Hijacking Bitcoin: Routing Attacks on Cryptocurrencies

https://btc-hijack.ethz.ch

Maria Apostolaki
ETH Zürich
apmaria@ethz.ch

Aviv Zohar
The Hebrew University
avivz@cs.huji.ac.il

Laurent Vanbever
ETH Zürich
lvanbever@ethz.ch

arXiv:1605.07524v2 [cs.NI] 24 Mar 2017

***Abstract*—As the most successful cryptocurrency to date, Bitcoin constitutes a target of choice for attackers. While many attack vectors have already been uncovered, one important vector has been left out though: attacking the currency via the Internet routing infrastructure itself. Indeed, by manipulating routing advertisements (BGP hijacks) or by naturally intercepting traffic, Autonomous Systems (ASes) can intercept and manipulate a large fraction of Bitcoin traffic.**

This paper presents the first taxonomy of routing attacks and their impact on Bitcoin, considering both small-scale attacks, targeting individual nodes, and large-scale attacks, targeting the network as a whole. While challenging, we show that two key properties make routing attacks practical: *(i)* the efficiency of routing manipulation; and *(ii)* the significant centralization of Bitcoin in terms of mining and routing. Specifically, we find that any network attacker can hijack few (<100) BGP prefixes to isolate ∼50% of the mining power—even when considering that mining pools are heavily multi-homed. We also show that on-path network attackers can considerably slow down block propagation by interfering with few key Bitcoin messages.

We demonstrate the feasibility of each attack against the deployed Bitcoin software. We also quantify their effectiveness on the current Bitcoin topology using data collected from a Bitcoin supernode combined with BGP routing data.

The potential damage to Bitcoin is worrying. By isolating parts of the network or delaying block propagation, attackers can cause a significant amount of mining power to be wasted, leading to revenue losses and enabling a wide range of exploits such as double spending. To prevent such effects in practice, we provide both short and long-term countermeasures, some of which can be deployed immediately.

I. Introduction

With more than 16 million bitcoins valued at ∼17 billion USD and up to 300,000 daily transactions (March 2017), Bitcoin is the most successful cryptocurrency to date. Remarkably, Bitcoin has achieved this as an open and fully decentralized system. Instead of relying on a central entity, Bitcoin nodes build a large overlay network between them and use consensus to agree on a set of transactions recorded within Bitcoin's core data structure: the blockchain. Anyone is free to participate in the network which boasts more than 6,000 nodes [4] and can usually connect to any other node.

Given the amount of money at stake, Bitcoin is an obvious target for attackers. Indeed, numerous attacks have been described targeting different aspects of the system including: double spending [43], eclipsing [31], transaction malleability [21], or attacks targeting mining [24], [44], [38] and mining pools [23].

One important attack vector has been overlooked though: attacking Bitcoin via the Internet infrastructure using *routing attacks*. As Bitcoin connections are routed over the Internet—in clear text and without integrity checks—any third-party on the forwarding path can eavesdrop, drop, modify, inject, or delay Bitcoin messages such as blocks or transactions. Detecting such attackers is challenging as it requires inferring the exact forwarding paths taken by the Bitcoin traffic using measurements (e.g., traceroute) or routing data (BGP announcements), both of which can be forged [41]. Even ignoring detectability, mitigating network attacks is also hard as it is essentially a human-driven process consisting of filtering, routing around or disconnecting the attacker. As an illustration, it took Youtube close to 3 hours to locate and resolve rogue BGP announcements targeting its infrastructure in 2008 [6]. More recent examples of routing attacks such as [51] (resp. [52]) took 9 (resp. 2) hours to resolve in November (resp. June) 2015.

One of the reasons why routing attacks have been overlooked in Bitcoin is that they are often considered too challenging to be practical. Indeed, perturbing a vast peer-to-peer network which uses random flooding is hard as an attacker would have to intercept many connections to have any impact. Yet, two key characteristics of the Internet's infrastructure make routing attacks against Bitcoin possible: *(i)* the efficiency of routing manipulation (BGP hijacks); and *(ii)* the centralization of Bitcoin from the routing perspective. First, individuals, located anywhere on the Internet, can manipulate routing to intercept all the connections to not only one, but many Bitcoin nodes. As we show in this paper, these routing manipulations are prevalent today and do divert Bitcoin traffic. Second, few ASes host most of the nodes and mining power, while others intercept a considerable fraction of the connections.

This work In this paper, we present the first taxonomy of routing attacks on Bitcoin, a comprehensive study of their impact, and a list of deployable countermeasures. We consider two general attacks that AS-level attackers can perform. First, we evaluate the ability of attackers to isolate a set of nodes from the Bitcoin network, effectively partitioning it. Second, we evaluate the impact of delaying block propagation by manipulating a small number of key Bitcoin messages. For both exploits, we consider *node-level* attacks along with more challenging, but also more disruptive, *network-wide* attacks.

Partitioning attacks The goal of a partition attack is to *completely* disconnect a set of nodes from the network. This requires the attacker to divert and cut all the connections between the set of nodes and the rest of the network.

We describe a complete attack procedure in which an attacker can *verifiably* isolate a selected set of nodes using BGP hijacks. Our procedure is practical and only requires basic knowledge of the Bitcoin topology, namely the IP addresses of the nodes the attacker wants to isolate. Due to the complexity of the Bitcoin network (e.g. multi-homed pools, and secret peering agreements between pools), the initial isolated set might contain nodes that leak information from and to the rest of the network. We explain how the attacker can identify and remove these leakage points until the partition is complete.

Delay attacks The goal of a delay attack is to slow down the propagation of blocks towards or from a given set of nodes. Unlike partition attacks, which require a perfect cut, delay attacks are effective even when a subset of the connections are intercepted. As such, attackers can perform delay attacks on connections they are naturally intercepting, making them even harder to detect.

We again describe a complete attack procedure an attacker can run on intercepted Bitcoin traffic so that the delivery of blocks is delayed by up to 20 minutes. The procedure consists of modifying few key Bitcoin messages while making sure that the connections are not disrupted.

Practicality We showcase the practicality of each attack and evaluate their network-wide impact using a comprehensive set of measurements, simulations and experiments.

Regarding partitioning attacks, we show that hijacks are effective in diverting Bitcoin traffic by performing a hijack in the wild against our own nodes. We find that it takes less than 90 seconds to re-route all traffic flows through the attacker once a hijack is initiated. We also show that *any AS* in the Internet hijacking *less than 100* prefixes can isolate up to 47% of the mining power, and this, even when considering that mining pools are multi-homed. Hijacks involving that many prefixes are frequent and already divert Bitcoin traffic.

Regarding delay attacks, we show that an attacker intercepting 50% of a node connections can leave it uninformed of the most recent Bitcoin blocks ~60% of the time. We also show that intercepting a considerable percentage of Bitcoin traffic is practical due to the centralization of Bitcoin at the routing level: one AS, namely Hurricane Electric, can *naturally* intercept more than 30% of *all* Bitcoin connections.

Impact on Bitcoin The damages caused to Bitcoin in case of a successful routing attack can be substantial. By isolating a part of the network or delaying the propagation of blocks, attackers can force nodes to waste part of their mining power as some of the blocks they create are discarded. Partitioning also enables the attacker to filter transactions that clients try to include in the blockchain. In both cases, miners lose potential revenue from mining and render the network more susceptible to double spending attacks as well as to selfish mining attacks [24]. Nodes representing merchants, exchanges and other large entities are thus unable to secure their transactions, or may not be able to broadcast them to the network to begin with. The resulting longer-term loss of trust in Bitcoin security may trigger a loss of value for Bitcoin. Attackers may even short Bitcoin and gain from the resulting devaluation [35].

Our work underscores the importance of proposed modifications which argue for encrypting Bitcoin traffic [47] or traffic exchanged among miners [34]. Yet, we stress that not all routing attacks will be solved by such measures since attackers can still disrupt connectivity and isolate nodes by dropping Bitcoin packets instead of modifying them.

