ConMan: A Connection Manipulation-based Attack Against Bitcoin Networking

Wenjun Fan*§, Sang-Yoon Chang†, Xiaobo Zhou†, and Shouhuai Xu†
*Department of Communications and Networking, School of Advanced Technology
Xi'an Jiaotong-Liverpool University, Suzhou, Jiangsu, P. R. China, 215123
Email: Wenjun.Fan@xjtlu.edu.cn
†Department of Computer Science, College of Engineering and Applied Science
University of Colorado Colorado Springs, Colorado Springs, United States, CO 80918
Email: {schang2, xzhou, sxu}@uccs.edu

***Abstract*—Bitcoin is a representative cryptocurrency system using a permissionless peer-to-peer (P2P) network as its communication infrastructure. A number of attacks against Bitcoin have been discovered over the past years, including the Eclipse and EREBUS Attacks. In this paper, we present a new attack against Bitcoin's P2P networking, dubbed ConMan because it leverages *con*nection *man*ipulation. ConMan achieves the same effect as the Eclipse and EREBUS Attacks in isolating a target (i.e., victim) node from the rest of the Bitcoin network. However, ConMan is different from these attacks because it is an *active* and *deterministic* attack, and is more effective and efficient. We validate ConMan through proof-of-concept exploitation in an environment that is coupled with real-world Bitcoin node functions. Experimental results show that ConMan only needs a few minutes to fully control the peer connections of a target node, which is in sharp contrast to the tens of days that are needed by the Eclipse and EREBUS Attacks. Further, we propose several countermeasures against ConMan. Some of them would be effective but incompatible with the design principles of Bitcoin, while the anomaly detection approach is positively achievable. We disclosed ConMan to the Bitcoin Core team and received their feedback, which confirms ConMan and the proposed countermeasures.**

***Index Terms*—Cryptocurrency, Bitcoin, P2P Network, Eclipse, Connection Manipulation, Anomaly Detection**

I. Introduction

Bitcoin [1] uses a decentralized peer-to-peer (P2P) network and computational Proof-of-Work (PoW) consensus for mining blocks and maintaining a distributed ledger of transactions. The financial incentive (1 BTC $\approx$ US$32,222.50 as of Jan. 28, 2021) for mining Bitcoins has drawn worldwide attackers' attentions. A number of Bitcoin vulnerabilities have been identified, which allow attackers to break down the Bitcoin operations and manipulate transactions [2], [3]. Also, the network-layer attacks (e.g., Eclipse Attack [4], Bitcoin Partitioning Attack [5], and EREBUS Attack [6]) can be waged to control the peer connections of the target Bitcoin nodes, disrupt the PoW consensus protocol, and make the attackers gain illegal rewards via double-spending [7], [8], selfish mining [9], [10], and block withholding [11]–[13].

The attacks that attempt to control the peer connections of Bitcoin nodes can be categorized into two classes: *node-based* and *network-based*. On the one hand, node-based attacks attempt to isolate some target node(s) from the rest of the Bitcoin network. Two attacks in this class are Eclipse Attack [4] and EREBUS Attack [6], which poison the *peer-tables* of some target nodes (by replacing their legitimate peer identifiers with malicious ones controlled by the attacker). These attacks allow the attacker to control a target node's Bitcoin functions and benefit from this control. On the other hand, the network-based attacks introduce mechanisms targeting multiple nodes. The Bitcoin Partitioning Attack [5] achieves such feat by using BGP hijacking to manipulate the inter-Autonomous System (AS) routing so that the partitioned or victim network's packets go through the AS controlled by the attacker, effectively isolating the victim network from the Bitcoin Mainnet.

In this paper, we focus on node-based attacks against Bitcoin. To clarify the relationship between the new attack and the known node-based attacks (i.e., Eclipse Attack [4] and EREBUS Attack [6]), we characterize the latter as follows: (i) they require the attacker to poison the peer-tables of the target nodes; (ii) they are probabilistic, meaning that their success probability depends on the fraction of the peer-identifiers in a target node's peer-tables that are poisoned by the attacker (i.e., these peers are under the attacker's control); (iii) they take effect only after a victim node selects a number of poisoned peer-identifiers, which may force the attacker to wait for a long period of time. The preceding characteristics suggest that both Eclipse Attack [4] and EREBUS Attack [6] are *opportunistic*. The state-of-the-art is that Bitcoin Core version 0.18.0 (released on May 2, 2019) has fixed a number of *peer-table* related bugs, which effectively renders Eclipse Attack [4] infeasible. Moreover, the Bitcoin Core version 0.20.0 (released on June 3, 2020) has provided several patches against EREBUS Attack [6], but their effectiveness is not clear at the time of writing.

This paper proposes a new node-based attack, dubbed *Con*nection *Man*ipulation (ConMan)[1]. Unlike Eclipse Attack [4] and EREBUS Attack [6], which require the attacker to poison a target node's peer-table, a ConMan

§This work was done while Dr. Fan was a Postdoctoral Research Associate at University of Colorado Colorado Springs.

[1]According to `www.collinsdictionary.com`, "*A con man is a man who persuades people to give him their money or property by lying to them*," which coincides with ConMan's consequence that a target or victim node believes that its neighbors are legitimate, while they aren't.

attacker controls a target node's peer connections as follows: (i) monopolizing the target node's *inbound* peer connections, meaning that these connections are initiated by the attacker to the target node; (ii) hijacking a target node's *outbound* peer connections, which are the connections initiated by the target node to some legitimate nodes; and (iii) spoofing these legitimate outbound peers to the target node by replying to the target node's requests with expected messages. As a consequence, ConMan can cause the same kind of damages as what can be caused by Eclipse Attack [4] and EREBUS Attack [6]. However, ConMan is more powerful in the following sense: ConMan is in multiple orders of magnitude *efficient* because it is *deterministic*, meaning that it can take effect immediately; whereas, the other two are *probabilistic*, namely that their success depends on the occurrence that the target node selects some peers that are controlled by the attacker. ConMan's attack power is rooted in what it exploits, namely directly hijacking the TCP connections between a target node and its outbound peers without the target node's notice (otherwise, the target node would re-select its outbound peers, which would be out of the attacker's control).

The rest of the paper is organized as follows. Section II presents a primer on Bitcoin P2P network. Section III gives an overview of ConMan. Section IV presents the proof-of-concept of ConMan and the experimental results. Section V explores countermeasures against ConMan. Section VI reviews the related work. Section VII concludes the paper.

