Bijack: Breaking Bitcoin Network with TCP Vulnerabilities

Shaoyu Li[1], Shanghao Shi[1], Yang Xiao[2], Chaoyu Zhang[1],
Y. Thomas Hou[1], and Wenjing Lou[1]

[1] Virginia Polytechnic Institute and State University, VA, USA
{shaoyuli, shanghaos, chaoyu, thou, wjlou}@vt.edu
[2] University of Kentucky, KY, USA
xiaoy@uky.edu

Abstract. Recent studies have shown that compromising Bitcoin's peer-to-peer network is an effective way to disrupt the Bitcoin service. While many attack vectors have been uncovered such as BGP hijacking in the network layer and eclipse attack in the application layer, one significant attack vector that resides in the transport layer is largely overlooked. In this paper, we investigate the TCP vulnerabilities of the Bitcoin system and their consequences. We present Bijack, an off-path TCP hijacking attack on the Bitcoin network that is able to terminate Bitcoin connections or inject malicious data into the connections with only a few prior requirements and a limited amount of knowledge. This results in the Bitcoin network topology leakage, and the Bitcoin nodes isolation. We measured the real Bitcoin network and discovered that more than 1700 (27%) of the reachable Bitcoin nodes are vulnerable to our attack whose physical locations are spread across the world. We evaluated the efficiency and impacts of the Bijack attack in real-world settings, and the results show that Bijack successfully realizes several fatal Bitcoin attacks without too much effort.

Keywords: Bitcoin · TCP · Network security.

1 Introduction

With a market capitalization of more than 534 billion US dollars (May 9th, 2023), Bitcoin is among the most successful cryptocurrencies. The fundamental appeal of Bitcoin stems from its underlying design, the blockchain system, which is characterized as a fully decentralized architecture [33] that relies on a unique consensus protocol to ensure its security and immutability. Within this large and decentralized system, tens of thousands of Bitcoin nodes have formed a global peer-to-peer network overlaying upon the Internet. This peer-to-peer network, commonly referred to as the Bitcoin network, enables Bitcoin nodes to transmit transactions and blocks to each other and is critical to the fundamental consensus security of Bitcoin [46].

As a global and public infrastructure, the Bitcoin network has attracted various attacks from different perspectives that aim to disrupt the security and performance of the Bitcoin system. For example, the eclipse attack aims to dominate a victim node's communication with the main network in order to isolate it from the consensus [29, 43]. The topology inference attack seeks to extract the connection profiles of targeted nodes to manipulate their consensus status [5, 32, 36]. Other network-based Bitcoin attacks include delay attacks [7, 22] and deanonymization attacks [2, 5], for which Section 8 provides a detailed discussion. In order to realize these network-based attacks, the attacker needs to manipulate the P2P connections of the victim, which ultimately requires tampering with the Internet functions that underpin the P2P network. To this regard, the BGP hijacking attack [3] and its stealthier variant [43] exploit the vulnerabilities of the BGP protocol to allow an autonomous system (AS)-level attacker to redirect all traffic from/to a victim toward its malicious routers. More recent connection manipulation attacks [15, 16] leverage the positional advantage of the routing-level attackers to eavesdrop, monitor, and tamper with specific Bitcoin traffic.

Limitation of On-path Attacks The aforementioned connection manipulation attacks are predominantly performed by an *on-path* attacker. This assumption is impractical and often does not yield an attack reward comparable to the potential cost. On-path attackers, who can intercept, monitor, and modify network traffic trespassing them, are classified into two categories: routing-level attackers, such as switches or routers, and AS-level attackers. However, in the case of specific connection attacks, routing-level attackers are unlikely to cause a significant impact on the overall network because they can only disrupt the traffic passing through them, which affects only a small fraction of Bitcoin nodes. As for AS-level attackers, although they have the ability to monitor and tamper with a large volume of network traffic, they often refrain from doing so due to the need to carefully weigh the costs and potential reputation impact of their malicious actions against the potential gains of the attack. These large actors may face serious commercial and regulatory consequences when they are detected. Moreover, the open and dynamic nature of the Bitcoin network, whose topology is subject to constant change, imposes an additional cost for the on-path attacker to adapt and re-launch the attack.

Another commonality among existing network-based attacks is the overlook of Transport Layer vulnerabilities of the Bitcoin network. Like most connection-oriented network applications, Bitcoin relies on the TCP protocol for end-to-end data transmission between nodes, utilizing TCP connections established through the TCP three-way handshake. However, TCP itself has no authentication mechanism to build up secure channels between Bitcoin nodes and cannot verify the integrity of transmitted Bitcoin data. This creates an opportunity for attackers to manipulate Bitcoin connections by compromising the TCP connections and substituting legitimate data with malicious data. Worse yet, the Bitcoin protocol stack naturally transmits all traffic in plaintext, and Bitcoin does not employ TLS (Transport Layer Security, [13]) to guarantee the security of the TCP connections as in normal web apps like email and VoIP (Voice over Internet Protocol, [27]). Therefore, anybody in the network is able to eavesdrop, capture, and analyze the

TCP traffic of the victim nodes, opening up opportunities for *off-path* attackers to conduct TCP-based manipulation attacks on the Bitcoin network.

Our Work In this paper, we propose Bijack, a new off-path Bitcoin TCP hijacking attack against the Bitcoin network. As an off-path attack, Bijack does not require the attacker to have knowledge of on-path communication traffic between Bitcoin peers, nor need any information about the internal operating information of Bitcoin nodes. We exploit a TCP protocol vulnerability of the Linux system [17, 18, 37] to devise our attack, which is based on a security flaw of the mixed IPID assignment method in some versions of the Linux kernel. Our attack can be conducted in three phases. First, the attacker discovers the victim node by a flaw detection mechanism to identify whether the node is subject to the TCP vulnerability we have mentioned. Second, the attacker identifies the Bitcoin connections between the victim node and its peers. The Bitcoin connections will be tricked into downgrading the IPID assignment method from the per-packet-based method to the globally 2048 hash-based method and a side channel method based on the globally hash-based IPID assignments is utilized to infer the three-tuple `[victim node's port number, peer's IP address, peer's port number]`. The attacker completely hijacks the connections by inferring the sequence and acknowledgment numbers of the victim connections. After a successful hijack, the attacker can terminate the TCP connections by sending a forged TCP RST segment or injecting malicious Bitcoin data into the connections to disrupt the Bitcoin system. As a result, the attacker can take over the Bitcoin connections and send malicious transactions or blocks to the victim nodes to break the Bitcoin consensus.