Contributions Our main contributions are:[1]

- The first comprehensive study of network attacks on Bitcoin (Section III) ranging from attacks targeting a single node to attacks affecting the network as a whole.
- A measurement study of the routing properties of Bitcoin (Section VI). We show that Bitcoin is highly centralized: few ASes host most of the nodes while others intercept a considerable fraction of the connections.
- A thorough evaluation of the practicality of routing attacks (partitioning and delay attacks). Our evaluation is based on an extensive set of measurements, large-scale simulations and experiments on the actual Bitcoin software and network.
- A comprehensive set of countermeasures (Section IX), which can benefit even early adopters.

While our measurements are Bitcoin-specific, they carry important lessons for other cryptocurrencies which rely on a randomly structured peer-to-peer network atop of the Internet, such as Ethereum [1], Litecoin [9], and ZCash [14], [45].

II. Background

A. BGP

Protocol BGP [42] is the de-facto routing protocol that regulates how IP packets are forwarded in the Internet. Routes associated with different IP prefixes are exchanged between neighboring networks or Autonomous Systems (AS). For any given IP prefix, one AS (the origin) is responsible for the original route advertisement, which is then propagated AS-by-AS until all ASes learn about it. Routers then set their next hop and pick one of the available routes offered by their neighbors (this is done independently for each destination).

In BGP, the validity of route announcements is not checked. In effect, this means that any AS can inject forged information on how to reach one or more IP prefixes, leading other ASes to send traffic to the wrong location. These rogue advertisements, known as BGP "hijacks", are a very effective way for an attacker to intercept traffic en route to a legitimate destination.

BGP hijack An attacker, who wishes to attract all the traffic for a legitimate prefix p (say, 100.0.0.0/16) by hijacking could either: *(i)* announce p; or *(ii)* announce a more-specific (longer)

[1]Our software, measurements and scripts can be found online at `https://btc-hijack.ethz.ch`

prefix of p. In the first case, the attacker's route will be in direct competition with the legitimate route. As BGP routers prefer shorter paths, the attacker will, on average, attract 50% of the traffic [30]. In the second case, the attacker will attract all the traffic (originated anywhere on the Internet) addressed to the destination as Internet routers forward traffic according to the longest-match entry. Note that traffic internal to an AS cannot be diverted via hijacking as it does not get routed by BGP but by internal routing protocols (e.g., OSPF).

For instance, in order to attract all traffic destined to p, the attacker could advertise 100.0.0.0/17 *and* 100.0.128.0/17. Routers in the entire Internet would then start forwarding any traffic destined to the original /16 prefix according to the two covering /17s originated by the adversary. Advertising more-specific prefixes has its limits though as BGP operators will often filter prefixes longer than /24 [33]. Yet, we show that the vast majority of Bitcoin nodes is hosted in shorter prefixes (Section VI) and is thus susceptible to hijacking.

By default, hijacking a prefix creates a black hole at the attacker's location. However, the attacker can turn a hijack into an *interception* attack simply by making sure she leaves at least one path untouched to the destination [41], [30].

B. Bitcoin

Transactions Transaction validation requires nodes to be aware of the ownership of funds and the balance of each Bitcoin address. All this information can be learned from the Bitcoin blockchain: an authenticated data structure that effectively forms a ledger of all accepted transactions. Bitcoin main innovation lies in its ability to synchronize the blockchain in an asynchronous way, with attackers possibly attempting to disrupt the process. Synchronization is crucial: conflicting transactions attempting to transfer the exact same bitcoins to different destinations may otherwise be approved by miners that are unaware of each other.

Block creation Bitcoin's blockchain is comprised of blocks, batches of transactions, that are appended to the ledger serially. Each block contains a cryptographic hash of its predecessor, which identifies its place in the chain, and a proof-of-work. The proof-of-work serves to make block creation difficult and reduces the conflicts in the system. Conflicts, which take the form of blocks that extend the same parent, represent alternative sets of accepted transactions. Nodes converge to a single agreed version by selecting the chain containing the highest amount of computational work as the valid version (usually the longest chain). The proof-of-work also serves to limit the ability of attackers to subvert the system: they cannot easily create many blocks, which would potentially allow them to create a longer alternative chain that will be adopted by nodes and thus reverse the transfer of funds (double spend).

The difficulty of block creation is set so that one block is created in the network every 10 minutes on average which is designed to allow sufficient time for blocks to propagate through the network. However, if delays are high compared to the block creation rate, many forks occur in the chain as blocks are created in parallel. In this case, the rate of discarded blocks (known as the *orphan rate* or the *fork rate*) increases and the security of the protocol deteriorates [20], [26], [49]. Newly created blocks are propagated through the network using a gossip protocol. In addition to the propagation of blocks, nodes also propagate transactions between them that await inclusion in the chain by whichever node creates the next block.

Network formation Bitcoin acts as a peer-to-peer network with each node maintaining a list of IP addresses of potential peers. The list is bootstrapped via a DNS server, and additional addresses are exchanged between peers. By default, each node randomly initiates 8 *unencrypted* TCP connections to peers in different /16 prefixes. Nodes additionally accept connections initiated by others (by default on port 8333). The total number of connections nodes can make is 125 by default.

Nodes continually listen to block announcements which are sent via INV messages containing the hash of the announced block. If a node determines that it does not hold a newly announced block, it sends a GETDATA message to a *single* neighbor. The peer then responds by sending the requested information in a BLOCK message. Blocks that are requested and do not arrive within 20 minutes trigger the disconnection of the peer and are requested from another. Transaction propagation occurs with a similar sequence of INV, GETDATA, and TX messages in which nodes announce, request, and share transactions that have not yet been included in the blockchain.

Mining pools Mining pools represent groups of miners that divide block creation rewards between them in order to lower the high economic risk associated with infrequent (but high) payments. They usually operate using the Stratum protocol [15]. The pool server is connected to a bitcoind node that acts as a gateway to the Bitcoin network. The node collects recent information regarding newly transmitted transactions and newly built blocks which are then used to construct a new block template. The template header is then sent via the Stratum server to the miners who attempt to complete it to a valid block. This is done by trying different values of the nonce field in the header. If the block is completed, the result is sent back to the Stratum server, which then uses the gateway node to publish the newly formed block to the network.

Multi-homing Mining pools often use multiple gateways hosted by diffrent Internet Service Providers. We refer to the number of different ISPs a pool has as its multi-homing degree.

III. Routing attacks on Bitcoin

In this section, we give an overview of the two routing attacks we describe in this paper: *(i)* partitioning the Bitcoin network (Section III-A); and *(ii)* delaying the propagation of blocks. For each attack, we briefly describe its effectiveness and challenges as well as its impact on the Bitcoin ecosystem (Section III-B).

A. Partitioning the Bitcoin Network

In this attack, an AS-level adversary seeks to isolate a set of nodes P from the rest of the network, effectively partitioning

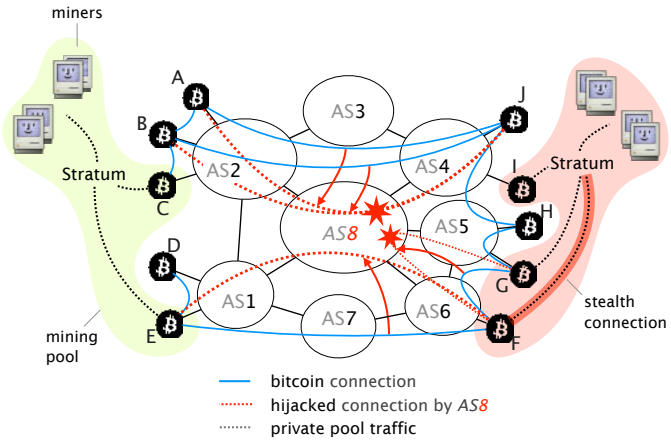


Fig. 1: Illustration of how an AS-level adversary (AS8) can intercept Bitcoin traffic by hijacking prefixes to isolate the set of nodes $P = (A, B, C, D, E, F)$.