II. A Primer on Bitcoin Network Security

Bitcoin network. The Bitcoin network is a P2P overlay built on top of the TCP protocol. That is, two Bitcoin nodes establish a TCP connection and then use Bitcoin's `VERSION` and `VERACK` messages to establish an Bitcoin session for application-layer interactions. Since Bitcoin P2P network is permissionless, there is no admission control. Moreover, all Bitcoin messages are transmitted in plaintext through TCP connections with neither confidentiality nor integrity protection because the current Bitcoin network does not use any cryptographic protection for Bitcoin P2P communications [5]. The lack of cryptographic protections makes the Bitcoin P2P network vulnerable to TCP spoofing and TCP hijacking. TCP spoofing means an attacker can inject spoofed TCP segments [14], [15], which requires the attacker to impersonate an end node at both the transport layer and the network layer. TCP hijacking means that an attacker can impersonate one end node to the other end node of an established TCP connection without the latter's notice.

In the Bitcoin P2P network, the peer identifier is a pair of IP address and TCP Port number, denoted by `[IP:Port]`. This means that a Bitcoin node with one IP address can initiate multiple peer connections through multiple ports. Since a Bitcoin node always listens on the TCP Port 8333 to receive inbound peer connections, a malicious node can wage a Sybil attack [16] by initiating multiple connections (via multiple private Port numbers between 49,152 and 65,535) to a target node on Port 8333. Each node maintains two peer-tables for tracking the IP addresses of its peers in the network. A `new` table contains the IP addresses it has received from the `ADDR` messages but has yet to connect, and a `tried` table maintains the IP addresses to each of which it has successfully made an outbound connection. The `new` table originally has 16,384 slots and the `tried` table originally has 4,096 slots, both of which are later expanded to cope with Eclipse Attack [4].

Bitcoin's *private* vs. *public* nodes. A Bitcoin node can be a *private* node if it uses a private IP address (i.e., IP address behind Network Address Translation or NAT), or a *public* node if it uses an IP address which is publicly routable. Note that a private node will not have any inbound peer connections, unless there is at least one peer that resides in the same private network. Currently, a private node can have at most 11 outbound peer connections, including: (i) 2 *blocks-only* connections that only transmit `BLOCK` messages (all 26 message types refer to Bitcoin Developer[2]); and (ii) 1 *feeler* connection that connects/disconnects quickly to test if an IP address in the `new` table is valid or not. More precisely, a new feeler connection is created in every two minutes[3], by selecting an IP address randomly from the `new` table to test if the IP address is valid or not. On the other hand, every feeler connection only persists for α seconds, where $\alpha \in [1, 2]$ is a random number. Thus, the feeler connection is transient rather than persistent, meaning that a private node often has 10 persistent outbound peer connections. In contrast, a *public* node has 11 outbound peer connections and up to 117 inbound peer connections (i.e., up to 128 connections in total, or 127 persistent connections other than the feeler connection).

Bitcoin's *inbound* vs. *outbound* peer connection. If Bitcoin node A initiates a P2P connection to node B, then B is an *outbound* peer of node A, and A is an *inbound* peer of node B. The inbound/outbound connection types are tracked in a distributed manner; for example, node A registers its connection to node B as an inbound peer connection because A initiated the connection. Note that the Bitcoin protocol treats inbound and outbound connections differently. For example, in order to mitigate Eclipse Attack [4], the current Bitcoin Core makes nodes randomly select outbound peers without any bias in regards to timestamps. This can mitigate Eclipse Attack [4] unless a node's peer-table is already fully occupied by malicious peers controlled by the attacker.

III. Overview of the ConMan Attack

A. Basic Idea

As reviewed above, the Bitcoin protocol allows a (public) node to have: (i) up to 117 inbound peer connections that are initiated by other nodes to the target node in question; and (ii) up to 10 outbound peer connections that are initiated by the target node. Recall that the Bitcoin network nodes are connected to each other through a permissionless P2P network, and the communications between them are not protected by cryptographic means (i.e., neither confidentiality nor integrity

[2]Available on: `https://developer.bitcoin.org/reference/p2p_networking.html`, Nov.30, 2020.

[3]In fact, the timestamp corresponding to the time at which the next feeler connection will be added is computed by the function known as `PoissonNextSend(FEELER_INTERVAL)`.

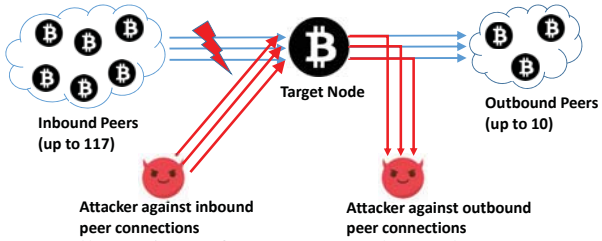

Fig. 1: Illustration of ConMan, where the target node's *inbound* connections are initiated by the attacker and *outbound* connections are hijacked by the attacker.

is assured). This means that in principle, an attacker can intercept, disrupt, hijack, and control any node's inbound and outbound peer connections to achieve the effect of Eclipse Attack [4] and EREBUS Attack [6]. This explains the basic idea of ConMan.

As illustrated in Figure 1, the *target* node is defined as the eclipse-victim, and the ConMan attack exploits the lack of cryptographic protection to intercept the connections between the target node and its peers, as follows: (i) monopolizing the inbound peer connections to the target node, which can be legitimately achieved by the attacker; (ii) hijacking the outbound peer connections of a target node, which can be achieved by spoofing the original legitimate outbound peer; and (iii) replying to the target node's requests with expected messages that to benefit the attacker.

ConMan is different from the Eclipse Attack [4] and EREBUS Attack [6] as follows. (i) ConMan causes the same kind of damages as Eclipse Attack [4] or EREBUS Attack [6] does, but much more *efficiently*. This is because ConMan is a *deterministic* attack and takes effect immediately; whereas, the other two are *probabilistic* because as mentioned above, their success depends on when the target node selects the IP addresses controlled by the attacker as its outbound peers. (ii) The approaches they leverage are different. ConMan manipulates and hijacks TCP connections, while making the hijacking stealthy and not noticed by the target node (otherwise, the target node would re-select its outbound peers, which is out of the attacker's control). In contrast, Eclipse Attack [4] poisons a target node's peer-table with IP addresses under its control; and EREBUS Attack [6] poisons a target node's peer-table by exploiting adversary AS to spoof shadow IP addresses (which can be any IP addresses whose target-to-IP routes go through the adversary AS [6]). (iii) The defenses against them are different. As we will present later, the defenses that are effective against Eclipse Attack [4] and EREBUS Attack [6] are not effective against ConMan.