To show the potential impact of Bijack, we demonstrate two Bitcoin network attacks mentioned earlier—the topology inference attack and the eclipse attack—for which an off-path attacker can perform based on Bijack. For the topology inference attack, the attack goal is to know the Bitcoin network topology around the victim nodes. The victim nodes are tricked by the attacker to send the known addresses to the attacker, helping it to detect the potential connections. Bijack allows the off-path attacker to build connections with the victim nodes and send forged network packets, and then infer the other connections of victim nodes. For the eclipse attack, the attacker aims to isolate the victim node from the rest of the Bitcoin network by surrounding it with malicious nodes, effectively controlling all incoming and outgoing connections of the victim. With Bijack, the off-path attacker who controls a swarm of malicious nodes (similar to the Sybil attack) can continuously disrupt the benign connections established by the victim until all of the victim's connections are established with the malicious nodes.

Evaluation We provided global network-wide measurement and surprisingly found out that more than 27% of the total Bitcoin nodes are vulnerable to our attack as of May 2023. We also implemented the Bijack attack in the real world and evaluated the efficiency and impacts of the Bijack attack. For the topology inference attack, when given the address list of 46262 potential peers, the attacker was able to infer all connected peers of the victim node in 25.68 hours. When performing the eclipse attack, the attacker discovered all initial ten outbound connections of the victim node in 168 minutes and successfully isolated the victim node in 11.6 hours. Finally, we propose practical countermeasures (Section 7.2) from the perspective of the network and Bitcoin system to detect and defend against the Bijack attack.

In summary, we make the following contributions:

- To the best of our knowledge, this is the first work that focuses on the TCP vulnerabilities of the Bitcoin network. We identify this unique attack vector and its security impacts imposed on the Bitcoin network.
- We propose Bijack, an off-path Bitcoin TCP attack that only requires very little prior knowledge of the victim nodes. The attack can be launched by any malicious party within the Bitcoin network, resulting in a complete hijacking of the communication session between victim nodes. Bijack can lead to further catastrophes results including topology leakage, eclipse, and even double-spending.
- We measured all the reachable nodes from the Bitcoin network and found that more than 27% of them are vulnerable to our attack, calling for an urgent need to fix this vulnerability. We implemented Bijack attack in real Bitcoin networks by performing the topology inference and eclipse attack, and the experiment results confirm the efficiency and effectiveness of our attack.

2 Background

2.1 Bitcoin Network Formation

As a peer-to-peer network, Bitcoin requires each node to maintain a list of IP addresses of potential peers. This list stored in the local addresses database is initially acquired from a public DNS server, and additional addresses are exchanged among connected peers. Each Bitcoin node pseudo-randomly selects peers from the list to build unencrypted TCP connections with them. By default, each Bitcoin node establishes 10 outbound connections (including 2 block-relay connections) and accepts up to 117 inbound connections on TCP port 8333.

Nodes request connected peers' known addresses by sending GETADDR messages and the peer responds with ADDR messages containing up to (but necessarily) 1000 known node addresses. In addition, most nodes will unsolicitedly propagate their own addresses in ADDR messages to their peers when building new connections. Currently, in order to avoid topology leakage, each node can only propagate at most 1000 addresses per day [34].

2.2 TCP Vulnerability

The TCP vulnerability revealed in 2020 [37] enables an off-path attacker to monitor TCP connections of the victim hosts when they run the Linux kernels prior to version 5.17 [17, 18]. In this attack, the attacker first pretends to be a router and sends a forged ICMP "Fragmentation Needed" error message [38] to the victim node in order to trigger it to downgrade the IPID assignment of the victim connection from the per-socket-based method to the insecure hash-based method. For the hash-based method, the node uses a total of 2048 (11 bits) IPID counters determined by $IPID_{counter} = HASH(sourceIP, destIP, protocol, Boot\_key)$ to assign IPIDs for its IP packets. However, this method has been shown to be insecure [17] as the hash collision space is too small and an attacker is able to use many IP addresses and its desired protocol to trigger a hash collision. For example, the attacker can achieve this by using ICMP protocol and trying different destination IP addresses, as shown in Eq. (1):

$$\begin{aligned} &hash(victim\_node\_IP, peer\_IP, TCP, Boot\_key) = \\ &hash(victim\_node\_IP, attacker\_IP, ICMP, Boot\_key) \end{aligned} \tag{1}$$

In practice, the attacker can send ICMP echo request messages with its IP addresses and observe the IPID of the returned ICMP echo reply messages. If one IP address collides with the targeted TCP connection, the attacker can observe a non-linear IPID increment in its received ICMP messages because the victim connection and the attacker's ICMP connection are using the same IPID counter. As a result of this hash collision, the attacker is able to monitor the IPID changes in the victim's TCP connection by monitoring its own ICMP connection. More details about IPID can be found in Appendix 10.1.

3 Bijack: Hijacking Bitcoin TCP Connections

3.1 Attack Model

The goal of our Bijack attack is to hijack the Bitcoin connections of the victim node. Fig. 1 shows the attack model of Bijack, in which three types of nodes are involved, including the victim node V, the list of peers connected to the victim $P = \{p_1, p_2, \cdots, p_n\}$, and an off-path attacker A.

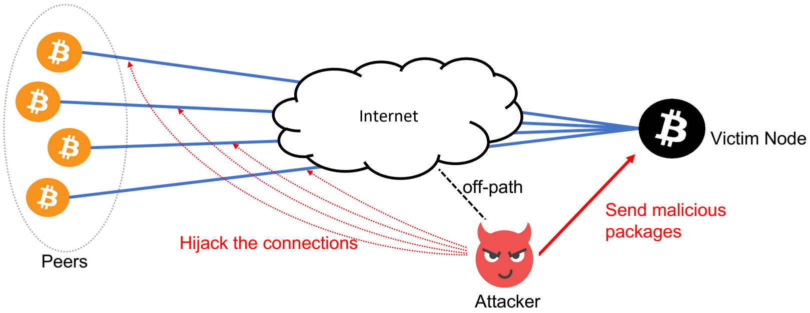


Fig. 1. The off-path attack model

We assume the off-path attacker is unable and does not necessarily need to monitor any inbound or outbound network traffic of the victim node. The attacker also has no information about any internal operating parameters and configurations of the victim node except the victim node's IP address, which is used as the public identifier of the victim node. We assume the attacker is able to craft and send malicious IP packets to the network, as well as possessing many IP addresses, following the convention of the existing Bitcoin network attacks [19]. We assume attacker A has the ability to send forged TCP segments, ICMP messages, and Bitcoin messages to victim V, without needing to manipulate the ASes (Autonomous Systems) to relay the forged packets, as over a quarter of ASes do not discard packets with spoofed source addresses in their networks [31]. In practice, any node in the Bitcoin network, such as a Bitcoin mining node or a light node, can become an attacker.