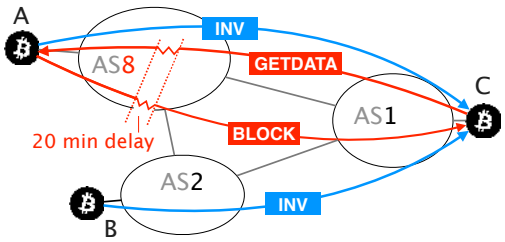


Fig. 2: Illustration of how an AS-level adversary (AS8) which naturally intercepts a part of the traffic can delay the delivery of a block for 20 minutes to a victim node (C).

the Bitcoin network into two disjoint components. The actual content of P depends on the attacker's objectives and can range from one or few merchant nodes, to a set of nodes holding a considerable percentage of the total mining power.

Attack The attacker first diverts the traffic destined to nodes in P by hijacking the most-specific prefixes hosting each of the IP address. Once on-path, the attacker intercepts the Bitcoin traffic (e.g., based on the TCP ports) and identifies whether the corresponding connections cross the partition she tries to create. If so, the attacker drops the packets. If not, meaning the connections are contained within P, she monitors the exchanged Bitcoin messages so as to detect "leakage points". Leakage points are nodes currently within P, which maintain connections with nodes outside of P, that the attacker cannot intercept, namely "stealth" connections. The attacker can detect these nodes automatically and isolate them from others in P (Section IV). Eventually, the attacker isolates the maximal set of nodes in P that can be isolated.

Example We illustrate the partition attack on the simple network in Fig. 1 that is composed of 8 ASes, some of which host Bitcoin nodes. Two mining pools are depicted as a green (left) and a red (right) region. Both pools are multi-homed and have gateways in different ASes. For instance, the red (right) pool has gateways hosted in AS4, AS5, and AS6. We denote the initial Bitcoin connections with blue lines, and those that have been diverted via hijacking with red lines. Dashed black lines represent private connections within the pools. Any AS on the path of a connection can intercept it.

Consider an attack by AS8 that is meant to isolate the set of nodes $P = (A, B, C, D, E, F)$. First, it hijacks the prefixes advertised by AS1, AS2 and AS6, as they host nodes within P, effectively attracting the traffic destined to them. Next, AS8 drops all connections crossing the partition: i.e., (A, J), (B, J) and (F, G).

Observe that node F is within the isolated set P, but is also a gateway of the red pool with which F most likely communicates. This connection may not be based on the Bitcoin protocol and thus it cannot be intercepted (at least, not easily). As such, even if the attacker drops all the Bitcoin connections she intercepts, node F may still learn about transactions and blocks produced on the other side and might leak this information within P. Isolating P as such is infeasible. However, AS8 can identify that node F is the leakage point during the attack and exclude it from P, essentially isolating $I' = (A, B, C, D, E)$ instead. This I' is actually the maximum subset of P that can be isolated from the Bitcoin network.

Practicality We extensively evaluate the practicality of isolating sets of nodes of various sizes (Section VII). We briefly summarize our findings. *First*, we performed a real BGP hijack against our own Bitcoin nodes and show that it takes less than 2 minutes for an attacker to divert Bitcoin traffic. *Second*, we estimated the number of prefixes to hijack so as to isolate nodes with a given amount of mining power. We found that hijacking only 39 prefixes is enough to isolate a specific set of nodes which accounts for almost 50% of the overall mining power. Through a longitudinal analysis spanning over 6 months, we found that much larger hijacks happen regularly and that some of them have already impacted Bitcoin traffic. *Third*, we show that, while effective, partitions do not last long after the attack stops: the two components of the partition quickly reconnect, owing to natural churn. Yet, it takes hours for the two components to be densely connected again.

Impact The impact of a partitioning attack depends on the number of isolated nodes and how much mining power they have. Isolating a few nodes essentially constitutes a denial of service attack and renders them vulnerable to 0-confirmation double spends. Disconnecting a considerable amount of mining power can lead to the creation of two different versions of the blockchain. All blocks mined on the side with the least mining power will be discarded and all included transactions are likely to be reversed. Such an attack would cause revenue loss for the miners on the side with least mining power and a prominent risk of double spends. The side with the most mining power would also suffer from an increased risk of selfish mining attacks by adversaries with mining power.

B. Slowing down the Bitcoin network

In a delay attack, the attacker's goal is to slow down the propagation of new blocks sent to a set of Bitcoin nodes without disrupting their connections. As with partitioning, the attack can be targeted, aimed at selected nodes, or network-wide, aimed at disrupting the ability of the entire network to reach consensus [20]. Unlike partitioning attacks though, an attacker can delay the overall propagation of blocks towards a node even if she intercepts a subset of its connections.

Attack Delay attacks leverage three key aspects of the Bitcoin protocol: *(i)* the asymmetry in the way Bitcoin nodes exchange blocks using INV, GETDATA, and BLOCK messages (Section II); *(ii)* the fact that these messages are not protected against tampering (unencrypted, no secure integrity checks); and *(iii)* the fact that a Bitcoin node waits for 20 minutes after having requested a block from a peer before requesting it again from another peer. These protocol features enable an attacker intercepting even one direction of the victim's connection to delay the propagation of a block, as long as this connection is traversed by either the actual BLOCK message or the corresponding GETDATA.

Specifically, if the attacker intercepts the traffic *from* the victim, she can modify the content of the GETDATA message the victim uses to ask for blocks. By preserving the message length and structure and by updating the TCP and Bitcoin checksums, the modified message is accepted by the receiver and the connection stays alive. If the attacker intercepts the traffic *towards* a node, she can instead corrupt the content of the BLOCK message such that the victim considers it invalid. In both cases, the recipient of the blocks remains uninformed for 20 minutes.

Example As an illustration, consider Fig. 2, and assume that AS8 is the attacker and C, the victim. Suppose that A and B both advertise a block (say, block X) to C via an INV message and that, without loss of generality, the message from A arrives at C first. C will then send a GETDATA message back to A requesting block X and start a 20 minute timeout count. By modifying the content of the GETDATA node A receives, AS8 indirectly controls what node A will send to node C. This way the attacker can delay the delivery of the block by up to 20 minutes while avoiding detection and disconnection. Alternatively, AS8 could modify the BLOCK message.

Practicality We verified the practicality of delay attacks by implementing an interception software which we used against our own Bitcoin nodes. We show that intercepting 50% of a node connections is enough to keep the node uninformed for 63% of its uptime (Section VIII).

We also evaluated the impact that ASes, which are naturally traversed by a lot of Bitcoin traffic, could have on the network using a scalable event-driven simulator. We found that due to the relatively high degree of multi-homing that pools employ, only very powerful coalitions of network attackers (e.g., all ASes based in the US) could perform a network-wide delay attack. Such an attack is thus unlikely to occur in practice.

Impact Similarly to partitioning attacks, the impact of a delay attack depends on the number and type (e.g., pool gateway) of impacted nodes. At the node-level, delay attacks can keep the victim eclipsed, essentially performing a denial of service attack or rendering it vulnerable to 0-confirmation double spends. If the node is a gateway of a pool, such attacks can be used to engineer block races, and waste the mining power of the pool. Network-wide attacks increase the fork rate and render the network vulnerable to other exploits. If a sufficient number of blocks are discarded, miners revenue is decreased and the network is more vulnerable to double spending. A slowdown of block transmission can be used to launch selfish mining attacks by adversaries with mining power.

IV. Partitioning Bitcoin

In this section, we elaborate on partition attacks in which an AS-level adversary seeks to isolate a set of nodes P. We first describe which partitions are feasible by defining which connections may cause information leakage (Section IV-A) to the isolated set. We then discuss how an attacker may better select a P that is feasible, if she has some view of the Bitcoin topology (Section IV-B). Next, we walk through the entire attack process, starting with the interception of Bitcoin traffic, the detection of leakage points and the adaptation of P until the partition is successfully created (Section IV-C). In particular, we present an algorithm which, given a set of nodes P, leads the attacker to isolate the maximal feasible subset. Finally, we prove that our algorithm is correct (Section IV-D).