B. Requirements for ConMan to Succeed

To make ConMan succeed, the following three requirements must be satisfied: (i) the attacker must be able to obtain the TCP states of the target node's inbound and outbound connections, which is necessary for preparing TCP spoofing and hijacking; (ii) the attacker must be able to monopolize the target node's inbound peer connections; and (iii) the attacker must be able to hijack the target node's outbound peer connections. As illustrated in Figure 1, satisfying these requirements makes the target node effectively communicate only with the attacker, despite that the target node "thinks" it is communicating at least with some legitimate peers. This coincides with the meaning of the term *con man* mentioned above. In what follows we show how these requirements can be satisfied.

Satisfying Requirement 1: Obtaining the TCP states of the target node's inbound and outbound connections. To break the target node's inbound connections and hijack its outbound connections with legitimate nodes, the attacker must be able to obtain the TCP states of these inbound and outbound connections, namely the `seqnums` and `acknums`. We call an attacker who can obtain such information by an *sniffing-capable* attacker or *sniffing-incapable* attacker otherwise (i.e., the latter must infer the TCP states through other means).

A sniffing-capable attacker can monitor all of the networking packets from/to the target node to learn the 4-tuple `<source IP, source Port, destination IP, destination port>` information and to derive the current `seqnum` and `acknum` of those connections. A sniff-capable attacker can be instantiated as: (i) a node that resides in the same local area network as the target node, such as a 802.11 wireless network or a wired network using promiscuous mode; (ii) a node resides on the path of the connection between the target node and a legitimate peer, which is the same as the attacks studied in [5], [17], [18], namely that the attacker manipulates routes via BGP hijacking. Note that BGP hijacking can be achieved by compromising the routers/switches near the target node, or by leveraging the AS(es) under the attacker's control to spoof shadow IP addresses as shown in [6]. It is worth mentioning that what is required for ConMan to succeed is strictly weaker (so that it is easier to fulfill and thus greater security risk) than what can be achieved by the attacks studied in [5], [6], [17], [18].

While the preceding discussion has justified how a sniffing-capable attacker can obtain the TCP states of the target node's inbound and outbound connections, it is still interesting to explore whether a sniffing-incapable attacker can achieve the same. In contrast, a sniffing-incapable attacker can wage the ConMan attack, by leveraging the side-channels related to the shared variables of certain implementations of the TCP stack, such as the global ACK Challenge Counter [19] and the IP-ID Counter [20], [21].

Satisfying Requirement 2: Monopolizing the target node's inbound peer connections. To isolate the target node from the rest of the Bitcoin network, the attacker must take over the target node's inbound connections first. For this purpose, the attacker needs to break the target node's existing inbound peer connections and then create inbound connections to the target node. This can be achieved by an attacker who has a computer with multiple Sybil identities/connections or controls multiple computers, because the current Bitcoin protocol allows inbound peer connections to come from a same IP address with different TCP ports.

Figure 2 illustrates how the attacker, Mallory, monopolizes the target node Alice's inbound connections. Suppose Alice

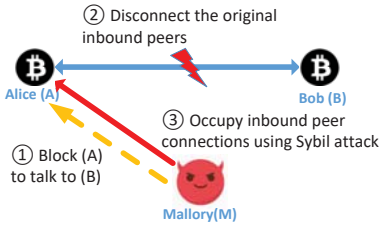

Fig. 2: Illustrating how ConMan attacker Mallory "takes over" Alice's inbound connection with legitimate peer Bob.

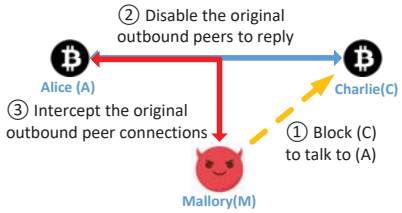

Fig. 3: Illustrating how ConMan attacker Mallory hijacks Alice's outbound connection with legitimate peer Charlie.

has an inbound connection with Bob. ① Mallory wages an appropriate attack (e.g., TCP Reset) to cause Alice to disconnect her inbound connection with Bob. ② Alice tears down the inbound connection with Bob. ③ Mallory initiates an inbound connection to Alice. By repeating the preceding ①-③ as many times as needed, Mallory can successfully monopolize Alice's (up to) 117 inbound connections. In practice, Mallory would not need to tear down 117 inbound connections, because the pool of inbound connections of a node is often not full, which is less than 117 most of the time. Further, if the pool of inbound connections is full, the attacker can evict the benign existing inbound connection even much easier, i.e., without needing TCP Reset attack, but just keep creating (malicious) inbound connections. This is so because starting Bitcoin Core version 0.17.0 [22], a node is required to start evicting (benign) inbound connection when reaching 117 inbound connections.

Satisfying Requirement 3: Hijacking the target node's outbound peer connections. To isolate the target node from the rest of the Bitcoin network, the attacker must control the target node's outbound connections as well. Since the outbound connections are initiated by Alice randomly selecting peers, Mallory cannot guarantee that the peers under her control will be selected by Alice. This explains why Mallory needs to hijack and maintain Alice's outbound connections with legitimate peers, while assuring that Charlie cannot interfere with the attack.

Figure 3 illustrates how Mallory (i) hijacks target node Alice's outbound peer connections with legitimate node, say Charlie, and (ii) spoof Charlie to communicate with Alice. The preceding (ii) is important because Alice will establish a new outbound connection otherwise. ① Mallory wages an attack (e.g., TCP Reset or flooding-based DoS) against Charlie to prevent him from sending data to Alice. ② Charlie stops to send data to Alice. ③ Mallory spoofs Charlie to communicate with Alice by hijacking the Alice-Charlie connection. To make this hijacked connection active, Mallory must be able to receive the data sent by Alice to Charlie and send Alice meaningful data in return.

IV. Prototype and Experiments

A. Prototype Implementation

In order to validate ConMan, we set up a real Bitcoin network in a lab environment. Specifically, we create a real Bitcoin node and connect it to the public Bitcoin Mainnet as the target node. The target node runs Bitcoin Core (software version Satoshi 0.20.0 and protocol version 70015) in its default setting. We develop a set of dedicated programs in another computer as the attacker node. Both the attacker node and the target node run on virtual machines (VMs) having the same specifications (i.e., Ubuntu 18.10 64-bit operation system, Intel Core i7 4GHz CPU, 4GB memory, and Intel PRO/1000 MT Desktop network adapter).