3.2 Detailed Procedures of Bijack

Phase-1: Victim Detection Discovering vulnerable Bitcoin nodes that deploy a vulnerable Linux kernel is necessary for an off-path attacker to perform both node-level and network-level attacks because the attacker aims to detect the Bitcoin connections of the vulnerable nodes. Fig. 2 illustrates the workflow of detecting the vulnerable nodes from the Bitcoin network.

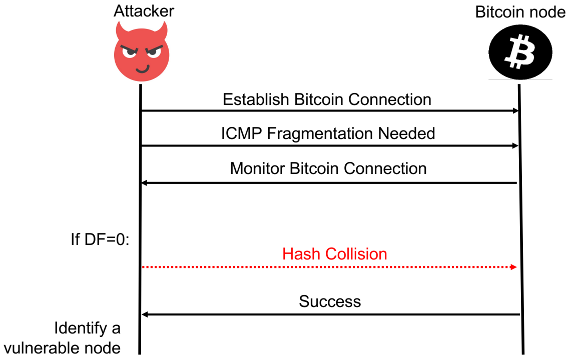

Attacker
Bitcoin node
Establish Bitcoin Connection
ICMP Fragmentation Needed
Monitor Bitcoin Connection
If DF=0:
Hash Collision
Success
Identify a vulnerable node

Fig. 2. Discovering a victim node

The attacker first establishes a Bitcoin connection with the target node to test if it is vulnerable. The attacker attempts to downgrade the IPID assignment method of the Bitcoin connection by sending an ICMP "Fragmentation Needed" message to the tested node. Only the **vulnerable** Bitcoin node will reply to the attacker with Bitcoin messages whose DF field changed from one to zero. After monitoring this, the attacker conducts the hash-collision as we have described in Section 2.2, and if it succeeds, it confirms that the current node is a vulnerable one.

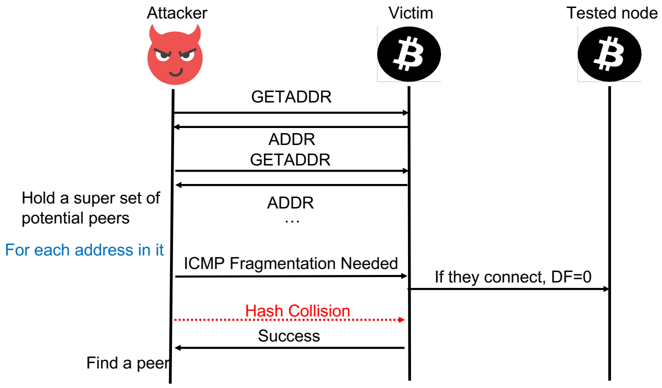


Fig. 3. Finding the victim's peer IP address

Phase-2: Connection Detection For each victim node, the attacker attempts to reveal the details of the victim's existing Bitcoin connections established with peers. Each Bitcoin connection can be treated as a four-tuple vector, i.e., [**victim node's IP address, victim node's port number, peer's IP address, peer's port number**]. The attacker only knows the victim node's IP and it will infer the other three components.

Step 1: Finding Victim's Peer IP Addresses The workflow of this IP detection process is shown in Fig. 3. To begin with, the attacker sends GETADDR messages to the victim node and collects the addresses in the replied ADDR messages, which may contain the connected peers as described in Section 2.1. Then each IP address in the ADDR messages will be tested to see if it connects to the victim nodes. The attacker sends a forged ICMP "Fragmentation Needed" message with the tested IP address to the victim node. If the victim node does have a connection with the tested IP, the connection will be triggered to downgrade the IPID assignment to the hash-based method, which will be detected by the attacker through hash-collision mentioned in Section 2.2.

Step 2: Inferring Port Numbers of the Victim and Peers In this step, the attacker infers the port numbers between the victim and its peers. The attacker will first assume one node uses the destination port (typically 8333, while it can be detected by network scanning) and infer the other one's port number. If unsuccessful, swap the assumption. As a bonus, after the port inferring process, the attacker obtains knowledge about whether the current Bitcoin connection is inbound or outbound.

The workflow of port inference is illustrated in Fig. 4. For each of the identified peers, the attacker starts with continuous monitoring of the IPID increment between the victim node and the peer. The attacker can do so by continuously sending hash-collided ICMP messages (already succeed in the previous phase) to the victim and observing the returned messages. To infer the port number, the attacker sends forged TCP SYN/ACK segments to victim nodes with different

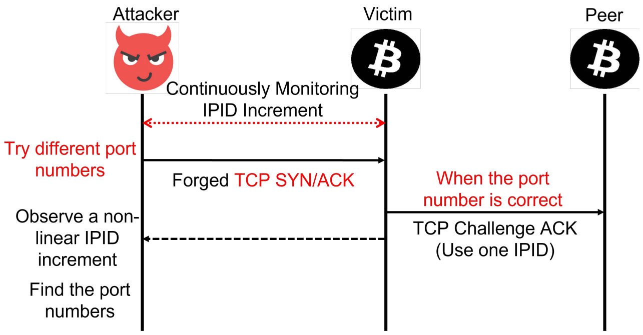

Attacker
Victim
Peer
Continuously Monitoring IPID Increment
Try different port numbers
Forged TCP SYN/ACK
When the port number is correct
TCP Challenge ACK (Use one IPID)
Observe a non-linear IPID increment
Find the port numbers

Fig. 4. Inferring the port numbers between the victim and its peer

port numbers across the range (from 1024 to 65535). When the port number is correct, the victim will send a TCP Challenge ACK segment [41] to the peer, and if not, the victim responds with a TCP RST segment to the peer with a 0 value of the IPID [1, 30]. Because this TCP Challenge ACK segment uses one or more additional IPIDs shared between the victim and attack connections, the attacker can observe a non-linear IPID increment, which is the indicator of the success of our inferring process. Note that this inferring process can be finished in a short time, we do assume there is no other TCP connection between the victim and its peer.