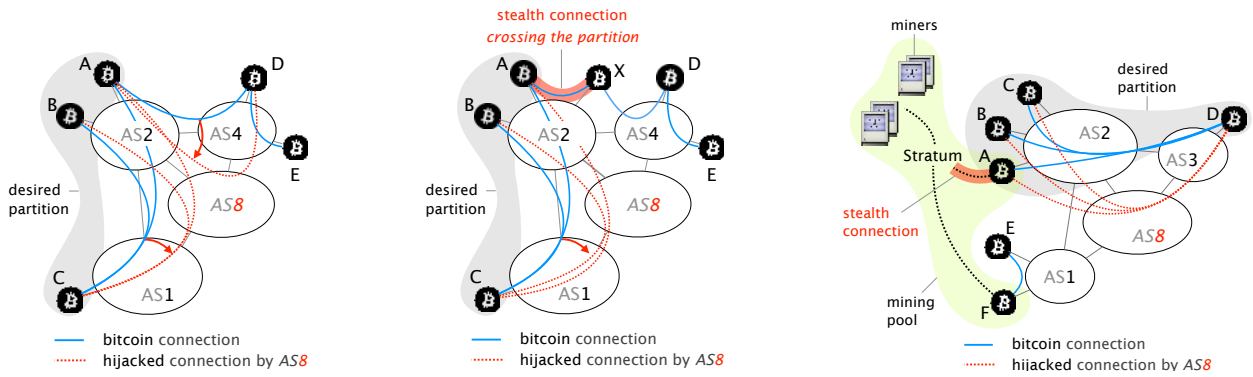
A. Characterizing feasible partitions

An attacker can isolate a set of nodes P from the network if and only if *all* connections (a, b) where $a \in P$ and $b \notin P$ can be intercepted. We refer to such connections as *vulnerable* and to connections that the attacker cannot intercept as *stealth*.

Vulnerable connections: A connection is *vulnerable* if: *(i)* an attacker can divert it via a BGP hijack; and *(ii)* it uses the Bitcoin protocol. The first requirement enables the attacker to intercept the corresponding packets, while the second enables her to identify and then drop or monitor these packets.

As an illustration, consider Fig. 3a, and assume that the attacker, AS8, wants to isolate $P = \{A, B, C\}$. By hijacking the prefixes pertaining to these nodes the attacker receives all traffic from nodes A and B to node C, as well as the traffic from node D to node A. The path the hijacked traffic follows is depicted with red dashed lines and the original path with blue lines. As all nodes communicate using the Bitcoin protocol, their connections can easily be distinguished, as we explain in Section IV-C. Here, AS8 can partition P from the rest of the network by dropping the connection from node D to node A.

Stealth connections: A connection is *stealth* if the attacker cannot intercept it. We distinguish three types of stealth connections: *(i)* intra-AS; *(ii)* intra-pool; and *(iii)* pool-to-pool.



(a) Feasible partition

(b) Infeasible partition because of intra-AS connections

(c) Infeasible partition because of intra-pool connections

Fig. 3: Not all Bitcoin connections can be diverted by an attacker implying that some partitions cannot be formed.

intra-AS: An attacker cannot intercept connections within the same AS using BGP hijack. Indeed, internal traffic does not get routed by BGP, but by internal routing protocols (e.g., OSPF, EIGRP). Thus, any intra-AS connection crossing the partition border renders the partition infeasible. Such connections represent only 1.14% of all the possible connection nodes can create (this percentage is calculated based on the topology we inferred in Section VI).

As an illustration, consider Fig. 3b and assume that the attacker, AS8, wants to isolate $P = \{A, B, C\}$. By hijacking the corresponding BGP prefixes, AS8 can intercept the connections running between nodes A and B to node C. However, she does not intercept the intra-AS connection between A and X. This means that node X will inform node A of the blocks mined in the rest of the network, and node A will then relay this information further within P. Thus, $P = \{A, B, C\}$ is not feasible. Yet, observe that isolating $I = \{B, C\}$ is possible. In the following, we explain how the attacker can detect that A maintains a stealth connection leading outside of the partition and dynamically adapt to isolate I instead.

intra-pool: Similarly to intra-AS connections, an attacker might not be able to cut connections between gateways belonging to the same mining pool. This is because mining pools might rely on proprietary or even encrypted protocols for internal communication.

As an illustration, consider Fig. 3c and assume that the attacker, AS8, wants to isolate $P = \{A, B, C, D\}$. By hijacking the corresponding prefixes, she would intercept and cut all Bitcoin connections between nodes A, B, C, D and nodes E, F. However, nodes A and F would still be connected internally as they belong to the same (green) pool. Again, observe that while isolating $P = \{A, B, C, D\}$ is not feasible, isolating $I = \{B, C, D\}$ from the rest of the network is possible.

pool-to-pool: Finally, an attacker cannot intercept (possibly encrypted) private connections, corresponding to peering agreements between mining pools. From the attacker's point of view, these connections can be treated as intra-pool connections and the corresponding pair of pools can be considered as one larger pool. Note that such connections are different than public initiatives to interconnect all pools, such as the Bitcoin relays [13]. Unlike private peering agreements, relays *cannot* act as bridges to the partition (see Appendix E).

B. Preparing for the attack

In light of these limitations the attacker can apply two techniques to avoid having stealth connections crossing the partition she creates. First, she can include in P either all or none of the nodes of an AS, to avoid intra-AS connections crossing the partition. This can be easily done as the mapping from IPs to ASNs is publicly available [11]. Second, she can include in P either all or none of the gateways of a pool, to avoid intra-pool connections crossing the partition. Doing so requires the attacker to know all the gateways of the mining pools she wants to include in P. Inferring the gateways is outside the scope of this paper, yet the attacker could use techniques described in [36] and leverage her ability to inspect the traffic of almost every node via hijacking (see Appendix C). Even with the above measures, P may still contain leakage points that the attacker will need to identify and exclude (see below). Yet, these considerations increase the chances of establishing the desired partition as well as reducing the time required to achieve it.

C. Performing the attack

We now describe how a network adversary can successfully perform a partitioning attack. The attack is composed of two main phases: *(i)* diverting relevant Bitcoin traffic; and *(ii)* enforcing the partition. In the former phase, the adversary diverts relevant Bitcoin traffic using BGP hijacking. In the latter phase, the attacker cuts all vulnerable connections that cross the partition and excludes from P nodes which are identified as leakage points. Leakage points are nodes that are connected to the rest of the network via stealth connections.

Intercept Bitcoin traffic: The attacker starts by hijacking all the prefixes pertaining to the Bitcoin nodes she wants to isolate, i.e. all the prefixes covering the IP addresses of nodes in P. As a result, she receives all the traffic destined to these prefixes, which she splits into two packet streams: relevant and

Algorithm 1: Partitioning algorithm.

Input: - P, a set of Bitcoin IP addresses to disconnect from the rest of the Bitcoin network; and
- $S = [pkt_1, \cdots]$, an infinite packet stream of diverted Bitcoin traffic resulting from the hijack of the prefixes pertaining to P.

Output: False if there is no node $\in P$ that can be verifiably isolated;

1 **enforce_partition**(P, S):
2 **begin**
3 $U \leftarrow \varnothing$;
4 $L \leftarrow \varnothing$;
5 **while** $P \setminus (L \cup U) \neq \varnothing$ **do**
6 **for** $pkt \in S$ **do**
7 **if** $pkt.ip_src \in P \wedge pkt.ip_src \notin L$ **then**
8 $last_seen[pkt.ip_dst] = now()$;
9 $U \leftarrow U \setminus \{pkt.ip_src\}$;
10 $detect_leakage(U, pkt)$;
11 **else**
12 $drop(pkt)$;
13 **for** $src \in P \wedge src \notin L$ **do**
14 **if** $last_seen[src] > now() - threshold$ **then**
15 $U \leftarrow U \cup \{src\}$
16 **return** *false*;

Algorithm 2: Leakage detection algorithm.