The programs running in the attacker node leverages Bitcoinlib and Scapy as well as the TCP Reset attack as aforementioned. Specifically, we use the python-bitcoinlib [23] (i.e., a Bitcoinlib-based program) to facilitate the Bitcoin networking protocol for the attacker node to create up to 117 Sybil identifiers (i.e., `[IP:Port]` pairs). We apply Scapy [24] which is a tool that can satisfy the sniffing-capable attack requirement mentioned above, to (i) learn the states of the TCP states of the target node's peer connections and (ii) craft spoofing data to manipulate the target connections.

We also specify the experimental attack environment, which is connected to the real-world Bitcoin network. For ethical research, we stress that the target node's other inbound and outbound peers than our attacker node are real-world Bitcoin nodes which reside out of our lab environment, and we make sure that our attack experiment is only waged against the target node in our lab (i.e., there is no side-effect on the operation of the real-world Bitcoin network). Corresponding to the sniffing-capable threat model, we make the attacker node be able to sniff the target node's packets to learn the TCP `seqnum` and `acknum`. To mimic real attacks, we consider various networking conditions, including arbitrary delays between the connected peers and the packet drop rate. We also consider the absence of a ConMan attacker (dubbed the *no-attack* case), which serves as the baseline for validating the Bitcoin node operations and providing reference measurements for our experimental analysis.

B. Baseline Experiment in the Absence of ConMan

We took an empirical measurement of the number of peer connections of a public Bitcoin node during 25 consecutive days (10/10/2020-11/4/2020), where the node ran the by-default code/setting and there was no ConMan attacker. As stated, a Bitcoin node by design should have 10 outbound connections and up to 117 inbound connections; we call this *worst-case (protocol)* because it is specified by the protocol and represents the worst-case scenario to the attacker (i.e., demanding maximum attack effort). Our measurement result (see Figure 4) shows that the node can have less than 10 outbound connections due to its disconnection and re-connection to the Bitcoin Mainnet (e.g., the Bitcoin node had less than 10 connections corresponding to 2.162% of the lifetime of the experiment). During these 25 days, the node has a maximum of 82 total connections (occurring on 10/19/2020),

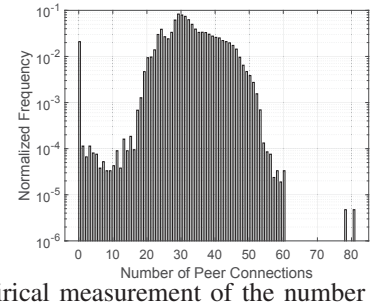

Fig. 4: Empirical measurement of the number of connections (log scale) when there are no ConMan attacks.

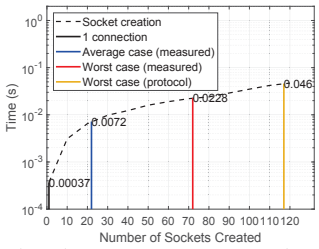

Fig. 5: ConMan's computing overhead for attacking inbound connections.

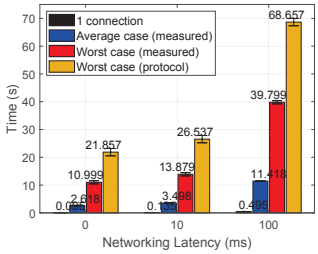

Fig. 6: ConMan's networking overhead for attacking inbound connections.

including 10 outbound and 72 inbound connections; we call this *worst-case (measured)* because it represents the worst-case scenario in practice. On average, the node has 32 connections (32.00046 to be precise) on a daily basis, including 10 outbound and 22 inbound connections; we call this the *average-case (measured)* scenario.

C. ConMan Attack Experiment and Results

1) The ConMan Attack Experiment: Built on the prototype, the attacker can monopolize the target node's inbound connections and hijack the target node's outbound connections. Specifically, this is conducted as follows. (i) The attacker node generates and sends a TCP Reset packet to the target node to disconnect an inbound peer, or to the outbound peer to make the outbound peer connection half-closed (i.e., making the original outbound peer connection open at the target node's end). (ii) The attacker creates inbound peer connections to the target node, and periodically sends Bitcoin `PING` messages to the target node at a certain interval (e.g., once a minute) to keep the inbound peer connections alive. (iii) The attacker node generates a `PONG` message to reply to `PING` message sent by the target node, by spoofing the original outbound peer to prove the pinging outbound peer that is still alive. This is needed because the Bitcoin Core disconnects any connection when the corresponding peer does not respond to the `PING` message within 20 minutes by default. To make this realistic, we let the Scapy-based program construct the corresponding `PONG` message according to the received `PING` message as in the real world. This is important because Bitcoin Core (protocol version 60001) makes `PONG` send back the nonce that is received in `PING`, while noting that `PONG` has the same format as `PING` except the packet header.

2) Experimental Results: Figure 5 plots the ConMan attacker node's computing overhead for attacking inbound

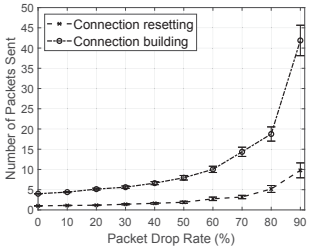

Fig. 7: The packets needed for ConMan with respect to the drop rate. The "connection resetting" (in TCP Reset) equally applies to the attacks against inbound and outbound connections, but the "connection building" only applies to the attacks against inbound connections.

connections. We observe that the average computing overhead is 0.37 milliseconds per victim inbound connection, where average is taken between when the attack starts and when the attack succeeds in isolating the target node. Figure 6 plots the ConMan attacker node's networking overhead in terms of the incurred network latency, including the TCP three-way handshake time and the Bitcoin-application level `VERSION`/`VERACK` handshake time. These latencies are measured in three networking environments, namely 0 ms vs. 10 ms vs. 100 ms delay in one-way network transmission. We observe, e.g., that in the average case, the measured networking delay is correspondingly 2.618 seconds vs. 3.498 seconds vs. 11.418 seconds, incurred by attacking and taking over all of the inbound connections. In Figure 7, the "connection building" curve shows the number of packets that are needed for the ConMan attacker to send including the TCP three-way handshake and the `VERSION` and `VERACK` message exchange, but excluding the TCP Reset packets, while the "connection resetting" curve shows the number of the needed TCP Reset packets, with respect to varying packet drop rate. We see that the worse the networking condition, the greater the packet drops, and the greater the number of packets the ConMan attacker needs to send.