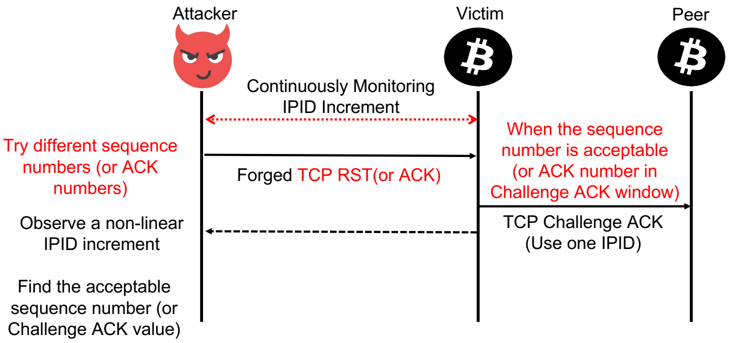

Attacker
Victim
Peer
Continuously Monitoring IPID Increment
Try different sequence numbers (or ACK numbers)
Forged TCP RST(or ACK)
When the sequence number is acceptable (or ACK number in Challenge ACK window)
TCP Challenge ACK (Use one IPID)
Observe a non-linear IPID increment
Find the acceptable sequence number (or Challenge ACK value)

Fig. 5. Inferring acceptable sequence and ACK numbers

Step 3: Inferring Sequence Number and Acknowledgment Number The attacker infers the exact sequence number and acceptable acknowledgment number in order to gain full knowledge of the victim's connection. The attacker achieves this in serial steps including first inferring an **acceptable sequence number**, then an **acknowledgment number located in the Chal-**

lenge ACK window, and finally the **exact sequence number** as well as the **acceptable ACK number**.

The workflow and related terms of inferring an acceptable sequence number and an ACK number located in the Challenge ACK window are illustrated in Fig. 5. To infer an acceptable sequence number, the attacker sends the forged TCP RST segments with their guessed sequence numbers to the victim node, which will respond with a Challenge ACK segment to its peer if and only if the guessed sequence numbers fall in an acceptable window. Similar to the previous step, this Challenge ACK segment triggers a non-linear increment on the shared IPID counter between the attacker and the victim node, detectable by the attacker as the signal of finding an acceptable sequence number. Then with this acceptable sequence number, the attacker can infer an ACK number located in the Challenge ACK window (ranging from 1G to 2G [6, 9, 10]) by sending forged ACK segments and monitoring the IPID increment in the same fashion [41]. After that, the attacker infers the exact sequence number with a well-known method [17] as to sending forged ACK segments to the victim with decreasing sequence numbers from the acceptable sequence number and monitoring the reply rate of the TCP segments (or the IPID non-linear incremental rate). In the beginning, there is a burst of challenge ACK segments sent by the victim at the limited speed of 500 ms per segment by the protocol design of TCP. Once the sequence number reaches the lower bound, the sequence number is the exact one and the victim nodes will send ACK segments to its peer without any speed limitation. When inferring the acceptable ACK number, the lower bound of the challenge ACK window can be inferred in the same way and then the attacker uses it to calculate the *sequence number of the first unacknowledged octet* (the lower bound value adding 2G), which can be used with the known *typical size of the send window* to finally calculate the acceptable ACK number.

Phase-3: Hijack and Manipulation With the correct inference of the underlying TCP layer information of the victim's Bitcoin connections, the off-path attacker is able to send spoofed traffic to the victim nodes to influence the victim's normal Bitcoin activities. The connections could be forcefully terminated by the attacker, using the knowledge of either the TCP or Bitcoin protocol. Moreover, the attacker can inject malicious Bitcoin data including fake transactions and blocks into the connections, which will disrupt the victim node from understanding the blockchain ledger, further influencing the integrity and stability of the Bitcoin consensus. We will explore these vulnerabilities in the next two sections.

4 Compromising Bitcoin Network Nodes

Hijacking Bitcoin connections can pose significant security risks to both Bitcoin nodes and the Bitcoin network. In this section, we have demonstrated two Bitcoin node manipulation attacks based on Bijack: (i) Bitcoin topology inference and (ii) eclipse attack. We will demonstrate how they are launched and their consequences on the Bitcoin network.

4.1 Bitcoin Topology Inference

Compared to the previous Bitcoin topology inference attack [5, 12, 28, 32, 36], Bijack can directly infer the inbound and outbound connections of the victim node through message feedback directly obtained from the network traffic without the requirement of collecting and analyzing detailed Bitcoin transactions or blocks. The attacker can detect all or at least most of the inbound and outbound connections of a targeted node.

In practice, to infer more connections, the attacker can repeatedly request ADDR messages as long as it does not exceed the limitation set by the Bitcoin system to gain as many potential peer IP addresses as possible. This allows the attacker to build up a superset of all the IP addresses it receives, up to 1,000 addresses per day. According to Bitcoin's design rules, the victim will randomly select peers to establish connections from their known IP addresses, which are highly likely to be within the IP superset, provided that the superset is large enough.

The attacker acquires the topology information of the victim nodes by launching this attack, which can be further exploited to conduct more severe Bitcoin attacks. For example, the attacker can identify the most connective nodes as the key or super nodes, and place corrupted nodes in these key locations or attack the super nodes to disrupt data transmission in the network. The attacker may even infer the complete topology of a local (e.g. in a certain network domain) or global Bitcoin network through mathematical modeling and analyzing the inbound and outbound relationships of the nodes [32], leaving space for the attacker to conduct the eclipse attack that isolates the victims. Moreover, the attacker is able to perform the 0-confirming double-spending attacks on the victim. After inferring the connections of the victim merchant, the attacker sends the double-spend transaction only to the victim's peers and sends others the legal transaction. The merchant will confirm the double-spending transaction after receiving it from most of its peers, while the legal transaction will be selected in the blockchain.

4.2 Eclipse Attack

The eclipse attack is a severe Bitcoin attack that aims to isolate the victim nodes from the rest of the network. It can render the victim nodes vulnerable to a double spending attack because the attacker controls the propagation of transactions to the victim nodes. It can also waste the mining power by manipulating the victim's view of the blockchain. Moreover, if the attacker is able to isolate a large number of Bitcoin nodes, the whole Bitcoin network may be partitioned. Unfortunately, Bijack can help the attacker to accomplish this in the following way.

The attacker first continuously sends Bitcoin ADDR messages to the victim node with multiple malicious IP addresses controlled by it. Because the current Bitcoin protocol lets the node accept all the received IP addresses without any verification, the attacker can gradually pollute the local IP database of the victims by increasing the portion of malicious IP addresses, from where the victim nodes establish outbound connections. In practice, to increase the number of nodes stored in the victim's database, the attacker can inject IP addresses with different prefixes to circumvent the built-in address discarding mechanism in the database—the database allocates a limited quota for IP addresses with the same prefix, and any exceeding ones will be discarded [29, 43].