Input: - U, a set of Bitcoin IP addresses the attacker cannot monitor; and
- pkt, a (parsed) diverted Bitcoin packet.

1 **detect_leakage**(U, pkt):
2 **begin**
3 **if** $contains_block(pkt) \vee contains_inv(pkt)$ **then**
4 **if** $hash(pkt) \in Blocks(\neg(P \setminus L))$ **then**
5 $L \leftarrow L \cup \{pkt.ip_src\}$;
6 $drop(pkt)$;

irrelevant. Relevant traffic includes any Bitcoin traffic destined to nodes in P. This traffic should be further investigated. Irrelevant traffic corresponds to the remaining traffic which should be forwarded back to its legitimate destination.

To distinguish between relevant and irrelevant traffic, the attacker applies a simple filter matching on the IP addresses, the transport protocol and ports used, as well as certain bits of the TCP payload. Specifically, the attacker first classifies as irrelevant all non-TCP traffic as well as all traffic with destination IPs which are not included in P. In contrast, the attacker classifies as relevant all traffic with destination or source TCP port the default Bitcoin port (8333). Finally, she classifies as relevant all packets which have a Bitcoin header in the TCP payload. Any remaining traffic is considered irrelevant.

Partitioning algorithm: Next, the attacker processes the relevant traffic according to Algorithms 1 and 2. We start by presenting their goal before describing them in more details.

The high-level goal of the algorithms is to isolate as many nodes in P as possible. To do so, the algorithms identify L, the nodes that are leakage points, and disconnect them from the other nodes in P. Also, the algorithms maintain a set of verifiably isolated nodes $P' = P \setminus \{U \cup L\}$, where U corresponds to the nodes that cannot be monitored (e.g., because they never send packets). In particular, Algorithm 2 is in charge of identifying L, while Algorithm 1 is in charge of identifying U and performing the isolation itself.

We now describe how the algorithms work. Algorithm 1 starts by initializing L and U to $\varnothing$. For every received packet, the algorithm first decides whether the packet belongs to a connection internal to $P \setminus L$ or to one between a node in $P \setminus L$ and an external node based on the source IP address. If the source IP is in $P \setminus L$, the packet belongs to an internal connection and it is given to Algorithm 2 to investigate if the corresponding node acts as a leakage point (Algorithm 1, Line 10). Otherwise, the packet belongs to a connection that crosses the partition and is dropped (Algorithm 1, Line 12).

Given a packet originated from $P \setminus L$, Algorithm 2 checks whether the sender of the packet is advertising information from outside of $P \setminus L$. Particularly, the attacker checks whether the packet contains an INV message with the hash of a block mined outside of $P \setminus L$ (or the block itself). If it does so, the sender must have a path of stealth connections to a node outside of $P \setminus L$ from which the block was transmitted. Thus the sender is a leakage point and is added to L (Algorithm 2, Line 5). The actual packet is also dropped to prevent this information from spreading.

To detect whether a node in $P \setminus L$ is a leakage point, an attacker should be able to intercept at least one of that node's connections. Specifically, the node should have a vulnerable connection to another node within $P \setminus L$, so that the attacker can monitor the blocks it advertises. To keep track of the nodes that the attacker cannot monitor, Algorithm 1 maintains a set U which contains the nodes she has not received any packets from for a predefined time threshold. (Algorithm 1, Line 15). Whenever one of these nodes manages to a establish a connection that the attacker intercepts, it is removed from U (Algorithm 1, Line 9).

Example: We now show how the algorithms work on the example of Fig. 3b in which the attacker, AS8, aims to isolate $P = \{A, B, C\}$. By hijacking the prefixes corresponding to these nodes, the attacker intercepts the connections (B, C) and (A, C) and feeds the relevant packets to the algorithms. Recall that the partition is bridged by a stealth (intra-AS) connection between nodes A and X which cannot be intercepted by the attacker. When a block outside P is mined, node X will inform A which then will advertise the block to C. The attacker will catch this advertisement and will conclude that node A is a leakage point. After that, the attacker will drop the packet and will add A to L. As such, all future packets from A to other nodes within $P \setminus L = \{A, B\}$ will be dropped. Observe that the partition isolating $P \setminus L = \{B, C\}$ is indeed the maximum feasible subset of P.

D. Correctness of the partitioning algorithm

We now prove the properties of Algorithm 1.

Theorem 1. *Given P, a set of nodes to disconnect from the Bitcoin network, there exists a unique maximal subset $I \subseteq P$ that can be isolated. Given the assumption that Bitcoin nodes advertise blocks that they receive to all their peers, Algorithm 1 isolates all nodes in I, and maintains a set $P' = P \setminus \{U \cup L\} \subseteq I$ that contains all nodes in I that have a monitored connection and are thus known to be isolated.*

Proof. Consider the set of nodes $\mathcal{S} \subseteq P$ that has a path of stealth connections to some nodes not in P. Clearly, nodes in $\mathcal{S}$ cannot be isolated from the rest of the network by the attacker. Let $I = P \setminus \mathcal{S}$. Notice that I is the maximal set in P that can be disconnected by an attacker. Now, notice that every node in S is placed in sets L or U by the algorithm: if the node has a monitored connection and is caught advertising external blocks it is placed in L (Algorithm 2 Line 5). If it is not monitored then it is placed in U (Algorithm 1, Line 15).

Notice also that the entire set I is isolated from the network. If some node has no stealth connection outside, and was removed solely for the lack of monitoring, it is still having all its packets from outside of $P \setminus L$ dropped – Algorithm 1 Line 12).

□

V. Delaying Block Propagation

While partitioning attacks (Section IV) are particularly effective and can be performed by any AS, they require full control over the victim's traffic and are also highly visible. In this section, we explore *delay attacks*, which can cause relatively severe delays in block propagation, even when an attacker intercepts only one of the victim's connections, and wishes the attack to remain relatively undetectable.

In this attack, the adversary delays the delivery of a block by modifying the content of specific messages. This is possible due to the lack of encryption and of secure integrity checks of Bitcoin messages. In addition to these, the attacker leverages the fact that nodes send block requests to the first peer that advertised each block and wait 20 minutes for its delivery, before requesting it from another peer.

The first known attack leveraging this 20 minutes time-out [28] mandates the adversary to be connected to the victim and to be the first to advertise a new block. After a successful block delay, the connection is lost. In contrast, network-based delay attacks are more effective for at least three reasons: *(i)* an attacker can act on existing connections, namely she does not need to connect to the victim which is very often not possible (e.g, nodes behind a NAT); *(ii)* an attacker does not have to be informed about recently mined blocks by the victim's peers to eclipse it; and *(iii)* the connection that was used for the attack is not necessarily lost, prolonging the attack.

Particularly, the effectiveness of the delay attack depends on the direction and fraction of the victim's traffic the attacker intercepts. Intuitively, as Bitcoin clients request blocks from one peer at a time, the probability that the attacker will

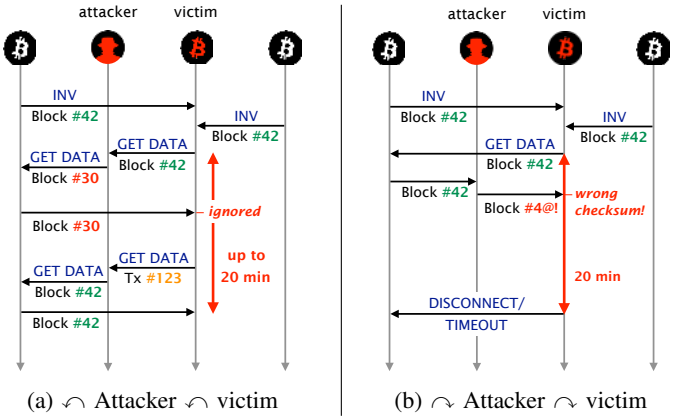


Fig. 4: An attacker can perform different delay attacks depending on the direction of traffic she intercepts. When on the path from the victim, it can modify the GETDATA message (a), while it can modify the BLOCK message when intercepting the opposite path (b).

intercept such a connection increases proportionally with the fraction of the connections she intercepts. In addition, Bitcoin connections are bi-directional TCP connections, meaning the attacker may intercept one direction (e.g., if the victim is multi-homed), both, or none at all. Depending on the direction she intercepts, the attacker fiddles with different messages. In the following, we explain the mechanism that is used to perform the attack if the attacker intercepts traffic *from* the victim (Section V-A) or *to* the victim node (Section V-B). While in both cases the attacker does delay block propagation for 20 minutes, the former attack is more effective.

