On the Sustainability of Bitcoin Partitioning Attacks

Conference Paper

Author(s):
Ha, Jaehyun; Baek, Seungjin; Tran, Muoi; Kang, Min Suk

Publication date:
2024

Permanent link:
https://doi.org/10.3929/ethz-b-000611944

Rights / license:
In Copyright - Non-Commercial Use Permitted

Originally published in:
Lecture Notes in Computer Science 13951, https://doi.org/10.1007/978-3-031-47751-5_10

On the Sustainability of Bitcoin Partitioning Attacks

Jaehyun Ha[1], Seungjin Baek[1], Muoi Tran[2], and Min Suk Kang[1]

[1] KAIST {jaehyunh1, seungjinb, minsukk}@kaist.ac.kr
[2] ETH Zurich dutran@ethz.ch

Abstract. A series of recent studies have shown that permissionless blockchain peer-to-peer networks can be partitioned at low cost (e.g., only a few thousand bots are needed), stealthily (e.g., no control plane detection is available), or at scale (e.g., the entire bitcoin network can be divided into two). In this paper, we focus on the *sustainability* of partitioning attacks in Bitcoin, which is barely discussed in the literature. Existing studies investigate new partitioning attack strategies extensively but not how long the partition they create lasts. Our findings show that, fortunately for Bitcoin, the permissionless peer-to-peer network can be partitioned but only for a short time. In particular, two recent partitioning attacks (i.e., Erebus [12], SyncAttack [10]) do not maintain partitions for more than 10 minutes in most cases. After analyzing Bitcoin's peer eviction mechanism (which makes the two original attacks difficult to sustain), we propose optimization strategies for the two attacks and calculate the total cost of the optimized attacks for a 1-hour attack duration. Our results complement the original attack studies: (i) the optimized Erebus attack shows that it requires at least one adversary-controlled Bitcoin node close to a target and a few additional expensive attack steps for sustainable attacks, and (ii) the optimized SyncAttack can create sustainable partitions only with excessive cost.

1 Introduction

Blockchain systems are desired to maintain highly reliable network connectivity across distributed nodes even in the face of severe network attacks or failures. However, Bitcoin and many permissionless blockchains are at risk of connection starvation attacks in which all available connections of public nodes are occupied by an adversary, leaving no resource for legitimate nodes. Connection starvation is first discussed as a part of the Eclipse attack in 2015 [7]. An adversary opens many connections to a target Bitcoin node and prevents other legitimate Bitcoin peers from establishing connections to it, effectively partitioning the target node from the rest of the peer-to-peer (P2P) network.

After the publication of the Eclipse attack, Bitcoin developers deployed several countermeasures, including a *peer eviction* mechanism in which a fully connected node terminates one of its existing connections to make room for a new incoming connection [3]. This peer eviction, along with several other measures, mitigates the Eclipse attacks by allowing legitimate nodes to connect to the partitioned target and thus deliver new blocks from the canonical chain; that is, the target node is no longer partitioned. In other words, the peer eviction mechanism makes the Eclipse attacks unsustainable.

Since then, there are two notable attacks showing partitioning the Bitcoin P2P network is still possible. In the Erebus attacks [12], a malicious autonomous system (AS) partitions a single target node by poisoning its internal peer database. SyncAttack [10] splits the entire Bitcoin P2P network into two partitions by monopolizing the incoming connection slots of all public Bitcoin nodes with a few tens of malicious peers. We observe, however, these studies have not properly evaluated the effect of the peer eviction mechanism against their attacks. Unfortunately, thus, sustainability of these attacks is left untested. Their original reports pay significant attention to the end-to-end attack evaluation, i.e., from the attack preparation phase to the attack completion phase with 100% connection occupation. Yet, the two studies do not show empirical evidence that their attacks are sustainable enough (e.g., partition continues while several blocks are generated), which is a critical condition for follow-up attacks, such as double-spending attacks or stubborn mining attacks [9]. The lack of rigorous evaluation on attack sustainability motivates us to test these partitioning attacks in our independent study.

To that end, we first test the *sustainability* of the original versions of the Erebus attack and SyncAttack. As we measure how long these attacks successfully partition a node (in the case of the Erebus attack) or two groups of nodes (in the case of the SyncAttack), we face a number of practical difficulties. For example, to accurately measure the moment when a partition ends, we should evaluate the Erebus attack in the Bitcoin mainnet, which has not been conducted in existing literature due to the challenging attack setup and ethical concerns. Evaluating SyncAttack accurately is even more challenging as it would create two partitions of thousands of nodes. We carefully apply several workarounds for these evaluations and estimate the attack sustainability effectively. Our evaluation of the sustainability of these partitioning attacks, the first of its kind, shows that the two attacks would not maintain successful network partitioning to some meaningful extent, significantly limiting the effectiveness of these attacks in practice.