After this, the attacker attempts to manipulate all the victim node's connections through the Bijack. Once the attacker finds that the victim node is shut down and restarted, it immediately occupies all the inbound connections with its controlled IP addresses. This is achievable because the Bitcoin system does not specify its nodes to verify or authenticate the inbound connection requests. Moreover, existing work has shown that the Bitcoin nodes may restart for several reasons such as software updating, power failure, and DDoS attacks [11, 40, 42, 44]. For the outbound connections, even if the attacker has polluted the local addresses database of the victim node, the benign IP addresses still constitute a large fraction and the victim may still establish connections with them. To terminate these benign connections and allow the attacker to fully control the victim's connections, the attacker needs to first detect and hijack all the benign connections with Bijack. The attacker then impersonates the corresponding peers of these connections to disrupt them by either sending forged TCP RST segments to trigger connection termination or sending malicious Bitcoin messages to the victim nodes, which causes ban scores of the benign peers to increase until they reach 100, resulting in a one-day blacklisting [15, 16].

As a result, the attacker disconnects all the benign nodes from the victims and fully controls all their inbound and outbound connections, accomplishing the eclipse attack.

5 How Vulnerable is Bitcoin to Bijack?

Evaluating the impact of our Bijack attack requires a good knowledge of the vulnerable nodes in the Bitcoin network. In this section, we conduct a measurement of the Bitcoin network to explore the number of vulnerable nodes as well as their mining power in the network and analyze Bijack potential impact.

5.1 Measurement on Real Bitcoin Network

We utilized one scanner Bitcoin node running Bitcoin Core version v24.99.0, with the IP address of *38.68.237.175*. Our scanner node was installed with Ubuntu 18.04 (Linux kernel version 4.15) and was capable of sending ICMP messages to other nodes using spoofed IP addresses with the Python Scapy package.

The victim detection phase (following Section 3.1) was carried out on the entire Bitcoin network for 10 days, from April 27th, 2023, to May 7th, 2023 (the detailed procedure is shown in Appendix 10.2). During this time period, we discovered and successfully established Bitcoin connections with 6405 Bitcoin nodes and we found that 27.14% of connected nodes (1738 nodes) are vulnerable to the Bijack. We show our experiment results in Table 1, in which we present the geo-location, the number of vulnerable and reachable nodes, as well as the total scanning time.

Table 1. The top ten countries with the highest number of vulnerable nodes

Location	Victim Clients	Total Clients	Scan Time(min)
USA	332	1200	711.5
Germany	300	726	396.9
Netherlands	119	264	142.3
France	102	266	148.4
Finland	95	210	113.1
Canada	65	195	111.6
Singapore	53	114	61.0
United Kingdom	46	137	80.6
Japan	43	83	43.8
Switzerland	43	135	78.5

We measured the vulnerable mining nodes from the Bitcoin network during the same time period. We collected all the nodes that first relay new blocks, considering them as gateways of mining pools. We also collected the IP addresses of the mining devices by scanning the IPv4 network. We found that over 90% of the vulnerable nodes are associated with mining activities, with approximately 40% being mining nodes and the remaining portion belonging to mining pool gateways.

5.2 Bitcoin Impact Analysis

Our measurement found more than 27% Bitcoin nodes are vulnerable to our attack, spreading across different geographical locations. Therefore, these nodes are directly exposed to the threats we have mentioned in the previous section such as topology leakage, eclipse, and even double-spending. From the network's perspective, the attacker can cause more severe consequences as it can **partition** the whole Bitcoin network considering that 27% is a considerably large fraction and over 90% of the detected victim nodes belong to the mining nodes or mining pool nodes. These nodes possess a significant amount of computational power within the Bitcoin system. After partitioning the network, the attacker gains control over these nodes, and its computation power increases significantly, giving him a huge advantage to perform the selfish-mining attacks [14, 21, 35], in which the attacker may strategically conceal and release newly mined blocks to realize unfair mining gain when the attacked-controlled mining power exceeds a certain threshold β. Assuming the attacker's released block wins the fork competition of 50% chance, the threshold β becomes 25%. There is a non-negligible chance that the mining power of the 27% victim population may well exceed 25% of the total.

6 Experiment and Evaluation

We conducted the **topology inference attack** and **eclipse attack** on the real Bitcoin network to evaluate the effectiveness of Bijack. By launching the topology inference attack, we can infer **all** of the connected peers of the victim nodes (from the list of 46262 potential peers) in **25.68 hours**. By launching the eclipse attack, we isolate the victim node in **11.6 hours**.

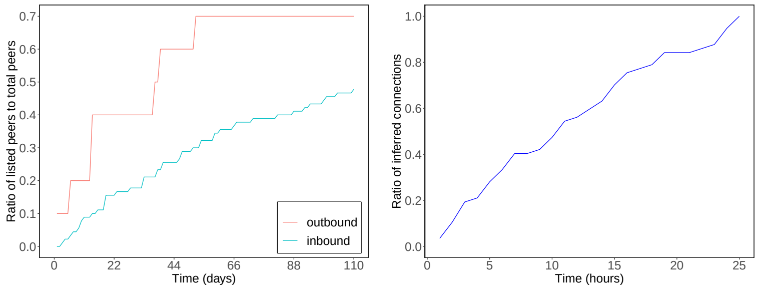
Ethical Considerations In order to prevent any potential harm or negative repercussions on the Bitcoin network and market, we only conduct the vulnerable detection phase of our attack without the following steps, which will jeopardize the operation of Bitcoin. For these steps that may cause actual harmful consequences, we implemented them only on our own machines. Our experimental activities do not pose any threat to other Bitcoin nodes. We did not send a large number of IP packets in the public Bitcoin network in order to not increase the burden on the network, and we maintain confidentiality regarding the list of nodes that are susceptible to the vulnerability.

6.1 Experiment Setup

We deployed one victim node with Bitcoin Core version v24.99.0 on the Amazon cloud by using an AWS EC2 virtual machine with Ubuntu 20.04 (Linux kernel version 5.5) located in the US East. Before our experiment, we ran the victim Bitcoin client on the node for 65 days to get it to fit into the environment of the Bitcoin system. We deployed twenty attacker nodes with Bitcoin Core version v24.99.0 equipped with Ubuntu 20.04 (kernel version 5.5). The prefix of the IP addresses for these nodes is 38.68.237.0/24. We own over 5000 addresses with the prefix of 71.178.0.0/16, 96.231.0.0/16, and 38.68.160.0/20 for hash collision and eclipse attack.

