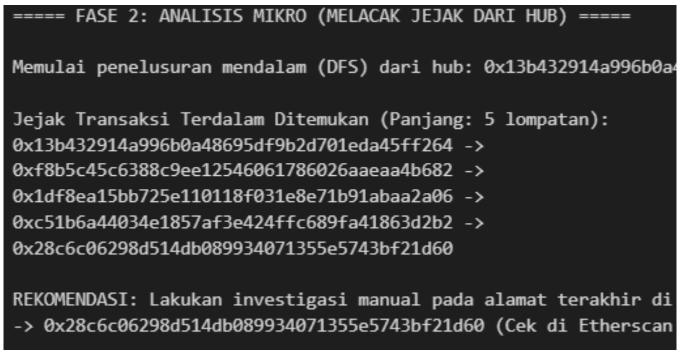


```
========== FASE 1: ANALISIS MAKRO (MENCARI HUB) ==========

Top 5 Hub Ditemukan (Alamat Paling Aktif):
  1. Hub: 0x13b432914a996b0a48695df9b2d701eda45ff264 (Skor Ko
  2. Hub: 0x3cce738ddce462823f1064d6ea97e1eb3cfd9c62 (Skor Ko
  3. Hub: 0xc51b6a44034e1857af3e424ffc689fa41863d2b2 (Skor Ko
  4. Hub: 0xab5801a7d398351b8be11c439e05c5b3259aec9b (Skor Ko
  5. Hub: 0xf8b5c45c6388c9ee12546061786026aaeaa4b682 (Skor Ko
```

Figure 4.5 Macro Analysis on the PolyNetwork Hack

The find_hubs analysis confirmed this, but with a critical nuance. The top-ranked "hubs" were identified as public infrastructure: a 1inch DEX Aggregator contract (0x13b4...) and an automated MEV Bot (0x3cce...) reacting to the market volatility. The true intermediary wallets controlled by the hacker (e.g., 0xc51b..., 0xf8b5...) were also identified by the algorithm, but with lower, more realistic connectivity scores (~50). This initial mapping proves the attacker's strategy was to "live off the land," using public protocols as both a tool and as on-chain camouflage, a key difference from the KuCoin strategy.

2. Algorithmic Application: Revealing a Protocol-Driven Strategy



```
===== FASE 2: ANALISIS MIKRO (MELACAK JEJAK DARI HUB) =====

Memulai penelusuran mendalam (DFS) dari hub: 0x13b432914a996b0a

Jejak Transaksi Terdalam Ditemukan (Panjang: 5 lompatan):
0x13b432914a996b0a48695df9b2d701eda45ff264 ->
0xf8b5c45c6388c9ee12546061786026aaeaa4b682 ->
0x1df8ea15bb725e110118f031e8e71b91abaa2a06 ->
0xc51b6a44034e1857af3e424ffc689fa41863d2b2 ->
0x28c6c06298d514db089934071355e5743bf21d60

REKOMENDASI: Lakukan investigasi manual pada alamat terakhir di
-> 0x28c6c06298d514db089934071355e5743bf21d60 (Cek di Etherscan
```

Figure 4.6 DFS Search on the PolyNetwork Hack

While the graph mapped the key actors, DFS was used to connect the dots and trace a complete narrative of the laundering lifecycle. The trace_deepest_path function, when initiated from one of the identified intermediary wallets (0xf8b5...), found a concise but highly informative 5-hop chain. This path revealed the full, modern laundering strategy:

1. Obfuscation via DeFi: Funds were first passed to a DeFi protocol like the 1inch DEX Router (0x13b4…) for an asset swap (e.g., ETH to a stablecoin), immediately breaking the chain of the original stolen asset.

2. Minimal Layering: The "cleaned" asset was then moved through a small number of private EOA wallets (0xf85c…, 0x1df8…) to create a small degree of separation

3. Exit to CeFi: Finally, the funds were deposited at the Binance CEX hot wallet endpoint (0x28c6…). The DFS trace was able to perfectly map the attacker's efficient, multi-stage strategy from the initiation until the transfer to cash-out the stolen funds, a process that would've been difficult to piece together manually.

3. Actionable Intelligence & Corporate Alignment

The output of the DFS indicates many possible scenarios for the analysts and authorities to cut off the perpetrator's movements, through a series of alerts and collaborations.

- Pinpointing the Root Cause: Analysis showed that the critical initial interaction was with a smart contract, not a simple transfer, the best move would be to immediately direct a security team to audit their own code for logical flaws. This aligns perfectly with Poly Network's real-world response, which was to patch the smart contract vulnerability that allowed the exploit in the first place, rather than focusing on compromised keys.

- Guiding Ecosystem Collaboration: With the information that the attacker is using the 1inch protocol to swap assets and routing them through wallets 0xf8b5... and 0xc51b... towards Binance, this allowed the Poly Network team to provide highly specific, credible information to all involved parties, the DeFi protocols being abused, and the CEXs being targeted as exits. This precision is essential for securing cooperation in a decentralized ecosystem and allows partners to freeze or monitor funds with high confidence.