After learning that the two existing partitioning attacks in their original versions fail to last long enough (e.g., for 1 hour) for follow-up exploits, we go one step further and optimize for their maximal sustainability. Our optimization requires thorough reviews of the original attack strategies and the current Bitcoin's peer connection mechanism. Our optimized attacks are designed against the up-to-date Bitcoin Core implementation v23.0 (as of October 2022).

We then measure the required cost of the optimized attacks when aiming to maintain partitions for a given duration (e.g., 1 hour). For useful risk analysis of these attacks in practice, we define the attack cost as two-dimensional resources: (1) the number of unique, adversary-controlled network address groups, and (2) the distance (in network latency) from adversary-controlled nodes to the target client(s). Our analysis of the attack cost for these optimized attacks shows that the Erebus attack can be sustainable (e.g., maintain a partition for an hour) with moderately more attack resources (compared to what is described in its original version) and a few additional attack steps. In contrast, the SyncAttack shows its feasibility only with excessive cost.

The organization of the paper is as follows. We first provide a brief overview of recent partitioning attacks in Section 2. We then test the attack sustainability of two unmodified attacks in Section 3. In Section 4, we present our new optimized attack strategy based on Bitcoin's peer connection mechanism and calculate the cost of 1-hour sustainability with these optimized attacks.

2 Related Work

In this section, we review a few recent Bitcoin partitioning attacks [6, 10–12]. We particularly focus on the Erebus attack and SyncAttack, and highlight their differences in Table 1. We also briefly review two partitioning attacks based on Internet routing manipulation (i.e., BGP hijacking), although evaluating their attack sustainability is beyond the scope of this paper.

	Targets	Stealthiness	Attack resources	Preparation time	Peer table manipulation	Evaluation in the original paper	Deployed countermeasures
Erebus Attack	Single Bitcoin node	Yes	an AS	5-6 weeks	Required	No direct evaluation, only simulation with AS topology	Two deployed at Bitcoin Core
Sync Attack	Entire Bitcoin network	No	125 Bitcoin nodes	Not mentioned	Not required	No direct evaluation (measurement in mainnet, experiment in testnet)	None

Table 1. Comparison of two known Bitcoin partitioning attacks.

2.1 Erebus: a Stealthy Single-Node Partitioning Attack

In the Erebus attack [12], a malicious AS, such as a tier-1 or tier-2 Internet Service Provider, fills a target node's peer tables (i.e., new and tried) with adversary-controlled IP addresses. Connections from the target to any of those IPs traverse through the adversary AS and therefore, are controlled by the adversary. The target is partitioned when all of its peer connections go through the adversary AS. As the sending rate of the attack payloads to the target is negligible and no routing manipulation is involved in both the data plane and control plane, it is difficult to detect the Erebus attacks. The original publication [12] shows that individual Bitcoin nodes can be partitioned, possibly in parallel, within 5-6 weeks of attack execution. Unfortunately, the attack sustainability is unknown since no experiments are given to show what happens after the Erebus attack successfully partitions a node.

2.2 SyncAttack: a P2P Network Splitting Attack

SyncAttack [10] aims to partition a group of Bitcoin nodes from the rest of the P2P network, i.e., splitting it into two. In the SyncAttack, the adversary occupies all inbound connection slots of all reachable nodes to prevent other nodes connect to them. The paper [10] argues that when all inbound slots of existing nodes are occupied, newly-joined nodes have no choice but to establish connections to the adversarial nodes. As a result, the P2P network would be split into two partitions: existing nodes versus new, arriving nodes. To make an existing node prioritizes the adversary-generated connections, the adversary sends fresh block and transaction data via 16 connections from the same IP and establishes other 99 connections from IPs with distinct network groups to it. Again, the attack sustainability is not properly evaluated in the original publication [10].

2.3 Bitcoin Partitioning Attacks Using Routing Manipulation

Apostolaki et al. [6] show that a malicious AS can perform BGP hijacking attacks against the IP prefixes hosting targeted Bitcoin nodes. Once the traffic to such prefixes is hijacked, the adversary AS drops Bitcoin traffic at the network layer, effectively cutting off the communication between targeted Bitcoin nodes and the rest of the P2P network. This is further extended in a multi-cryptocurrency attack by Saad et al. [11]. Since evaluating the sustainability of such attacks inherently requires measuring the sustainability of the routing manipulation, we leave it for future work.

3 Re-evaluating Existing Partitioning Attacks

We aim to evaluate the sustainability of the Erebus attack [12], and SyncAttack [10] — in the face of the peer eviction mechanism that has made the Eclipse attack unsustainable. In particular, we use *partitioning duration* as the metric to measure the sustainability of the two attacks. It indicates the time difference between the moment when the target node(s) is partitioned and the moment when it receives a new block from the canonical chain. To measure the partitioning duration, one needs to conduct the two attacks (i.e., partitioning a node or a group of nodes) successfully and waits for the partitions to cease.

Thus, in this section, we test the sustainability of two existing partitioning attacks and provide empirical evidence that they do not create effective partitioning to any meaningful extent in practice. In the following Section 3.1–3.2, we describe how we re-evaluate the two attacks and measure their sustainability in practical test environments.