A. The attacker intercepts outgoing traffic

Once a node receives a notification that a new block is available via an INV message, it issues a download request to its sender using a GETDATA message. As an illustration in Fig. 4a the victim requests Block 42 from the first of its peers that advertised the block. Since the attacker intercepts the traffic from the victim to this peer, she can modify this GETDATA message to request an older block, instead of the original one. In Fig. 4a for example, the attacker replaces the hash corresponding to block 42 with that of block 30. The advantage of doing so, over just dropping the packet, is that the length of the message is left unchanged. Notice that if the packet was dropped the attacker would need to intercept both directions of the connection and update the TCP sequence numbers of all following packets to ensure that the connection is not dropped. If the block is not delivered, the victim node will disconnect after 20 minutes. To avoid a disconnection, the attacker can use another GETDATA, sent within the 20 minute window, to perform the reverse operation. Specifically, she modifies the hash back to the original one, requested by the victim. Since the GETDATA message for blocks and transactions have the same structure, the attacker is more likely to use the latter as these are much more common. In Fig. 4a for example, she changes the hash of the transaction (Tx #123)

to the hash of block 42. Since the block is delivered within the timeout neither of the nodes disconnects or has any indication of the attack (e.g., an error in the log files).

B. The attacker intercepts incoming traffic

We now describe the mechanism an attacker would use if she intercepts traffic towards the victim, *i.e.* she can see messages received by the victim, but not the messages that it sends. This attack is less effective compared to the attack working in the opposite direction, as it will eventually result in the connection being dropped 20 minutes after the first delayed block (similarly to [28]). In this case, the attack focuses on the BLOCK messages rather than on the GETDATA. A naive attack would be for the attacker to simply drop any BLOCK message she sees. As Bitcoin relies on TCP though, doing so would quickly kill the TCP connection. A better, yet still simple approach is for the attacker to corrupt the contents of a BLOCK message while preserving the length of the packet (see Fig. 4b). This simple operation causes the BLOCK to be discarded when it reaches the victim, because of a checksum mismatch. Surprisingly though, we discovered (and verified) that the victim will *not* request the block again, be it from the same or any other peer. After the 20 minute timeout elapses, the victim simply disconnects because its requested block did not arrive on time.

An alternative for the adversary is to replace the hash of the most recent Block with a hash of an older one in all the INV messages the victim receives. This attack however would fail if the attacker intercepts only a fraction of the connections, as the victim will be informed via other connections. As such, this practice is only useful when the attacker hijacks and thus intercepts all the traffic directed to the victim.

VI. How vulnerable is Bitcoin to routing attacks? A comprehensive measurement analysis

Evaluating the impact of routing attacks requires a good understanding of the routing characteristics of the Bitcoin network. In this section, we explain the datasets and the techniques used to infer a combined Internet and Bitcoin topology (Section VI-A). We then discuss our key findings and their impact on the effectiveness of the two routing attacks we consider (Section VI-B).

A. Methodology and datasets

Our study is based on three key datasets: *(i)* the IP addresses used by Bitcoin nodes and gateways of pools; *(ii)* the portion of mining power each pool possesses; *(iii)* the forwarding path taken between any two IPs. While we collected these datasets over a period of 6 months, starting from October 2015 through March 2016, we focus on the results from a 10 day period starting from November 5th 2015, as the results of our analysis do not change much through time.

Bitcoin IPs We started by collecting the IPs of regular nodes (which host no mining power) along with the IPs of the gateways the pools use to connect to the network. We gathered this dataset by combining information collected by two Bitcoin supernodes with publicly available data regarding mining pools. One supernode was connected to ~2000 Bitcoin nodes per day, collecting block propagation information, while the other was crawling the Bitcoin network, collecting the ~6,000 IPs of active nodes each day.

We inferred which of these IPs act as the gateway of a pool in two steps. *First*, we used block propagation information (gathered by the first supernode), considering that the gateways of a pool are most likely the first to propagate the blocks this pool mines. Particularly, we assigned IPs to pools based on the timing of the INV messages received by the supernode. We considered a given IP to belong to a gateway of a pool if: *(i)* it relayed blocks of that pool more than once during the 10 day period; and *(ii)* it frequently was the first to relay a block of that pool (at least half as many times as the most frequent node for that pool). *Second*, we also considered as extra gateways the IPs of the stratum servers used by each mining pool. Indeed, previous studies [36] noted that stratum servers tend to be co-located in the same prefix as the pool's gateway. Since the URLs of the stratum servers are public (Section II), we simply resolved the DNS name (found on the pools websites or by directly connecting to them) and add the corresponding IPs to our IP-to-pool dataset.

Mining power To infer the mining power attached to pools, we tracked how many blocks each pool mined during the 10 days interval [2] and simply assigned them a proportional share of the total mining power.

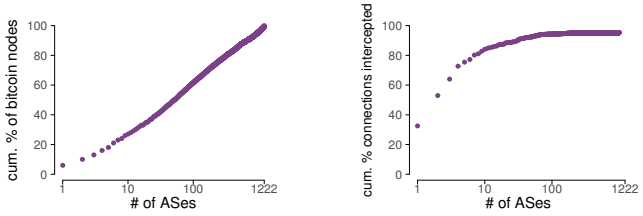
AS-level topology and forwarding paths We used the AS-level topologies provided by CAIDA [5] to infer the forwarding paths taken between any two ASes. An AS-level topology is a directed graph in which a node corresponds to an AS and a link represents an inter-domain connection between two neighboring ASes. Links are labeled with the business relationship linking the two ASes (customer, peer or provider). We computed the actual forwarding paths following the routing tree algorithm described in [30] which takes into account the business relationships between ASes.

Mapping Bitcoin nodes to ASes We finally inferred the most-specific prefix and the AS hosting each Bitcoin node by processing more than *2.5 million* BGP routes (covering all Internet prefixes) advertised on 182 BGP sessions maintained by 3 RIPE BGP collectors [10] (rrc00, rrc01 and rrc03). The mapping is done by associating each prefix to the origin AS advertising it and by validating the stability of that origin AS over time (to avoid having the mapping polluted by hijacks).

B. Findings

We now discuss the key characteristics of the Bitcoin network from the Internet routing perspective. We explain which of them constitute enablers or hindrances for an AS-level attacker.

A few ASes host most of the Bitcoin nodes Fig. 5a depicts the cumulative fraction of Bitcoin nodes as a function of the number of hosting ASes. We see that only 13 (resp. 50) ASes



(a) Only 13 ASes host 30% of the entire network, while 50 ASes host 50% of the Bitcoin network.

(b) Few ASes intercept large percentages of Bitcoin traffic: 3 of them intercept 60% of all possible Bitcoin connections.

Fig. 5: Bitcoin is heavily centralized from a routing viewpoint.

host 30% (resp. 50%) of the entire Bitcoin network. These ASes pertain to broadband providers such as Comcast (US), Verizon (US) or Chinanet (CN) as well as to cloud providers such as Hetzner (DE), OVH (FR) and Amazon (US). We observe the same kind of concentration when considering the distribution of Bitcoin nodes per IP prefix: only 63 prefixes (0.012% of the Internet) host 20% of the network.

Regarding delay attacks, this high concentration makes Bitcoin traffic more easy to intercept and therefore more vulnerable. With few ASes hosting many nodes, any AS on-path (including the host ASes) is likely to intercept many connections at once, making delay attacks more disruptive. Regarding partition attacks, the effect of the concentration is a bit more nuanced. Indeed, high concentration reduces the total number of feasible partitions because of intra-AS connections that cannot be intercepted (Section IV-A). At the same time, tough, the remaining feasible partitions are much easier to achieve since they require fewer hijacked prefixes (Section IV).