- Informing a Strategic Response: The clarity provided by the algorithmic analysis, knowing exactly how their system was being used and where the funds were flowing, gave the PolyNetwork team a position of knowledge. This was a start for any form of strategic response. In their unique case, this detailed understanding of the attacker's actions and control over the funds was a powerful tool that enabled them to open direct communication with the hacker, which ultimately led to the negotiation and return of the assets.

C. BFS vs DFS in Blockchain Forensics

The comparative analysis of these two hacks provides a clear framework for understanding the distinct and complementary roles of BFS and DFS in on-chain investigation. They are not interchangeable tools; they are symbiotic instruments applied to solve different, equally important forensic problems. The clear difference between them is their approach to exploration, which can be framed as **Efficiency** versus **Exhaustiveness**.

1. BFS: Efficiency and Immediacy

The primary prowess of BFS lies in its ability to find the shortest path. In a time-critical incident response, this is the single most valuable piece of intelligence. By exploring layer-by-layer, BFS guarantees that it finds the most direct route to a target first. As demonstrated in the KuCoin case, this allows an analyst to immediately identify the attacker's "express lane" to an endpoint like Tornado.Cash, enabling partners to attempt to freeze assets before they are anonymized. This is a direct counter to the attacker's speed.

Furthermore, the layer-by-layer exploration itself is a powerful analytical tool. The explosive growth in nodes between layers, as seen in the Poly Network case, is a clear signature of an attacker interacting with a complex protocol. This allows analysts to instantly diagnose the type of attack and understand how the perpetrator uses many layers of protocol interactions, not just wallets, to obfuscate their trail.

2. DFS: Exhaustiveness and Strategy

While BFS focuses on the what and how fast, DFS is uniquely suited to uncover the how and how complex. Its strength lies in its exhaustive, depth-first traversal of the graph. In the context of this framework, its application is to deconstruct an attacker's complete methodology.

By systematically exploring all possible paths from a hub to an endpoint, we move beyond just one "shortest" path. This reveals the attacker's full modus operandi (M.O.). For instance, discovering that the Poly Network hacker created multiple, redundant routes to the same Binance wallet proves a deliberate attempt to complicate analysis. This exhaustive map gives security teams a complete list of intermediary wallets to blacklist, effectively "cutting off" all known routes simultaneously. It transforms the analysis from simply following one trail to understanding the attacker's entire strategic playbook, which is crucial for building more resilient defenses in the future.

V. CONCLUSION

This paper has successfully demonstrated that a systematic, algorithmic approach, implementing Breadth-First Search (BFS) and Depth-First Search (DFS), can effectively untangle the complexity of a major on-chain security exploit. By conducting a comparative analysis of the KuCoin (2020) and Poly Network (2021) hacks, this research has not only mapped the flow of illicit funds but also revealed the prowess in each algorithm when it comes to giving a point of view for the forensics.

BFS excels as a first responder's tool, providing the macro level intelligence needed for a time-critical response by rapidly mapping the network's scale and identifying the most efficient path to critical endpoints. Conversely, DFS serves as the investigator's tool for deep analysis, exhaustively exploring all possible routes to uncover the complexity and methodology of the attacker's strategy. Ultimately, this research affirms that as criminal tactics on the blockchain evolve, so too must our analytical strategies, with foundational algorithms like BFS and DFS serving as the indispensable bedrock of modern digital investigation.

Code Link at Github

https://github.com/PeterWongsoredjo/Tracing-Fund-Flow-from-a-Blockchain-Exploit-A-BFS-and-DFS-Approach

Video Link at YouTube

https://youtu.be/FOQLFAbpxAI

Acknowledgment

The author would like to express utmost gratitude to Mrs. Dr. Nur Ulfa Maulidevi, S.T, M.Sc., as the lecturer of IF2211 Algorithm Strategy for her valuable guidance and passion to spread knowledge, which in turn contributes greatly to the completion of this paper. The author would also like to thank all his friends and family members for their moral support throughout the process of researching and writing this paper.

References

[1] https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf (Accessed 22 June 2025)
[2] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/13-BFS-DFS-(2025)-Bagian1.pdf (Accessed 22 June 2025)
[3] https://www.kucoin.com/id/announcement/en-the-latest-updates-about-the-kucoin-security-incident (Accessed 21 June 2025)
[4] https://coinmarketcap.com/academy/article/poly-network-suffers-612m-hack (Accessed 21 June 2025)
[5] https://dedaub.com/blog/poly-network-hack/ (Accessed 21 June 2025)
[6] https://telcoin.medium.com/kucoin-hack-post-mortem-7e15d1cc4bd4 (Accessed 21 June 2025)

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 1 Juni 2025

[illegible]

Peter Wongsoredjo 13523039