3.1 Re-evaluation of the Original Erebus Attack

Now, let us explain how we test the original Erebus attack with the sustainability metric.

Direct implementation & evaluation of the original attack. First, we explain how we are supposed to conduct the experiments of the original Erebus attack, according to its publication [12]. It follows five steps:

(i) Deploy a target Bitcoin node on the mainnet.

(ii) Send many adversary-chosen IPs (called shadow IPs) from a malicious ISP with various network groups to the target node.

(iii) Wait until the target node's peer table is filled with the shadow IPs and the target makes all its outgoing connections to the shadow IPs.

(iv) Monopolize the target node's inbound connections by making connections from the malicious ISP.

(v) Measure how long the target node is partitioned, checking the moment the target node receives new blocks.

Challenges. Measuring the sustainability of the Erebus attack, however, involves several challenges.

- *Mainnet experiment required.* For measuring the attack sustainability, we should identify the exact moment other benign Bitcoin nodes relay new blocks to the target node. Therefore, ideally, the experiment should be performed in the mainnet in order to reflect other real Bitcoin nodes' behavior.
- *Need huge network resources.* The original Erebus attack requires an adversary to be an autonomous system (AS), utilizing 500K shadow IPs for occupying the target node's peer table. To experiment it as is in the mainnet, we need to be an ISP or we need to own and control many Bitcoin nodes with many distributed IP addresses, both of which are extremely hard to achieve.
- *Long attack execution time.* The Erebus attack requires up to several weeks to be successfully mounted, which is too long for making practical evaluations in the mainnet.

Our workarounds. Let us describe how we address the three challenges with our workarounds. To simplify the experiment, we only consider the target node's behavior *after* being partitioned. This is acceptable as we only measure the attack sustainability, not the end-to-end attack effectiveness. Instead of implementing the end-to-end Erebus attack, we make the target node behaves *as if* it is partitioned by the Erebus attack. For this attack emulation, we do the followings: (i) we set up a simple, temporary firewall to prevent the target node from making new outbound connections, and (ii) we occupy all inbound connections of the target by manually disconnecting its peers (via RPC calls). As a result, the target node cannot make outbound connections to other benign nodes, and all inbound connections are occupied by the attacker as if the Erebus attack is successfully launched. This short-cut implementation of the Erebus attack for our sustainability evaluation can detour the challenges above: (a) we do not need to be an ISP or own/control large numbers of IP addresses, as we skip occupying the target node's peer table; (b) the attack does not require a long execution time because we manually disconnect all peers from the target node; (c) due to (a) and (b), our experiments now do not disturb other benign nodes in the mainnet.

Results. Table 2 shows the overall result of the original Erebus attack's partitioning duration from a mainnet experiment. The original Erebus attack maintains its partitioning to the target node for 297 seconds on average, where most of the experiments have ended within 900 seconds.

Partitioning duration: average±stdev	90% percentile	95% percentile
297 sec ± 293 sec	679 sec	847 sec

Table 2. Measured attack duration of the Erebus attack. The attack does not maintain partitions longer than 10 minutes in most cases.

Why does the attack not last long? The partition ends when a benign node delivers new blocks to the target; however, the original Erebus attack design does *not* have attack features that prevent such a partition-breaking event triggered by benign peers. The target node faces frequent inbound connections from other benign peers due to Bitcoin's peer eviction mechanism (see Section 4 for more details), and some of them would try relaying recent blocks right after establishing a connection. Even when only one benign node tries to relay blocks to the target node, as there is no barricade, the partition is broken by the delivered blocks.

The attack usually ends in less than 900 seconds, which is in sufficient for follow-up attacks. For example, double spending needs to make the target node confirm a fake transaction, which requires several blocks to be generated while the target is being partitioned.

3.2 Re-evaluation of the Original SyncAttack

Now, let us re-evaluate the original SyncAttack, by applying the same metric as we evaluated the original Erebus attack in the previous section. SyncAttck differs from the Erebus attack in a number of ways. The most notable difference is its scale; i.e., SyncAttack aims to split the *entire* Bitcoin network into two whereas the Erebus attack seeks to partition a *single* node. Therefore, we need to consider new challenges and strategies to make a reasonable SyncAttack evaluation.

Direct implementation and evaluation of the original attack. First, we explain how we conduct the sustainability experiments for the original SyncAttack. Experiments should follow these steps:

(i) Prepare a network with two benign partitions (i.e., existing nodes and arriving nodes) and a small group of adversary nodes.

(ii) Add a new node to the network and let this new node establishes new outbound connections referring to the DNS seeder.

(iii) Check whether the partition is maintained. We consider a partition ends when the new node establishes outgoing connections to both benign partitions.

(iv) Repeat (ii)-(iii), and end the experiment when the partition ends.