6.2 Experimental Results

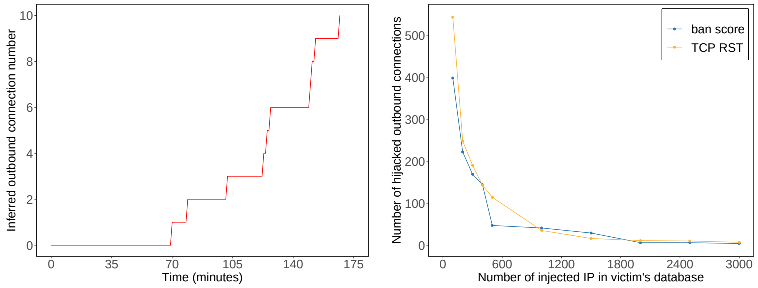
Bitcoin Topology Inference Attack We first conducted a 110-day experiment to evaluate the effectiveness of our peer detection process, i.e., the number of victim's connections that can be inferred from the address list collected from the ADDR messages. In our experiment, we continuously sent GETADDR messages to the victim node each day and collected the addresses returned by the victim nodes. The experimental results are shown in Fig. 6(a). Our experiment shows that after collecting addresses for a continuous period of 110 days, the attacker obtains the list that contains over 70% of the victim's outbound connected peers and over 50% of inbound connected peers. In total, the list contains 46242 addresses and 57 of them are connected peers. Subsequently, we executed our attack based on the list collected before and assessed the efficiency of the Bijack-based topology inference attack. The experimental results are shown in Fig. 6(b). In total, it took us 25.68 hours to find all 57 connections from the 46242 addresses. More specifically, the average time cost to examine one address in the list was 39.98 seconds and the average time cost to discover one connected peer was 23.94 seconds.



(a) Ratio of connected peers in the list to all connected peers

(b) Ratio of inferred connections in topology inference attack

Fig. 6. Topology Inference Attack Results



(a) Number of inferred outbound connections in eclipse attack

(b) The relationship between injected address and the hijacked connections

Fig. 7. Eclipse Attack Results

Eclipse Attack We first scanned the whole Bitcoin network and found 5222 active nodes on May 10th, 2023. Then we checked each node to detect if the victim node built a connection with it. We used 5000 addresses (controlled by us) to conduct the hash-collision with each node and if we fail, we consider that node does not have a connection with the victim node. The average time cost of checking an unconnected node is 39.98 seconds by attempting all the 5000 addresses. For connected nodes, we utilized an average of 3461 addresses to discover their connection with a time cost of 27.40 seconds. In total, we spent 168 minutes finding all ten outbound connections of the victim, and Fig. 7(a) illustrates the results of discovering all outbound connections.

Afterward, we kept sending ADDR messages to the victim nodes to inject the malicious IP addresses into the victim's database. Each time we sent 1000 IP addresses to the victim node in 6 TCP segments with a total payload of 17495 bytes. Finally, we sent TCP RST segments or fake Bitcoin blocks (ban-score-based method) to reset the Bitcoin connections. We disrupted each outbound connection until all of the connections were established to our attacker nodes.

Fig. 7(b) illustrates the relationship between the number of injected malicious IP addresses and the number of required hijacked connections to complete the attack. We found that the number of required hijacked connections decreases when the number of polluted IP addresses increases and in general the ban-score-based method requires fewer hijacked connections than the TCP RST-based method. In our experiment, the average timing overhead for resetting a Bitcoin connection was 163 seconds for the TCP RST-based method and 247 seconds for the Bitcoin ban-score-based method. Specifically, when 200 IP addresses were injected into the database, we used 11.6 hours to break the required 248 connections to accomplish our attack by sending TCP RST segments.

7 Discussion and Countermeasures

7.1 Discussion

The Bitcoin system transmits transactions and blocks in plaintext with the underlying TCP protocol and does not offer any encryption and authentication mechanism in order to reduce the payload of the network. Many other blockchain networks have similar properties including Litecoin [39], and Ripple [4]. Unfortunately, this makes them vulnerable to Bijack as the prerequisite requirement for successfully launching our attack is that the network traffic is not encrypted. For the blockchain networks that offer authenticated and encrypted traffic such as Ethereum [8], our attack fails to break their systems.

7.2 Countermeasures

The Bitcoin system may use the following feasible countermeasures to defend itself against Bijack.

Deploy a Customized Designed Intrusion Detection System Bijack introduces some extra abnormal traffic to the system that can be detected by an intrusion detection system (IDS). For example, the IDS can monitor the IPID increment of the Bitcoin connections, or carefully check the ICMP "Fragmentation Needed" messages.

Refuse Unsolicited ADDR Messages The node could choose to refuse the unsolicited ADDR messages with a large number of IP addresses, especially from incoming peers. This will prevent attackers from polluting the victim's address database, making it difficult to carry out an Bijack-based eclipse attack.

Encrypt the Traffic If Bitcoin traffic is transmitted using encryption, our attack's impact will be significantly reduced. It would be challenging for attackers to send spoofed messages. Considering the impact of encryption on network performance, we can allow nodes to choose whether to encrypt based on their own circumstances.

Using Tor Network Our attack cannot target the Tor network because our attack is based on the IPv4 network and we first need to identify the victim's IP address. Tor is anonymous by design and most existing Bitcoin attacks are not effective against the Tor network. Therefore, using the Tor network can mitigate network attacks.

8 Related Work

Bitcoin Network Attacks The security of the Bitcoin network has gained a lot of attention from the academic community. The well-known eclipse attack [29, 20] exploits the vulnerabilities of Bitcoin's built-in peer-selecting procedure by injecting the address database of victims with the attacker-controlled to isolate the victim Bitcoin nodes from the major Bitcoin network. BGP hijacking attack [3] and EREBUS attack [43] exploits the advantage of an AS-level attacker to delay messages received by nodes or partition nodes. The Topology Inference attacks [5, 12, 28, 32, 36] infer connections by analyzing the transmitted Bitcoin data or timestamps. On-path Bitcoin network attacks [15, 16] hijack the Bitcoin connections to disrupt the operation of the system. The delay attack [7, 22, 45] exploits network timing as the attack vector and impedes the reception time of certain Bitcoin messages of the victim nodes. The data received and stored by the victim node differs from that of the remaining nodes in the network within a certain period, resulting in wasted computing power and defaming the victim node to be susceptible to double-spending attackers. Lastly, the deanonymization attack [2, 5] reveals the real IP addresses of the victim nodes by analyzing the Bitcoin traffic, making every transaction associated with the victim's IP address public.

Off-path TCP Vulnerabilities Side-channel attack in the challenge ACK mechanism [9, 10] can infer the TCP utilization for one specific connection and then hijack it by inferring its sequence numbers and ACK numbers. Global IPID counter vulnerability is exploited to infer TCP connections and help attackers inject malicious data into the TCP connections to poison the HTTP and Tor traffic [23–26]. Mixed IPID assignment off-path attack [17, 18] leverages a new side channel vulnerability to downgrade the TCP connections of IPID assignment to the 2048-hash-based method, which helps the attacker infer the source port number and the destination port number of the connection, inferring the sequence numbers and the acknowledge numbers to hijack the TCP connection.