A few ASes naturally intercept the majority of the Bitcoin traffic Large transit providers (*i.e.*, Tier-1s) tend to be traversed by a large fraction of all the Bitcoin connections. Fig. 5b depicts the cumulative percentage of connections that can be intercepted by an increasing number of ASes (*e.g.*, by colluding with each other). We see that only *three* ASes, namely Hurricane Electric, Level3, and Telianet, can together intercept *more than 60% of all possible Bitcoin connections*, with Hurricane alone being on path for 32% of all connections.

Regarding delay attacks, these few ASes could act as powerful delay attackers. Regarding partition attacks, this observation does not have any direct implication as partitioning requires a *full* cut to be effective (Section IV).

>90% of Bitcoin nodes are vulnerable to BGP hijacks 93% of all prefixes hosting Bitcoin nodes are shorter than /24, making them vulnerable to a global IP hijack using more-specific announcements. Indeed, prefixes strictly longer than /24 (i.e., /25 or longer) are filtered by default by many ISPs. Observe that the remaining 7% hosted in /24s are *not* necessarily safe. These can still be hijacked by another AS performing a shortest-path attack, i.e., the attacker, who will advertise a /24 just like the victim's provider will attract traffic from all ASes that are closer to her in terms of number of hops.

While this finding does not have a direct impact on delay attacks, it clearly helps partition attackers as they can divert almost all Bitcoin traffic to their infrastructure (modulo stealth connections, see Section IV).

Mining pools tend to be distributed and multi-homed Mining pools have a complex infrastructure compared to regular nodes. We found that *all* pools use at least two ASes to connect to the Bitcoin network, while larger pools such as Antpool, F2Pools, GHash.IO, Kano use up to 5 ASes.

Pool multi-homing makes both network attacks more challenging and is one of the main precaution measures node owners can use against routing attacks. While harder, routing attacks are still possible in the presence of multi-homing as we illustrate in Section VIII.

Bitcoin routing properties are stable over time While numerous nodes continuously join and leave the Bitcoin network, the routing properties highlighted in this section are stable. As validation, we ran our analysis daily over a 4 month period. We found that the same IPs were present on average for 15.2 consecutive days (excluding IPs that were seen only once). Moreover, 50 ASes hosted each day 49.5% of Bitcoin clients (standard deviation: 1.2%) while 24.7% of Bitcoin nodes are found daily in just 100 prefixes (standard deviation: 1.77%).

VII. Partitioning Bitcoin: Evaluation

In this section, we evaluate the practicality and effectiveness of partitioning attacks by considering four different aspects of the attack. *First*, we show that diverting Bitcoin traffic using BGP hijacks works in practice by performing an actual hijack targeting our own Bitcoin nodes (Section VII-A). *Second*, we show that hijacking fewer than 100 prefixes is enough to isolate a large amount of the mining power due to Bitcoin's centralization (Section VII-B). *Third*, we show that much larger hijacks already happen in the Internet today, some already diverting Bitcoin traffic (Section VII-C). *Fourth*, we show that Bitcoin quickly recovers from a partition attack once it has stopped (Section VII-D).

A. How long does it take to divert traffic with a hijack?

We hijacked and isolated our own Bitcoin nodes which were connected to the live network via our own public IP prefixes. In the following, we describe our methodology as well as our findings with regard to the time it takes for a partition to be established.

Methodology We built our own virtual AS with full BGP connectivity using Transit Portal (TP) [46]. TP provides virtual ASes with BGP connectivity to the rest of the Internet by proxying their BGP announcements via multiple worldwide deployments, essentially acting as a multi-homed provider to the virtual AS. In our experiment, we used the `Amsterdam` TP deployment as provider and advertised 184.164.232.0/22 to it. Our virtual AS hosted six bitcoin nodes (v/Satoshi:0.13.0/). Each node had a public IP in 184.164.232.0/22 (.1 to .6 addresses) and could therefore accept connections from any other Bitcoin node in the Internet.

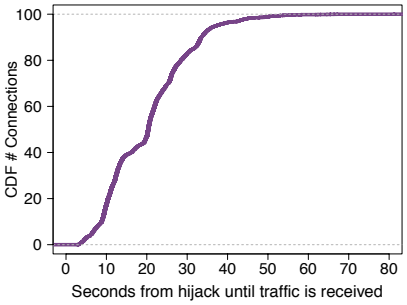


Fig. 6: Intercepting Bitcoin traffic using BGP hijack is fast and effective: all the traffic was flowing through the attacker within 90 seconds. Results computed while performing an *actual* BGP hijack against our own Bitcoin nodes.

We performed a partition attack against our 6 nodes using BGP hijacking. For this, we used `Cornell`, another TP deployment, as the malicious AS. Specifically, we advertised the prefix 184.164.235.0/24 via the `Cornell` TP. This advertisement is a more-specific prefix with respect to the announcement coming from the `Amsterdam` TP and covers all the IPs of our nodes. Thus, after the new prefix announcement is propagated throughout the Internet, Bitcoin traffic directed to any of our nodes will transit via `Cornell` instead of `Amsterdam`. To mimic an interception attack (Section II), we configured the `Cornell` TP to forward traffic back to our AS. As such, connections to our nodes stayed up during the hijack even though they experienced a higher delay.

We performed the attacks 30 times and measured the time elapsed from announcement of the most specific prefix until all traffic towards our nodes was sent via the `Cornell` TP.

Diverting Bitcoin traffic via BGP is fast (takes <2 minutes) The results of our experiment are shown in Fig. 6. The main insight is that the attacker received the hijacked traffic very quickly. After only 20 seconds, more than 50% of the connections are diverted. Within 1.5 minutes, all traffic was transiting via the malicious AS. Thus, attacked nodes are effectively isolated almost as soon as the hijack starts.

We took great care to ensure that our experiments did not negatively impact the actual Bitcoin network. We discuss the ethical considerations behind our experiments in Appendix F.

B. How many prefixes must be hijacked to isolate mining power?

Having shown that hijacking prefixes is an efficient way to divert Bitcoin traffic, we now study the practicality of isolating a specific set of nodes P. We focus on isolating sets holding mining power because they are: *(i)* more challenging to perform (as mining pools tend to be multi-homed); and *(ii)* more disruptive as successfully partitioning mining power can lead to the creation of parallel branches in the blockchain.

To that end, we first estimate the number of prefixes the attacker needs to hijack to isolate a specific set of nodes as a function of the mining power they hold. In the following subsection, we evaluate how practical a hijack of that many prefixes is with respect to the hijacks that frequently take place in the Internet.

Isolated mining power	*min. # pfxes to hijack*	*median # pfxes to hijack*	*# feasible partitions*
8%	32	70	14
30%	83	83	1
40%	37	80	8
47%	39	39	1

TABLE I: Hijacking <100 prefixes is enough to feasibly partition ~50% of the mining power. Complete table in Appendix B.

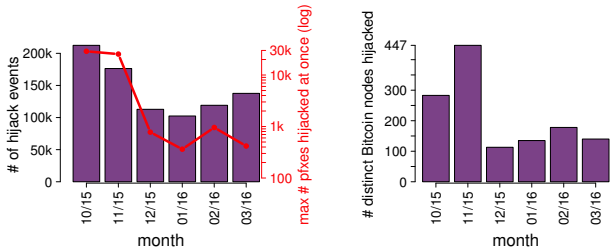
Methodology As described in Section IV, not all sets of nodes can be isolated as some connections cannot be intercepted. We therefore only determine the number of prefixes required to isolate sets P that are feasible in the topology we inferred in Section VI. In particular, we only consider the sets of nodes that contain: *(i)* either all nodes of an AS or none of them; and *(ii)* either the entire mining pool, namely all of its gateways or none of them. With these restrictions, we essentially avoid the possibility of having any leakage point within P, that is caused by an intra-AS or intra-pool stealth connection. However, we cannot account for secret peering agreements that may or may not exist between pools. Such agreements are inherently kept private and their existence is difficult to ascertain.