Challenges. Here, we list the challenges of the implementations above:

- *Implementation of a large-scale network split.* In contrast to the Erebus attack, evaluating the original SyncAttack requires a test network of thousands of Bitcoin clients to split into two. It is nearly impossible to conduct such an experiment in the mainnet. Simulating it would be challenging as well.
- *Implementation of churn-ins and their bootstrapping.* The Bitcoin network in practice experiences high churns, and new incoming nodes are important

Proportion of adversary nodes in seeder	71.4%	50%	33.3%
Prob. of maintaining the partition after single node arrival	12.5%	3.7%	1.5%
Prob. of maintaining the partition for 1 hour	0	0	0

Table 3. Probability of maintaining partitions with SyncAttack for (i) every single arriving node, and (ii) multiple arriving nodes within an hour. Partitions do not last at all after having multiple arriving nodes in an hour.

components for SyncAttack. When a new node enters the Bitcoin network, it requests reachable nodes' IP addresses to DNS seeds.

Our workarounds. Let us explain how we realize the re-evaluation of SyncAttack with the following workarounds.

- *Evaluation with over-estimations.* Evaluating SyncAttack is challenging mainly due to the size and complexity of the attack execution. To reduce the experiments to a manageable size and manage the complexity while offering meaningful experiment results, we simplify our sustainability experiments with over-estimation. That is, we take a number of reduction strategies for our experiments to obtain the attack sustainability results that are strictly longer than the actual attack sustainability. We ensure at every reduction step that the attacks in our simplified evaluations are strictly easier to sustain than the large-scale attacks in the mainnet. We argue that this simplification makes an effective experiment for proving the *unsustainability* of SyncAttack. By showing that the strictly easier SyncAttack cannot achieve sustainability, we can show that the original SyncAttack must also be unsustainable.

  First, we reduce the size of the experiment by scaling down the original SyncAttack's strategy in the regtest network. Our regtest network consists of three types of nodes: 10 adversary nodes, and existing/arriving nodes with the same number. To use the terminologies in the original publications of SyncAttack [10], the adversary nodes take the roles of A_u (occupy all available inbound slots of reachable nodes) and A_r (occupy all available outbound slots of reachable nodes). This is a strict advantage to the adversary since it is easier to maintain the partition in a small-scale test network rather than in mainnet.

  Second, we make an assumption that the adversary occupies almost all inbound connection slots of reachable nodes. Under this assumption, the adversary has a higher chance of maintaining the partition because new partition-breaking connections from existing nodes would not occur.

- *Implementing churn-ins with a real DNS seeder.* In our evaluation framework, we continuously add new Bitcoin nodes into the network and test whether SyncAttack successfully continues to partition the target network. For a realistic experiment of SyncAttack, we deploy a custom DNS seeder [1], and include all IP addresses of existing nodes, arriving nodes and adversary nodes in our experiment setup into the seeder. Each new Bitcoin node fetches IP addresses from the DNS seeder and establishes 10 outgoing connections among them.

Results. Table 3 shows the probability of maintaining the partitioning in SyncAttack by varying proportion of adversary nodes in the seeder. The experiment is conducted 1,000 times for each condition, checking if partitioning can be maintained after a new single node arrives in the network. The partitioning duration of SyncAttack turns out to be *extremely short*. Even in the case where the adversary occupies half of the IP lists in the DNS seeder, the probability of maintaining the partition after *single* node arrival is 3.7%. If the attacker tries to maintain the partitioning for an hour (i.e. minimum partitioning duration for performing double-spending), it has to withstand all new arriving nodes during that time. According to Bitnodes [13], 851 nodes joined the Bitcoin network every hour on average; the probability of maintaining the partitioning for an hour converges to zero. By looking at the fact that SyncAttack would not sustain in such advantageous conditions (i.e., controlling over the network with only a few dozen of nodes, and adversary occupying almost all inbound connection slots of reachable nodes), we claim that SyncAttack execution in mainnet is not feasible after all.

4 Optimization and Cost Analysis

Knowing that the two original attacks [10, 12] are unsustainable in practice, we aim to improve these attacks to see whether sustainable Bitcoin partitioning attacks are possible in practice. To make stronger final conclusions about their sustainability, we aim to optimize the attacks by maximizing the partitioning duration in consideration of Bitcoin's peer eviction mechanism in Section 4.1.

For accurate and realistic cost analysis, we measure the attack resources required for the optimized attacks in practice. We define and model the necessary resources for these optimized attacks as a two-dimensional attack resource vector: (1) the number of distinct network groups controlled by the adversary (i.e., netgroup cost), and (2) the physical distance to the target node(s) (i.e., latency cost). Upon this, we derive the required cost of the optimized attacks when attempting to maintain partitioning for a given time duration (e.g., 1 hour). Our optimization often requires significantly more attack resources than the original attacks to attain certain sustainability. Our analysis of the attack cost in these optimized attacks concludes that a node partition made by the optimized Erebus attack can be sustained with reasonably high investment (e.g., 100 unique network groups along with adversary closer to the target than 95% of all inbound connections towards it) while the optimized SyncAttack becomes sustainable only with extremely high attack resources (e.g., 22K unique network groups along with adversary closer to each target in the reachable node than 95% of all inbound connections towards them).