Also, we observe that the attacker node's CPU usage grows to 13% in the case of *worst case (protocol)*, 10% in the case of *worst case (measured)*, and 8% in the case of *average case (measured)* (and 6% for the one connection case as reference) respectively. The *average case* CPU usage decreases more drastically than the other two cases because it has a smaller number of inbound peer connections to build. After the connection generation stage, all the cases' resource usages become stable along with time, i.e., after 16.82 seconds the *average case* CPU usage stays at 0.3%, after 27.49 seconds the *worst case (measured)* CPU usage stays at 0.7%, and after 31.13 seconds the *worst case (protocol)* CPU usage stays at 1.1%. Along with the CPU usage testing, we test the memory usages for all these cases. We observe that all these cases have 0.3% memory usage in the stable stage other than the *worst case (protocol)* that increases the memory usage from 0.3% to 0.6% after 10 seconds.

For attacking outbound connections, the ConMan attacker constructs a single TCP Reset packet to the outbound peer of

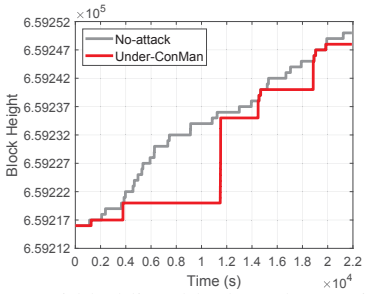

Fig. 8: ConMan withholding blocks. The vertical jumps and the momentary synchronization is from the eclipse slippage.

the target node (i.e., Charlie in Figure 3). This is sufficient in an ideal networking environment with no packet drop. In order to understand what happens if there are packet-drops in a realistic networking environment, Figure 7 plots the number of TCP Reset packets that need to be sent by the ConMan attacker with varying packet drop rates. The attacker hijacks an outbound connection by simply sending a `PONG` message to respond to the target node's (i.e., Alice's) `PING` message.

Overall, We observe that the overhead of ConMan for controlling all the inbound and outbound peer connections is in the order of magnitude of minutes, which is much smaller than what is incurred by Eclipse Attack [4] and EREBUS Attack [6].

Summarizing the preceding discussion, we draw:

Insight 1: The ConMan attack is relatively easy to wage and takes effect just in minutes (rather than tens of days).

D. Implication of ConMan

To show the damage that can be caused by ConMan, we now show its consequences when applied to delay the `BLOCK` delivering to the target node.

After an attacker wages ConMan against a target node, the attacker can deliver all of the relevant messages, except `BLOCK`, to the target node to maintain its inbound and outbound connections, which is equivalent to the *block withholding* attack [25]. When a spoofed TCP Reset packet that is sent to an outbound peer of the target node gets dropped (for whatever reason), an *eclipse slippage* event occurs, causing that at least one outbound peer not to be controlled by the attacker, and therefore the target node can still synchronize with the Bitcoin ledger.

Figure 8 plots the block height growing pattern in the absence vs. presence of ConMan. We observe that during the period of 6 hours for a measurement study, 34 blocks are generated, and 10 eclipse slippage events occur under ConMan, causing the target node to get synchronized with the Bitcoin ledger. Nevertheless, the attacker can quickly isolate the target node after each synchronization. In our experiment, the target node has the Bitcoin ledger during a total of 3,804 seconds, or $3,804/21,917$=17.356% of the 6-hour experiment.

Insight 2: Eclipse slippage can occur and a target node can occasionally synchronize with the Bitcoin ledger, but ConMan can isolate a target node as frequently as the attacker wants.

V. Exploring Countermeasures

This section explores countermeasures against ConMan. A status of the countermeasures up-to-date is presented in Table I, which shows that simple countermeasures, which are often effective against Eclipse Attack [4] and EREBUS Attack [6], would not be effective against ConMan.

For ethical disclosure, we sent our manuscript to the Bitcoin Core team [26] on January 13, 2021 to inform them the ConMan attack as well as the potential countermeasures. We received their response on February 4, 2021 by acknowledging ConMan and resonating the following countermeasures, despite that some of them might be effective while are not compatible with the design principle behind Bitcoin. However, they confirmed that the anomaly-detection-based countermeasure is positively achievable and have suggested that a better solution is to incorporate it into a future version of Bitcoin Core. Therefore, the countermeasure via anomaly detection will be elaborated more in this section.

A. Effective Countermeasures Are not Compatible with Bitcoin's Design Principles

We attribute the root cause of ConMan to (i) Bitcoin network is permissionless and (ii) Bitcoin communication is not cryptographically protected. This observation suggests the following countermeasures, which are effective but are not compatible with Bitcoin's design principle.

In order to prevent ConMan, it suffices to prevent the attacker from hijacking Bitcoin's peer connections. One solution to this problem is to employ cryptography to protect the integrity of the communications (e.g., each connection is associated with a cryptographic key of a message authentication scheme), because confidentiality is not necessary. However, any attempt to incorporate cryptography, e.g., transport layer security (TLS), appears to be incompatible with fundamental principle behind Bitcoin. An alternate approach is to use some trusted and centralized nodes, which can be leveraged to offer additional assurance. However, this approach also conflicts with the decentralization and permissionless principle of Bitcoin.

B. Countermeasure via Anomaly Detection

We build the anomaly detection approach to resist against ConMan. Though this is a passive countermeasure, the Bitcoin Security team positively considers that such a direction is achievable and beneficial for Bitcoin security.

1) Approach Overview: The proposed anomaly detection approach focuses on analyzing the traffic information rather than tracking the peer identifier that creates the anomalies, since in the permissionless Bitcoin P2P network, the spoofing and Sybil attacks would make an identifier-based detector ineffective. Also, the detection approach does not require modifying the Bitcoin Core implementation itself; the Bitcoin nodes implement a detector as a module that can monitor misbehaving traffic and report alerts. Our countermeasure identifies and presents the detection features specific to ConMan, in addition to building on those which are popularly used for anomaly detection in general.

TABLE I: Status of Countermeasures Up-to-Date.