9 Conclusion

In this paper, we propose Bijack, a new off-path Bitcoin TCP hijacking attack against the Bitcoin network by exploiting a TCP protocol vulnerability of the Linux system. We also demonstrate two Bitcoin network attacks—the topology inference attack and the eclipse attack—to show the impact of our attack on the Bitcoin network. We measure the number of vulnerable nodes in the real Bitcoin network and analyze the influence of our attack. We evaluate the efficiency of our attacks. Our experiments show that the off-path attackers can successfully carry out the topology inferring attack and eclipse attack effectively.

Acknowledgement

This work was supported in part by the US National Science Foundation under grants 2247560, 2154929, 1916902, and 2247561.

References

1. Alexander, G., Espinoza, A. M., and Crandall, J. R. Detecting tcp/ip connections via ipid hash collisions. *Proceedings on Privacy Enhancing Technologies 2019*, 4 (2019).
2. Apostolaki, M., Maire, C., and Vanbever, L. Perimeter: A network-layer attack on the anonymity of cryptocurrencies. In *Financial Cryptography and Data Security: 25th International Conference, FC 2021, Virtual Event, March 1–5, 2021, Revised Selected Papers, Part I 25* (2021), Springer, pp. 147–166.
3. Apostolaki, M., Zohar, A., and Vanbever, L. Hijacking bitcoin: Routing attacks on cryptocurrencies. In *2017 IEEE symposium on security and privacy (SP)* (2017), IEEE, pp. 375–392.
4. Armknecht, F., Karame, G. O., Mandal, A., Youssef, F., and Zenner, E. Ripple: Overview and outlook. In *Trust and Trustworthy Computing: 8th International Conference, TRUST 2015, Heraklion, Greece, August 24-26, 2015, Proceedings 8* (2015), Springer, pp. 163–180.
5. Biryukov, A., Khovratovich, D., and Pustogarov, I. Deanonymisation of clients in bitcoin p2p network. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security* (2014), pp. 15–29.
6. Borman, D., Braden, B., and Jacobson, V. Rfc 7323: Tcp extensions for high performance, 2014.
7. Boverman, A. Timejacking & bitcoin. *Culubas blog* (2011).
8. Buterin, V., et al. A next-generation smart contract and decentralized application platform. *white paper 3*, 37 (2014), 2–1.
9. Cao, Y., Qian, Z., Wang, Z., Dao, T., Krishnamurthy, S. V., and Marvel, L. M. Off-path tcp exploits: Global rate limit considered dangerous. In *USENIX Security Symposium* (2016), pp. 209–225.
10. Cao, Y., Qian, Z., Wang, Z., Dao, T., Krishnamurthy, S. V., and Marvel, L. M. Off-path tcp exploits of the challenge ack global rate limit. *IEEE/ACM Transactions on Networking 26*, 2 (2018), 765–778.
11. Core, B. Cve-2018-17144. `https://bitcoincore.org/en/2018/09/20/notice/` Accessed May 2023.
12. Delgado-Segura, S., Bakshi, S., Pérez-Solà, C., Litton, J., Pachulski, A., Miller, A., and Bhattacharjee, B. Txprobe: Discovering bitcoin's network topology using orphan transactions. In *Financial Cryptography and Data Security: 23rd International Conference, FC 2019, Frigate Bay, St. Kitts and Nevis, February 18–22, 2019, Revised Selected Papers 23* (2019), Springer, pp. 550–566.
13. Dierks, T., and Allen, C. Rfc2246: The tls protocol version 1.0, 1999.
14. Eyal, I., and Sirer, E. G. Majority is not enough: Bitcoin mining is vulnerable. *Communications of the ACM 61*, 7 (2018), 95–102.
15. Fan, W., Chang, S.-Y., Zhou, X., and Xu, S. Conman: A connection manipulation-based attack against bitcoin networking. In *2021 IEEE Conference on Communications and Network Security (CNS)* (2021), IEEE, pp. 101–109.
16. Fan, W., Wuthier, S., Hong, H.-J., Zhou, X., Bai, Y., and Chang, S.-Y. The security investigation of ban score and misbehavior tracking in bitcoin network. In *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)* (2022), IEEE, pp. 191–201.
17. Feng, X., Fu, C., Li, Q., Sun, K., and Xu, K. 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.

18. Feng, X., Li, Q., Sun, K., Fu, C., and Xu, K. Off-path tcp hijacking attacks via the side channel of downgraded ipid. *IEEE/ACM Transactions on Networking 30*, 1 (2021), 409–422.

19. Franzoni, F., and Daza, V. Sok: Network-level attacks on the bitcoin p2p network. *IEEE Access 10* (2022), 94924–94962.

20. Gervais, A., Karame, G. O., Capkun, V., and Capkun, S. Is bitcoin a decentralized currency? *IEEE security & privacy 12*, 3 (2014), 54–60.

21. Gervais, A., Karame, G. O., Wüst, K., Glykantzis, V., Ritzdorf, H., and Capkun, S. On the security and performance of proof of work blockchains. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security* (2016), pp. 3–16.

22. Gervais, A., Ritzdorf, H., Karame, G. O., and Capkun, S. Tampering with the delivery of blocks and transactions in bitcoin. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security* (2015), pp. 692–705.

23. Gilad, Y., and Herzberg, A. Off-path attacking the web. In *WOOT* (2012), pp. 41–52.

24. Gilad, Y., and Herzberg, A. Spying in the dark: Tcp and tor traffic analysis. In *Privacy Enhancing Technologies: 12th International Symposium, PETS 2012, Vigo, Spain, July 11-13, 2012. Proceedings 12* (2012), Springer, pp. 100–119.

25. Gilad, Y., and Herzberg, A. Off-path tcp injection attacks. *ACM Transactions on Information and System Security (TISSEC) 16*, 4 (2014), 1–32.

26. Gilad, Y., Herzberg, A., and Shulman, H. Off-path hacking: The illusion of challenge-response authentication. *IEEE Security & Privacy 12*, 5 (2013), 68–77.

27. Goode, B. Voice over internet protocol (voip). *Proceedings of the IEEE 90*, 9 (2002), 1495–1517.

28. Grundmann, M., Neudecker, T., and Hartenstein, H. Exploiting transaction accumulation and double spends for topology inference in bitcoin. In *Financial Cryptography and Data Security: FC 2018 International Workshops, BITCOIN, VOTING, and WTSC, Nieuwpoort, Curaçao, March 2, 2018, Revised Selected Papers 22* (2019), Springer, pp. 113–126.

29. Heilman, E., Kendler, A., Zohar, A., and Goldberg, S. Eclipse attacks on bitcoin's peer-to-peer network. In *24th {USENIX} Security Symposium ({USENIX} Security 15)* (2015), pp. 129–144.