4.1 Optimization of Two Original Attacks

We first look back at why the original partitioning attacks fail to sustain. A successful partitioning attack campaign begins to fail when a new benign connection is made to the target, and starts to deliver new blocks. This means that

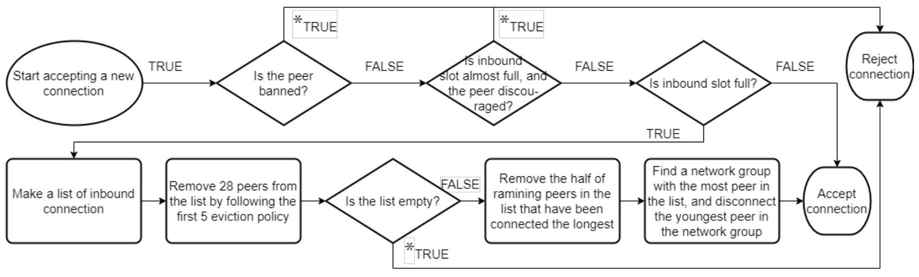
Start accepting a new connection
TRUE
Is the peer banned?
*TRUE
FALSE
Is inbound slot almost full, and the peer discou-raged?
*TRUE
FALSE
Is inbound slot full?
FALSE
TRUE
Reject connection
Make a list of inbound connection
Remove 28 peers from the list by following the first 5 eviction policy
Is the list empty?
FALSE
*TRUE
Remove the half of ramining peers in the list that have been connected the longest
Find a network group with the most peer in the list, and disconnect the youngest peer in the network group
Accept connection

Fig. 1. Flow chart of Bitcoin's connection handling logic. An unmodified benign Bitcoin node always accepts new incoming connections, with the exception of three abnormal cases (marked with *) that occur by extremely unlikely chance.

to maximize attack sustainability, we should prevent new benign connections from delivering new blocks to our target(s). To this end, we investigate Bitcoin's peer connection mechanism as it is where the decision to allow a new connection to the target node is made. Therefore, we investigate Bitcoin's peer connection management logic since now it is essential to know when and how a new connection is made to the target node. As we optimize the partitioning attacks, we optimize their attack strategies with respect to Bitcoin's peer connection management logic, particularly, the peer eviction mechanism.

Bitcoin's peer connection management logic. Figure 1 shows the overall flow diagram of Bitcoin's peer connection management logic. We analyze the CreateNodeFromAcceptedSocket() in net.cpp and extract this diagram without any omission. We summarize the two important design principles:

- *Admitting inbound connections (almost) unconditionally.* The Bitcoin client admits new inbound connections unconditionally in all practical scenarios except for the following three abnormal, unlikely cases:
  (i) the client has manually banned the new peer.
  (ii) the new peer has violated the Bitcoin protocol and is thus considered discouraged.
  (iii) the client has manually reduced the maximum number of inbound connections.
- *Evicting the least prioritized.* A new inbound connection may cause an eviction of an existing peer connection in order to free up one connection slot for this new inbound connection. Existing peer connections are ordered by their priority and only the least prioritized one is evicted.

Our conclusions drawn from these principles are two-fold. First, the remote attacker cannot prevent new benign connections from being made to the target node unless it can force the target and the benign nodes to turn into abnormal states, which is impossible. Second, therefore, partitioning adversaries have to evict any new benign connection before it delivers new blocks from the canonical blockchain.

Probability of withstanding a *single* inbound connection. Now, our goal is narrowed down to maintaining a partition by evicting new benign connections before they deliver new blocks to the target. As a target node may receive multiple new connections from other benign peers while being partitioned, we should evict multiple such benign connections. We can consider each new benign connection is made *independently* since benign peers are not coordinated in the permissionless Bitcoin network and also incoming connections do not change the target's internal state with respect to the peer eviction logic (unless the benign connection breaks the partition).

Let us define p_{pc} ($\underline{p}$artitioning $\underline{c}$ontinues), the probability the existing partitioning continues when a *single* new benign connection is made to the target. In other words, it is the probability that a new benign peer is evicted before delivering any new block. We want to ensure that we achieve p_{pc} very close to 1.0 since the target is partitioned until the first new inbound connection is made with the probability of $1 - p_{pc}$. Because all these events are independent, the expectation of the number of new inbound connections the adversary can withstand is $\frac{1}{(1-p_{pc})}$. For example, a five-nines probability for $p_{pc} = 0.99999$ ensures that partitioning can last after about $100,000$ new inbound connections on average.

Breaking down p_{pc}. Given the current Bitcoin's peer connection management logic, we summarize and propose the two orthogonal optimization strategies for maximizing p_{pc}. These two optimizations must be conducted independently to maximize the overall partitioning duration.