Countermeasure	Proposed by	Developed	Eclipse'15	EREBUS'20	ConMan (ours)
Deterministic random eviction	Eclipse'15	Yes	Effective	Ineffective	Ineffective
Random peer selection	Eclipse'15	Yes	Effective	Ineffective	Ineffective
Test before evict	Eclipse'15	Yes	Effective	Ineffective	Ineffective
Feeler Connections	Eclipse'15	Yes	Effective	Ineffective	Ineffective
More buckets	Eclipse'15	Yes	Effective	Ineffective	Ineffective
More outbound connections	Eclipse'15, EREBUS'20	Yes	Effective	Effective	Ineffective
Ban unsolicited ADDR messages	Eclipse'15	No	Effective	Ineffective	Ineffective
Diversify incoming connections	Eclipse'15	No	Limited Effective	Ineffective	Limited effective
Anomaly detection	Eclipse'15, ConMan	No	Effective but passive	Effective but passive	Effective but passive
Limited scalability	EREBUS'20	No	Effective	Effective	Ineffective
Centralization	EREBUS'20, ConMan	No	Effective	Effective	Effective
Selecting peers with AS topology information	EREBUS'20	Yes	Ineffective	Effective	Ineffective
Eviction policy protecting some peers	EREBUS'20	No	Effective	Effective	Ineffective
Single protected peer connection	ConMan	No	Ineffective	Ineffective	Effective

2) Key Features of Detection: The features are based on what the victim node experiences specifically due to ConMan, and we use a blend of features which are useful for detection during and after the attack occurs.

The following feature detects the anomolous networking while the attack is undergoing to reach the eclipse state:

Peer Connection Growth Rate (r_{conn}) This feature represents the creation rate of the peer connections (including both inbound and outbound ones) to the target node. We know that if the target node is under normal network context without ConMan, its peer connection increases gradually, while if it is under ConMan, since the attacker needs to monopolize the inbound peer connections as soon as possible and then maintains such connections, the whole peer connection number would grow drastically in a certain duration. In other words, the connection request arrivals would be more sporadic than the normal case, making the instantaneous connection growth rate an effective feature for detection.

The following features detect the networking anomalies after the ConMan attack, including the attacker behavior of withholding/delaying blocks and transactions:

Block Height Growth Rate (r_{blk}) This feature is for growth rate of the block height, which is very ConMan specific because of the implication of ConMan. It actually counts both the arrival BLOCK and CMPCTBLOCK messages (e.g., approximately 0.1 blocks per minute on average). It counts only the new blocks which contribute to the ledger growth, as opposed to the redundant blocks which are already in the ledger. It can detect the ConMan attack's implication like block withholding and selfish mining, since such implications change the rate of the block height growth of the target node, e.g., slow or sporadic block creation.

TX Message Rate (r_{tx}) This feature indicates the TX message count per minute (rate), which has the similar use of the block height growth rate in order to detect the ConMan attack's implications.

Inter-message Distribution (Λ) This feature represents the relative count distribution among all messages. That is used for detecting the anomalous networking behavior carried out by the inbound peer connections controlled by the attacker, because the attacker only sends PING message with a constant rate to the target node for maintaining the connections while never transmits any other types of messages. That anomalous traffic will change the inter-message distribution.

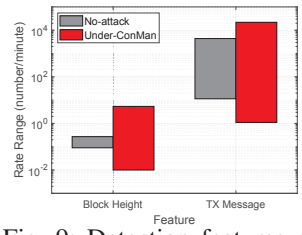

Fig. 9: Detection features of the block height growth rate (r_{blk}) and the TX message rate (r_{tx}).

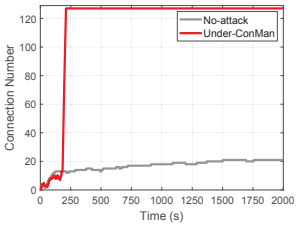

Fig. 10: Detection feature of the connection growth rate (r_{conn})

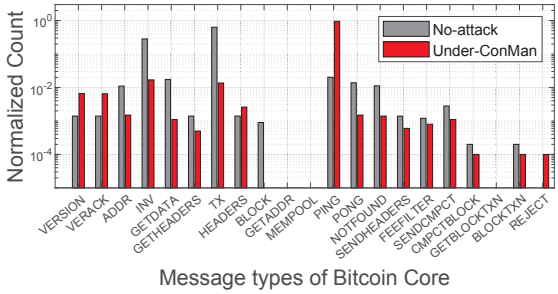

Fig. 11: Anomaly detection feature for comparing the frequency of messages (Λ): the no-attack case and the under-ConMan case have the correlation coefficient $\rho = 0.025$.

3) Anomaly Detection Performance: We establish the detection threshold by training the anomaly detector so that ConMan is detected if it falls outside of the threshold range. After training the model with statistical analysis using the normally collected data for approximate 35 hours, the threshold of r_{blk} is $\tau_{r_{blk}} = [0.09, 0.27]$ messages per minute, the threshold of r_{tx} is $\tau_{r_{tx}} = [11.4, 4395.6]$ messages per minute, the threshold of Λ (i.e., the similarity using correlation coefficient) is $\tau_{\Lambda} = 0.993$, and the threshold of r_{conn} is $\tau_{r_{conn}} = [0.71, 19.1]$ connections per minute.

Detection Accuracy With the reference profile and the fixed thresholds of the proposed detection features, we detect the anomaly networking traffic launched by ConMan. Figure 9 shows that when the target node is under ConMan, the block height growth rate can decrease to 0.01 blocks per minute, and that is lower than the lowerbound of $\tau_{r_{blk}}$. That is because the attacker withholds the new blocks. Also, we see that the block height growth rate can increase to 5.39 blocks per minute, which is even higher than the upperbound of $\tau_{r_{blk}}$. That is because the withheld blocks are transmitted to

TABLE II: Comparison of latency using different approaches.

Phase (sec)	Ours	LR	GB	RF	SVM	DNN	OC-SVM	AE
Training	6.15×10^{-4}	1.81	37.41	30.09	14744.31	498.09	2422.84	1414.86
Testing	4.64×10^{-4}	0.04	0.27	0.84	52.58	1.03	7538.69	21.37

the target node on a sudden, e.g., when the eclipse slippage event occurs. Also, the `TX` message rate has the similar result. In fact, the under-ConMan case can decrease both of the rates even lower and increase them even higher, this figure just shows one example of ConMan. Figure 10 presents that the under-ConMan case increases the target node's peer connections to the maximum number in a quite short time, in particular, it creates 117 inbound peer connections using 28.42 seconds in this example, which varies the connection growth rate significantly from the no-attack case. Figure 11 illustrates that the under-ConMan case has the `PING` message dominating the inter-message distribution, whereby the `PING` message takes 94.53% of the normalized count of the overall messages, it is 45.57 times greater than the `PING`'s normalized count in the no-attack case. That is because the under-ConMan case prevents the other messages to arrive to the target node (especially, the `BLOCK` and `TX` messages) after the attack succeeds, while the inbound peer connections controlled by the attacker keep sending the `PING` messages. Thus, the similarity of the no-attack case and the under-ConMan case becomes very low, i.e., $\rho = 0.025$ and it is largely lower than $\tau_\Lambda = 0.993$.