30. John, P. Transmission control protocol. *RFC 793* (1981).

31. Luckie, M., Beverly, R., Koga, R., Keys, K., Kroll, J. A., and Claffy, K. Network hygiene, incentives, and regulation: deployment of source address validation in the internet. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security* (2019), pp. 465–480.

32. Miller, A., Litton, J., Pachulski, A., Gupta, N., Levin, D., Spring, N., and Bhattacharjee, B. Discovering bitcoin's public topology and influential nodes. *et al* (2015).

33. Nakamoto, S. Bitcoin: A peer-to-peer electronic cash system. *Decentralized business review* (2008), 21260.

34. Naumenko, G. Pr 18991: Cache responses to getaddr 3420 to prevent topology leaks. `https://github.com/bitcoin/bitcoin/pull/18991` Accessed May 2020.

35. Nayak, K., Kumar, S., Miller, A., and Shi, E. Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)* (2016), IEEE, pp. 305–320.

36. Neudecker, T., Andelfinger, P., and Hartenstein, H. Timing analysis for inferring the topology of the bitcoin peer-to-peer network. In *2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)* (2016), IEEE, pp. 358–367.
37. of Standards, N. I., and Technology. Cve-2020-36516. `https://nvd.nist.gov/vuln/detail/CVE-2020-36516` Accessed May 2023.
38. Postel, J. Internet control protocol. *RFC 792* (1981).
39. project, L. Litecoin. `https://litecoin.org` Accessed May 2023.
40. Raikwar, M., and Gligoroski, D. Dos attacks on blockchain ecosystem. In *Euro-Par 2021: Parallel Processing Workshops: Euro-Par 2021 International Workshops, Lisbon, Portugal, August 30-31, 2021, Revised Selected Papers* (2022), Springer, pp. 230–242.
41. Ramaiah, A., Stewart, R., and Dalal, M. Rfc 5961: Improving tcp's robustness to blind in-window attacks, 2010.
42. Schuba, C. L., Krsul, I. V., Kuhn, M. G., Spafford, E. H., Sundaram, A., and Zamboni, D. Analysis of a denial of service attack on tcp. In *Proceedings. 1997 IEEE Symposium on Security and Privacy (Cat. No. 97CB36097)* (1997), IEEE, pp. 208–223.
43. Tran, M., Choi, I., Moon, G. J., Vu, A. V., and Kang, M. S. A stealthier partitioning attack against bitcoin peer-to-peer network. In *2020 IEEE symposium on security and privacy (SP)* (2020), IEEE, pp. 894–909.
44. Vasek, M., Thornton, M., and Moore, T. Empirical analysis of denial-of-service attacks in the bitcoin ecosystem. In *Financial Cryptography and Data Security: FC 2014 Workshops, BITCOIN and WAHC 2014, Christ Church, Barbados, March 7, 2014, Revised Selected Papers 18* (2014), Springer, pp. 57–71.
45. Walck, M., Wang, K., and Kim, H. S. Tendrilstaller: Block delay attack in bitcoin. In *2019 IEEE international conference on Blockchain (Blockchain)* (2019), IEEE, pp. 1–9.
46. Xiao, Y., Zhang, N., Lou, W., and Hou, Y. T. A survey of distributed consensus protocols for blockchain networks. *IEEE Communications Surveys & Tutorials 22*, 2 (2020), 1432–1465.
47. Yeow, A. Bitnodes. `https://bitnodes.io/nodes/#network-snapshot`, Accessed April 2023.

10 Appendix

10.1 IPID Assignment

IPID Assignment The identification field (IPID) in the Internet Protocol (IP) serves as a unique identifier for each IP packet and it occupies 16 bits in the IP packet. The IPID is assigned by the sender to aid in assembling the fragments of a datagram because IP datagrams may be fragmented into multiple fragments for transmission over the network during the transmission process. The generation of the IPID can employ different algorithms or strategies, but it must be unique within the sender's context. In certain versions, Linux employs a mixed IPID assignment method for packets [1]. There are two fundamental

IPID assignment policies: the per-socket-based IPID assignment method and the 2048-globally-hash-based IPID assignment method, the former being specific to socket-based protocols such as TCP and UDP.

Per-socket-based IPID Assignment This policy is specifically used for socket-based protocols such as TCP and UDP. A unique random value is initialized for each connection, and the counter is incremented by 1 each time it is used for transmitting a packet. This random counter makes it difficult for off-path attackers to infer the IPID value.

Hash-based IPID Assignment It involves assigning the IPID based on a hash counter. Linux has a total of 2048 hash counters, and the IPID is selected from one of these counters based on the hash value of four variables: the source IP address, destination IP address, the protocol number of the packet, and a random value generated by the Linux system. After the IPID value is copied from the selected counter, the counter is incremented by a uniform distribution value between 1 and the number of system ticks that have elapsed since the last packet transmission using the same counter.

Linux uses the Don't Fragment (DF) flag in the IP protocol to differentiate between the two methods. Normally the TCP and UDP use per-socket-based IPID assignment and the DF's value is one. For other network protocols (like ICMP), the DF is set as 0. For TCP, DF is set as 1 for TCP non-RST segments, enabling the MTU discovery (PMTUD) mechanism and signaling the use of the per-socket-based IPID assignment method, which is considered more secure. The IP examines the DF flag value set by the TCP protocol. If DF is 0, the hash-based IPID assignment method is used. If DF is 1 and the packet is not for a TCP SYN/ACK segment with both SYN and ACK flags set to 1 (assigned IPID of 0), the IP assigns the IPID using the per-socket-based method.

10.2 Bitcoin Network Measurement Procedure

We first scan all connectable nodes in the network based on the method in [47]. Then, we establish Bitcoin connections with these nodes for further testing. To reduce the bandwidth load on our node, we test only one Bitcoin node at a time and establish a connection with only that one Bitcoin node. Initially, we send malicious ICMP "Fragmentation Needed" messages to attempt to clear the DF flag. As for hash collision, we first observe the average rate m at which the tested node sends Bitcoin information to our node and the average IPID increment k between each message. Then, our scanner node sends forged ICMP messages with different source IP addresses to the tested node. For each source IP address, we will send the forged packets at a rate of $n * m$ for the time period of $1/m$. If we found that the IPID of a received Bitcoin message increased by $n * m + k$ compared to the most recent previous one, we considered the tested node collided. To minimize errors caused by network latency or the randomness of the IPID increment, when we observe the IPID increment value in the range of $n * m + k$, we repeat the test with the source IP address used for the collision to verify whether the collision really occurred.