The current Bitcoin's peer eviction policy has three rules[3] that prioritize 28 peer connections so that they are not evicted for a new (attacker's) connection:

- Rule ① *Netgroup-based prioritization.* Prioritize 4 peers with top-4 netgroupkey values.
- Rule ② *Ping-based prioritization.* Prioritize 8 peers with the lowest minimum ping times.
- Rule ③ *Message-history-based prioritization.* Prioritize 16 peers that have exchanged recent blocks and transactions.

The probability p_{pc} indeed is equivalent to the probability that the newly-made benign connection we want to evict is *not* prioritized by these rules. This is because, after prioritizing 28 peer connections according to the rules above, the target Bitcoin node always selects a single, the least prioritized connection (i.e., the youngest peer in the network group with the most connections) to evict among the rest. Thus, now we analyze these three rules and the adversary's chance of having the benign connection not prioritized after all.

First, p_{nt4} ($\underline{n}$on-$\underline{t}$op-$\underline{4}$), the probability that a new benign connection has non-top-4 netgroupkey value at the target. This can be increased by making many connections to the target with many unique netgroups. Second, p_{ebp} ($\underline{e}$victed-$\underline{b}$efore-$\underline{p}$ing): the probability that a new benign connection is evicted before its ping time (with respect to the target) is calculated. This is related to how quickly

[3] We exclude some rare-case rules found in the Bitcoin Core implementation, which are not useful for our current discussion.

an adversary can send new attack connection(s) to the target and evict the new benign connection before it finishes the ping calculation. This probability can be increased by locating the attacker's nodes very close to the target. Third, p_{ebbt} ($\underline{e}$victed-$\underline{b}$efore-$\underline{b}$lock/$\underline{t}$x): the probability that a new benign connection is evicted before any blocks or transactions are sent to the target. This probability can also be increased by locating the adversary's nodes close to the target.

Designing two orthogonal optimization strategies. Finally, we design detailed attack strategies. We first separate out the optimization for p_{nt4} since an event of being prioritized by Rule ① is independent of the events of being prioritized by the latter two rules. This is because the network groups of inbound connections are irrelevant to how quickly the adversary's new connections can be made to the target. In other words, having many network groups as attack resources do not have a direct correlation with getting physically close to the target. Therefore, our first attack strategy deals with Rule ① should maximize p_{nt4} by raising the bar to be selected as the top-4 netgroupkey values.

Our second attack strategy aims to optimize p_{ebp} by minimizing the time to send attacker's new connections to the target so that the admitted benign connection is not prioritized either by Rule ② or Rule ③. In practice, ping time is always calculated before allowing a new peer to send any blocks or transactions. Thus, Rule ③ can be ignored since the second attack strategy, if successful, prevents any blocks or transaction messages from a benign connection after all.

Therefore, the probability p_{pc} is simply derived as the multiplication of the two independent probabilities p_{nt4} and p_{ebp}, which is given by

$$p_{pc} = p_{nt4} \cdot p_{ebp} \tag{1}$$

Let us provide some more details on these two orthogonal attack strategies.

Netgroup-flooding. To maximize p_{nt4}, our attacker should flood connections to the target node with as many unique network groups as possible. To describe why such a simple (yet expensive) strategy ensures optimality, let us quickly sketch how the current Bitcoin implementation prioritizes nodes with certain netgroups. From Bitcoin 0.12.0, Bitcoin prioritizes four peer connections with the largest netgroupkeys among its inbound peers. The netgroupkey is calculated in a deterministic manner using a cryptographic hash function (i.e., SipHash) with a random seed that is known only to each node. The network group (i.e., the upper 16-bit of IPv4 addresses) is used as an input to the hash function and its digest becomes the netgroupkey of each peer. This way, the attacker cannot predict which network groups will be prioritized. As a result, the only way for the attacker to maximize p_{nt4} is to raise the bar for being selected as the top-4 netgroupkey value by flooding connections with many unique network groups.

Evict-before-ping. To maximize p_{ebp}, our attacker should make her new connections to the target node before it finishes the ping computation with the new benign connection. To achieve that, the attacker must first fill up all the available inbound connection slots of the target node in order to get notified whenever a new benign connection is made.

Next, to ensure that the target node receives and accepts the attacker's new connection request before finishing a ping computation of the new benign connection, the adversary should be located as close to the target as possible. We consider a number of practical scenarios to minimize the ping distance to the target node. First, if a target is in a public cloud (e.g., AWS, Google Cloud), an adversary can create instances at the same cloud data center and measure ping distance to the target until it obtains a small enough (e.g., a few milliseconds) ping value. Second, if a target is not in a public cloud but its approximate location is known by its IP address (e.g., [2], [4], [5]), an adversary may attempt to find bots or public clouds that are close enough to the target.