We find that our detection accuracy performance is 100% because the tested under-ConMan cases are not deliberately sophisticated and intelligent against the proposed detection approach. However, an intelligent attacker which controls its traffic for avoiding the detection, whereas the attack would have a smaller impact on the victim (e.g., the application data withholding reduces), and thus, our detection scheme has the security effect of mitigating the attack.

Detection Cost Overhead In addition, we present that our detection approach can just rely on lightweight statistical analysis rather than machine learning algorithms. We compare the time latencies of both training and testing between our approach and the machine learning (ML)-based approaches used for anomaly detection in the Bitcoin context described in the literature [27]–[32], including Logistic Regression (LR), Gradient Boosting (GB), Random Forest (RF), Support Vector Machine (SVM), Deep Neural Network (DNN), One-Class SVM (OC-SVM) and AutoEncoder (AE). Table II shows that our approach ("Ours" in the table) is at least four orders of magnitudes efficient than the ML-based approaches depending on the certain ML algorithm which is adopted.

VI. Related Work

We divide the related studies into two categories: those attacking the Bitcoin networks (i.e., the Eclipse and Partitioning attacks), and those serving as building-blocks to ConMan (i.e., TCP Hijacking).

Prior studies on the Eclipse attack. As discussed, the Eclipse attack isolates a target node from the rest of the Bitcoin network. The attack implementation against Bitcoin [4] requires the attacker to control hundreds of bots and IP addresses; the attack implementation against Ethereum [33] requires the attacker to use a single IP address. EREBUS Attack [6] leverages compromised ASes as man-in-the-middle between the target nodes and the rest of the Bitcoin network. Defenses against Eclipse include: (i) detecting suspicious block timestamps, which however takes 2-3 hours to take effect [34]; (ii) using a gossip protocol to make the Bitcoin clients connect to the servers to obtain the strongest blockchain view [34]; and (iii) using blockchain anomaly detection [35]. The ConMan attack is different from the Eclipse attacks because it is (i) *deterministic* rather than *probabilistic*, (ii) *resource-efficient* rather than demanding the attack to control hundreds of IP addresses, and (iii) *fast* rather than needing to await the target node to reboot. As a consequence, it is harder to defend because both the countermeasures mentioned above [34], [35] are not effective against ConMan. We attribute this difficulty to the fact that the the Bitcoin network uses plaintext TCP connections for performance reasons.

Prior studies on the Partitioning attack. The Partitioning attack attempts to partition the Bitcoin network into multiple isolated networks. Apostolaki et al. [5] showed how to achieve this by (i) poisoning the routing tables of the ASes near to the target network of Bitcoin victim nodes and (ii) forcing Bitcoin connections to go through those ASes controlled by the attacker, while assuming that the attacker can wage the BGP interception attack which is not always possible [17]. The ConMan attack is different from the Partitioning attack because it (i) targets Bitcoin nodes without incurring any partition, (ii) does not need to carry out the much harder-to-achieve route manipulations. Moreover, an effective defense against Partitioning is to carefully configure and protect the routing system by sanitizing and controlling the routing table of ASes. However, this defense is not effective against the ConMan attack because ConMan does not rely on redirecting the routing paths of the target's peer connections, meaning that preventing route manipulation cannot defend against ConMan.

Prior studies on the TCP Hijacking attack. This attack exploits the TCP protocol to intercept a TCP connection, by learning the expected `seqnum` and `acknum` and then injecting data into the connection. This attack is easy to wage by an *on-path* attacker that resides on the path of the connection. This attack can also be waged by an *off-path* attacker, which does not reside on the path of the connection, via one of the following methods: (i) exploiting a side-channel, which is known as the *global IP-ID counter* of the TCP protocol either in the Microsoft Windows implementation [20] or in the Linux implementation [21], to guess the expected `seqnum` and `acknum`; (ii) compromising a firewall to learn the `seqnum` of the packets passing through it [36]; (iii) exploiting a side-channel related to a host's packet counter to infer the `seqnum` [37]; (iv) leveraging a browser-based automated puppet script running on at the victim node to learn `seqnum` and `acknum` [38]; (v) using a side-channel of the TCP stack in the Linux implementation to infer the expected `seqnum` and `acknum`, which is made possible by the fact that

the challenge-ACK counter is a global variable shared by all TCP connections (as per RFC 5961) [19]. All these attacks can be leveraged by ConMan in a modular/plug-and-play fashion.

VII. Conclusion

We proposed a new attack against Bitcoin, dubbed ConMan, which manipulates the TCP connections used by Bitcoin nodes. ConMan can achieve the same effect as Eclipse Attack and EREBUS Attack against Bitcoin, but is much more efficient. We presented some countermeasures that are effective against Eclipse Attack and EREBUS Attack while are not effective against ConMan. Also, We proposed some countermeasures that would be effective but are not compatible with the design principles of Bitcoin. Further, we elaborated the anomaly detection as the more promising countermeasure for defending against ConMan. We disclosed ConMan to the Bitcoin Core team, and received their quick response (in three weeks) to confirm our attack and suggest research directions that would lead to employable solutions.

Acknowledgment

This work was supported in part by NSF Grant #2122631 (#1814825) and by Colorado State Bill 18-086.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[2] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten, "Sok: Research perspectives and challenges for bitcoin and cryptocurrencies," in *2015 IEEE Symposium on Security and Privacy*, 2015, pp. 104–121.

[3] M. Conti, E. Sandeep Kumar, C. Lal, and S. Ruj, "A survey on security and privacy issues of bitcoin," *IEEE Communications Surveys Tutorials*, vol. 20, no. 4, pp. 3416–3452, 2018.

[4] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg, "Eclipse attacks on bitcoin's peer-to-peer network," in *24th USENIX Security Symposium (USENIX Security 15)*, August 2015, pp. 129–144.

[5] M. Apostolaki, A. Zohar, and L. Vanbever, "Hijacking bitcoin: Routing attacks on cryptocurrencies," in *2017 IEEE Symposium on Security and Privacy (S&P)*, 2017, pp. 375–392.

[6] M. Tran, I. Choi, G. J. Moon, A. V. Vu, and M. S. Kang, "A stealthier partitioning attack against bitcoin peer-to-peer network," in *IEEE Symposium on Security and Privacy (S&P)*, 2020.