We accomplish the optimization strategies above by implementing the following steps with our simple attack script: (1) the adversary makes 114 connections (with nodes located close to the target node) to the target concurrently by establishing VERSION handshakes; (2) each connection performs a ping-pong exchange with the target every 2 minutes to stay alive; (3) each connection immediately reconnects to the target as soon as its TCP session with the target is terminated due to eviction. This way, we can ensure that the adversary is notified of the new benign connection, and evict it with minimum delays.

4.2 Cost Analysis of the Optimized Attacks

Finally, we evaluate the optimized attack strategies and estimate the required attack costs. Note again that we focus on the sustainability of these attacks and the cost incurred by maintaining an existing partition using the two attacks.

Cost analysis with no real-world experiments. We estimate the required cost for the sustainability of the two attacks *without* real-world experiments because making several simplifying assumptions (as we did for the sustainability tests in Section 3.1-3.2) is not allowed here. Instead, we derive analytical frameworks that allow us to compute the attack duration of the two attacks for varying the two-dimensional attack costs.

As shown in Eq. (1), optimizing the sustainability of these partitioning attacks is composed of two orthogonal attack strategies. The two strategies require two different attack resources: maximizing p_{nt4} requires a large number of adversary-controlled IP addresses with unique netgroups (the more unique netgroups, the higher p_{nt4}) while maximizing p_{ebp} requires adversary-controlled nodes that are close to the target node (the closer to the target, the higher p_{ebp}). We describe how we derive two costs.

Netgroup cost. As mentioned earlier, in each node, peers with top-4 netgroupkey values are prioritized. The adversary can raise the threshold for being chosen as the top-4 netgroupkey values by making connections from numerous distinct network groups. If the adversary can use G unique network groups to make connections to the target, p_{nt4} is simply derived as $\binom{G}{4}/\binom{G+1}{4} = \frac{G-3}{G+1}$ because its value should be smaller than top-4 values among $G+1$ unique network groups (G from the adversary, 1 from the benign peer).

Latency cost. For a successful evict-before-ping event, our adversary nodes should reconnect to the target earlier than a new benign connection. To be more specific, the adversary should reach the target *before* the target receives a *pong* message from the new benign connection. Analyzing the Bitcoin Core's

Netgroup\\P_{ebp}	0.9	0.95	0.97	0.99
10	0.001	0.002	0.003	0.004
20	0.022	0.043	0.055	0.070
50	0.106	0.203	0.260	0.333
100	0.174	0.333	0.427	0.546
1,000	0.269	0.515	0.661	0.845
10,000	0.281	0.538	0.691	0.882
22,000	0.282	0.539	0.692	0.884

Table 4. Probability of maintaining partitions with the optimized Erebus attack for one hour. With significant attack resources (e.g., owning 100 unique network groups with $p_{ebp} = 0.95, 0.99$), partitions can be maintained with moderate probability.

network protocol, we learn that two round-trip times (RTTs) are required for a benign node to finish a ping-time calculation and the same two RTTs are needed for an adversary node to make another connection to the target after being notified of the new benign connection. Therefore, ignoring minor perturbations in end-to-end network latency between these nodes, an adversary can continue its partitioning (i.e., succeed evict-before-ping) if her node is *closer* to the target compared to the benign peer; otherwise, the partition may end because of this new benign connection. We empirically confirm this with five mainnet Bitcoin nodes in five different locations in Amazon EC2. From this, we state that the adversary can maximize p_{ebp} by getting closer than other benign nodes with respect to the target node.

Final cost estimation of the two optimized attacks. We finally derive the estimated cost for the 1-hour sustainability of the two optimized attacks. Table 4 shows the probability of maintaining a partition with the optimized Erebus attack. Each value is derived by ${p_{pc}}^n = (p_{nt4} \cdot p_{ebp})^n$, where n is the number of benign peers that adversary has to withstand for one hour. In Section 3.1, we observe that about $n = 12$ ($\simeq 3,600/297$) new benign connections must be handled properly for the goal of 1-hour sustainability. As the Erebus attack partitions a single node, the adversary can locate her attack nodes close to the target and achieve a high p_{ebp} (e.g., 0.95, 0.97, 0.99). If the Erebus attacker owns 100 unique network groups (as it claims to be in the original paper), it can sustain the partitioning for an hour with a probability of 33% and 55% with p_{ebp}=0.95 and 0.99, respectively. With some more attack network resources, say 1,000 network groups, we can expect 52% and 85% of successful 1-hour attacks with p_{ebp}=0.95 and 0.99, respectively. Having unique network groups beyond 1,000, however, only marginally improves the attack duration. Note that the maximum unique netgroups we test is 22,000, which is the unique netgroups we find in a large Mirai botnet [8]

Table 5 shows the probability of maintaining a partition with the optimized SyncAttack. We assume benign reachable nodes are partitioned into two groups of size A and B, while the adversary owns G reachable nodes with unique net-

Netgroup\P_{ebp}	0.9	0.95	0.97	0.99
1,000	0.00	0.00	0.00	0.00
10,000	0.00	0.00	0.00	0.02
22,000	0.00	0.00	0.0013	0.44

Table 5. Probability of maintaining partitions with the optimized SyncAttack for one hour, suggesting that SyncAttack is nearly impossible to sustain despite the optimization.

work groups. Let us consider a single new arriving node to the network due to network churn. Each reachable node may end up establishing a new reliable connection from this new node with a probability $(1 - p_{pc})$ if the new node makes a connection request. Since the eviction process at each node is independent of each other, we can say that on average $A' = (1 - p_{pc})A$ nodes in one group and $B' = (1 - p_{pc})B$ nodes in the other group would be willing to establish reliable peer connections from the new benign node (again, if the benign node wishes to do so).

The new node can make a new connection to any of the three groups: A' nodes in one partition, B' nodes in the other partition, and X adversary nodes. The new node repeats reconnecting to other nodes when its connection is evicted quickly due to the optimal partitioning attack strategies until it finally makes a reliable peer connection to a reachable node. Thus, for the current partition to continue, a new connection from the new node should not be made to the first two groups at the same time (because that would bridge the two partitions). The probability for withstanding single arriving node P_{single} in SyncAttack would therefore be given by

$$P_{single} = \left(\frac{A' + G}{A' + B' + G}\right)^{10} + \left(\frac{B' + G}{A' + B' + G}\right)^{10} - \left(\frac{G}{A' + B' + G}\right)^{10}. \quad (2)$$

For Table 5, we consider two partitions of size 7,000 each (i.e., $A = B = 7,000$) [13]. For the number of new peer nodes per hour, we used the churn rate we measure earlier in Section 3.2; that is, we assume that 851 new peers appear on the mainnet for an hour on average.The overall probability for maintaining partition for an hour with the optimized SyncAttack is thus derived as ${P_{single}}^{851}$. SyncAttack turns out to be unsustainable in most cases even with optimization strategies. Even in the case where the adversary has attack resources with 22K network groups, it is hard to expect that SyncAttack would sustain for an hour (i.e., 0% with $p_{ebp} = 0.95$, 1% with $p_{ebp} = 0.97$). Our experiment shows that SyncAttack may maintain its partitioning with some reasonable probability of 44% *only* when it has a significant amount of attack resources. First, the adversary must control 22K or more nodes with unique network groups. This can be achieved only when the adversary has mega-size botnets (e.g., a Mirai botnet of 2.3M bots has 22K unique network groups). Second, the adversary should be close to all reachable nodes. That is, the adversary must be co-located with all the 14K reachable nodes in the Bitcoin network to achieve $p_{ebp} = 0.99$ (i.e., ensuring 99th percentile in terms of RTT distance for all the nodes). This appears to be challenging to achieve because reachable Bitcoin nodes are distributed in various networks.

5 Conclusion

That blockchain networks can be partitioned by unauthorized adversaries is still a serious threat to their security and safety. Our work helps us to further characterize existing partitioning attacks with the notion of attack sustainability, which is a critical yet less-emphasized metric for partitioning attacks. We hope this work guides the direction for developing additional countermeasures against partitioning attacks in Bitcoin and other blockchain networks.

References

1. Custom DNS seeder, dnsseed.netsptest.com.
2. Maxmind GeoIP2 Databases, https://www.maxmind.com/en/geoip2-databases
3. The peer eviction mechanism implementation of Bitcoin Core (Aug 2015), https://github.com/bitcoin/bitcoin/commit/2c701537c8fc7f4cfb0163ec1f49662120e61eb7
4. db-ip IP Address Databases (2022), https://db-ip.com/db/
5. IPinfo IP Address Databases (2022), https://ipinfo.io/products/ip-database-download
6. Apostolaki, M., Zohar, A., Vanbever, L.: Hijacking Bitcoin: Routing attacks on cryptocurrencies. In: Proc. IEEE S&P (2017)
7. Heilman, E., Kendler, A., Zohar, A., Goldberg, S.: Eclipse Attacks on Bitcoin's Peer-to-Peer Network. In: Proc. USENIX Security (2015)
8. Netlab360's Mirai Scanner, http://data.netlab.360.com/mirai-scanner/
9. Nayak, K., Kumar, S., Miller, A., Shi, E.: Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. In: 2016 IEEE European Symposium on Security and Privacy (EuroS&P). pp. 305–320. IEEE (2016)
10. Saad, M., Chen, S., Mohaisen, D.: SyncAttack: Double-spending in Bitcoin Without Mining Power. In: ACM CCS (2021)
11. Saad, M., Mohaisen, D.: Three Birds with One Stone: Efficient Partitioning Attacks on Interdependent Cryptocurrency Networks. In: Proc. IEEE S&P (2023)
12. Tran, M., Choi, I., Moon, G.J., Vu, A.V., Kang, M.S.: A Stealthier Partitioning Attack against Bitcoin Peer-to-Peer Network. In: Proc. IEEE S&P (2020)
13. Yeow, A.: Global Bitcoin nodes distribution (2021), https://bitnodes.io/