[7] G. O. Karame, E. Androulaki, M. Roeschlin, A. Gervais, and S. Capkun, "Misbehavior in bitcoin: A study of double-spending and accountability," *ACM Transactions on Information and System Security (TISSEC)*, vol. 18, no. 1, 2015.

[8] S. Zhang and J. Lee, "Double-spending with a sybil attack in the bitcoin decentralized network," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 10, pp. 5715–5722, Oct 2019.

[9] K. Nayak, S. Kumar, A. Miller, and E. Shi, "Stubborn mining: Generalizing selfish mining and combining with an eclipse attack," in *2016 IEEE European Symposium on Security and Privacy (Euro S&P)*, March 2016, pp. 305–320.

[10] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," *Commun. ACM*, vol. 61, no. 7, pp. 95–102, Jun. 2018.

[11] Y. Kwon, D. Kim, Y. Son, E. Vasserman, and Y. Kim, "Be selfish and avoid dilemmas: Fork after withholding (faw) attacks on bitcoin," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 195–209.

[12] M. Walck, K. Wang, and H. S. Kim, "Tendrilstaller: Block delay attack in bitcoin," in *2019 IEEE International Conference on Blockchain (Blockchain)*, 2019, pp. 1–9.

[13] S.-Y. Chang, Y. Park, S. Wuthier, and C.-W. Chen, "Uncle-block attack: Blockchain mining threat beyond block withholding for rational and uncooperative miners," in *Applied Cryptography and Network Security*, 2019, pp. 241–258.

[14] M. De Vivo, G. O. de Vivo, and G. Isern, "Internet security attacks at the basic levels," *ACM SIGOPS operating systems review*, vol. 32, no. 2, pp. 4–15, 1998.

[15] B. Harris and R. Hunt, "Tcp/ip security threats and attack methods," *Computer communications*, vol. 22, no. 10, pp. 885–897, 1999.

[16] J. Dinger and H. Hartenstein, "Defending the sybil attack in p2p networks: Taxonomy, challenges, and a proposal for self-registration," in *First International Conference on Availability, Reliability and Security (ARES'06)*, 2006.

[17] H. Ballani, P. Francis, and X. Zhang, "A study of prefix hijacking and interception in the internet," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 4, p. 265–276, Aug. 2007.

[18] M. Saad, V. Cook, L. Nguyen, M. T. Thai, and A. Mohaisen, "Partitioning attacks on bitcoin: Colliding space, time, and logic," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, 2019, pp. 1175–1187.

[19] Y. Cao, Z. Qian, Z. Wang, T. Dao, S. V. Krishnamurthy, and L. M. Marvel, "Off-path tcp exploits: Global rate limit considered dangerous," in *25th USENIX Security Symposium (USENIX Security 16)*, 2016, pp. 209–225.

[20] lkm, "Remote blind tcp/ip spoofing," 2007. [Online]. Available: http://phrack.org/issues/64/13.html

[21] X. Feng, C. Fu, Q. Li, K. Sun, and K. Xu, "Off-path tcp exploits of the mixed ipid assignment," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 1323–1335.

[22] "Attempt to evict connection when incoming slots are full," Released: 2019. [Online]. Available: https://github.com/bitcoin/bitcoin/blob/0.17/src/net.cpp#L1128-L1136

[23] L. Jongeneel, "Python bitcoin library," Released: September 8, 2020. [Online]. Available: https://pypi.org/project/bitcoinlib/

[24] P. Biondi and the Scapy community, "Scapy project," Released: September 28, 2020. [Online]. Available: https://scapy.net/

[25] M. Rosenfeld, "Analysis of bitcoin pooled mining reward systems," *CoRR*, vol. abs/1112.4980, 2011. [Online]. Available: http://arxiv.org/abs/1112.4980

[26] "Bitcoin security team." Feb. 2021. [Online]. Available: https://bitcoincore.org/en/contact/

[27] H. Sun Yin and R. Vatrapu, "A first estimation of the proportion of cybercriminal entities in the bitcoin ecosystem using supervised machine learning," in *2017 IEEE International Conference on Big Data (Big Data)*, Dec 2017, pp. 3690–3699.

[28] M. Harlev, H. Sun Yin, K. Langenheldt, R. Mukkamala, and R. Vatrapu, "Breaking bad: De-anonymising entity types on the bitcoin blockchain using supervised machine learning," in *Proceedings of the 51st Hawaii International Conference on System Sciences (HICSS)*, United States, 2018, pp. 3497–3506.

[29] H. Tang, Y. Jiao, B. Huang, C. Lin, S. Goyal, and B. Wang, "Learning to classify blockchain peers according to their behavior sequences," *IEEE Access*, vol. 6, pp. 71 208–71 215, 2018.

[30] J. Hirshman, Y. Huang, and S. Macke, "Unsupervised approaches to detecting anomalous behavior in the bitcoin transaction network," *3rd ed. Technical report, Stanford University*, 2013.

[31] S. SAYADI, S. B. REJEB, and Z. CHOUKAIR, "Anomaly detection model over blockchain electronic transactions," in *Proceedings of the 15th International Wireless Communications Mobile Computing Conference (IWCMC)*, June 2019, pp. 895–900.

[32] J. Kim, M. Nakashima, W. Fan, S. Wuthier, X. Zhou, I. Kim, and S.-Y. Chang, "Anomaly detection based on traffic monitoring for secure blockchain networking," in *IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, Sydney Australia, May 2021.

[33] Y. Marcus, E. Heilman, and S. Goldberg, "Low-resource eclipse attacks on ethereum's peer-to-peer network." *IACR Cryptol. ePrint Arch.*, vol. 2018, p. 236, 2018.

[34] B. Alangot, D. Reijsbergen, S. Venugopalan, and P. Szalachowski, "Decentralized lightweight detection of eclipse attacks on bitcoin clients," 2020.

[35] M. Signorini, M. Pontecorvi, W. Kanoun, and R. D. Pietro, "Bad: Blockchain anomaly detection," 2018.

[36] Z. Qian and Z. M. Mao, "Off-path tcp sequence number inference attack - how firewall middleboxes reduce security," in *2012 IEEE Symposium on Security and Privacy*, 2012, pp. 347–361.

[37] Z. Qian, Z. M. Mao, and Y. Xie, "Collaborative tcp sequence number inference attack: How to crack sequence number under a second," in *Proceedings of the ACM Conference on Computer and Communications Security*, 2012, p. 593–604.

[38] Y. Gilad and A. Herzberg, "Off-path tcp injection attacks," *ACM Transactions on Information and System Security (TISSEC)*, vol. 16, no. 4, pp. 1–32, 2014.