Hijacking <100 prefixes is enough to isolate ~50% of Bitcoin mining power In Table I we show the number of different feasible sets of nodes we found containing the same amount of mining power (4th and 1st column, respectively). We also include the minimum and median number of the prefixes the attacker would need to hijack to isolate each portion of mining power (2nd and 3rd column, respectively).

As predicted by the centralization of the Bitcoin network (Section VI), the number of prefixes an attacker needs to hijack to create a feasible partition is small: hijacking less than 100 prefixes is enough to isolate up to 47% of the mining power. As we will describe next, hijack events involving similar numbers of prefixes happen regularly in the Internet. Notice that the number of prefixes is not proportional to the isolated mining power. For example, there is a set of nodes representing 47% of mining power that can be isolated by hijacking 39 prefixes, while isolating 30% of the mining power belonging to different pools would require 83 prefixes. Indeed, an attacker can isolate additional mining power with the same number of hijacked prefixes when several pools in the isolated set are hosted in the same ASes.

C. How many hijacks happen today? Do they impact Bitcoin?

Having an estimate of the number of prefixes that need to be hijacked to partition the entire network, we now look at how common such hijacks are over a 6-months window, from October 2015 to March 2016. We show that BGP hijacks are not only prevalent, but also end up diverting Bitcoin traffic.



(a) Each month sees *at least* 100,000 hijacks, some of which involve *thousands* of prefixes.

(b) Each month, traffic for at least 100 *distinct* Bitcoin nodes end up diverted by hijacks.

Fig. 7: Routing manipulation (BGP hijacks) are prevalent today and do impact Bitcoin traffic.

Methodology We detected BGP hijacks by processing *4 billion* BGP updates advertised during this period on 182 BGP sessions maintained by 3 RIPE BGP collectors [10] (rrc00, rrc01 and rrc03). We consider an update for a prefix p as a hijack if the origin AS differs from the origin AS seen during the previous month. To avoid false positives, we do not consider prefixes which have seen multiple origin ASes during the previous month. We count only a single hijack per prefix and origin pair each day: if AS X hijacks the prefix p twice in one day, we consider both as part of a single hijack.

Large BGP hijacks are frequent in today's Internet, and already end up diverting Bitcoin traffic Fig. 7 summarizes our results. We see that there are *hundreds of thousands* of hijack events *each month* (Fig. 7a). While most of these hijacks involve a single IP prefix, large hijacks involving between 300 and 30,000 prefixes are also seen every month (right axis). Fig 7b depicts the number of Bitcoin nodes for which traffic was diverted in these hijacks. Each month, at least 100 Bitcoin nodes are victim of hijacks[2]. As an illustration, 447 distinct nodes (~7.8% of the Bitcoin network) ended up hijacked at least once in November 2015.

D. How long do the effects of a partition attack last?

Having investigated the methodology and the relative cost of creating a partition, we now explore how quickly the Bitcoin network recovers from a partition attack. We found out that while the two components quickly reconnect, they stay sparsely connected for a very long time. We first describe the experimental set-up. Next, we explain why partitions are not persistent in practice and briefly hint on how an attacker could prolong their lifetime.

Methodology We build a testbed composed of 1050 Bitcoin clients running the default `bitcoind` core (v0.12.1) in testnet mode. Each node runs in a virtual machine connected to a virtual switch and is configured with a different random IP address. Nodes automatically connect to other nodes in the testbed. We enforced a 50%–50% partition, by installing drop rules on the switch which discard any packet belonging to a connection crossing the partition. Observe that a 50%–50% split is the easiest partition to recover from, as after the attack the chance that a new connection would bridge the two halves is maximal. We measure the partition recovery time by recording the percentage of connections going from one side to the other in 30 minute intervals.

[2]The actual hijack attempt may have been aimed at other services in the same IP range, still, these nodes were affected and their traffic was re-routed.

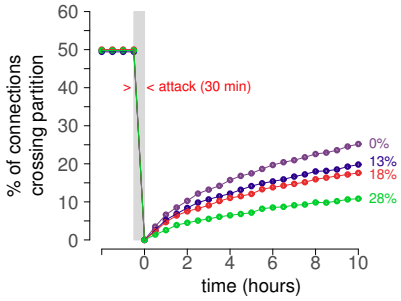


Fig. 8: Bitcoin heals slowly after large partition attacks. After 10h, only half as many connections cross the partition. Healing is even slower if the attacker is naturally on-path for 13%, 18%, 28% of the connections.

Bitcoin TCP connections are kept alive for extended periods. As such, new connections are mostly formed when nodes reconnect or leave the network (churn). To simulate churn realistically, we collected the lists of all reachable Bitcoin nodes [3], every 5 minutes, from February 10 to March 10 2016. For every node i connected in the network on the first day, we measured t_i as the elapsed time until its first disappearance. To determine the probability of a node to reboot, we randomly associated every node in our testbed with a type t_i and assumed this node reboots after a period of time determined by an exponential distribution with parameter $\lambda = \frac{1}{t_i}$. The time for next reboot is again drawn according to the same distribution. This method produces churn with statistics matching the empirical ones. We repeat each measurement at least 5 times and report the median value found.

Bitcoin quickly recovers from a partition attack, yet it takes hours for the network to be densely connected again We measured how long it takes for the partition to heal by measuring how many connections cross it, before, during and after its formation (Fig. 8). We consider two different attack scenarios: *(i)* the adversary does not intercept any bitcoin traffic before or after the attack; and *(ii)* the adversary intercepts some connections naturally.

Case 1: The adversary intercepts no traffic after the attack. It takes 2 hours until one fifth of the initial number of connections crossing the partition are established, while after 10 hours only half of the connections have been re-established. The slow recovery is due to the fact that nodes on both sides do not actively change their connections unless they or their neighbors disconnect.

Case 2: The adversary intercepts some traffic after the attack. If an AS-level adversary is naturally on-path for some of the connections, she can significantly prolong the partition's lifetime. To do so, the attacker would just continue to drop

packets on connections she naturally intercepts. We measured the effect of such attacks for attackers that are on-path for 14%, 18%, and 28% of the connections, respectively (Fig. 8). We see that an AS-adversary who is initially on-path for 28% of the connections can prolong the already slow recovery of the partition by 58%. Many other ways to increase the persistence of a partition exist. Due to space constraints, we discuss some of them in Appendix D.

Despite the long healing time, the orphan rate of the network returned to normal even with 1% of all connections crossing the partition. This fact shows that partitions need to be perfect in order to affect the network significantly.

VIII. Delaying Block Propagation: Evaluation

In this section, we evaluate the impact and practicality of delay attacks through a set of experiments both at the node-level and at the network-wide level. We start by demonstrating that delay attacks against a single node work in practice by implementing a working prototype of an interception software that we then use to delay our own Bitcoin nodes (Section VIII-A). We then evaluate the impact of network-wide delay attacks by implementing a scalable event-driven Bitcoin simulator. In constrast to partitioning attacks, and to targeted delay attacks, we show that Bitcoin is well-protected against network-wide delay attacks, even when considering large coalitions of ASes as attackers (Section VIII-B).

A. How severely can an attacker delay a single node?

Methodology We implemented a prototype of our interception software on top of Scapy [12], a Python-based packet manipulation library. Our prototype follows the methodology of Section V and is efficient both in terms of state maintained and processing time. Our implementation indeed maintains only 32B of memory (hash size) for each peer of the victim node. Regarding processing time, our implementation leverages pre-recompiled regular expressions to efficiently identify critical packets (such as those with BLOCK messages) which are then processed in parallel by dedicated threads. Observe that the primitives required for the interception software are also supported by recent programmable data-planes [19] opening up the possibility of performing the attack entirely on network devices.

We used our prototype to attack one of our own Bitcoin nodes (v/Satoshi:0.12.0/, running on Ubuntu 14.04). The prototype ran on a machine acting as a gateway to the victim node. Using this setup, we measured the effectiveness of an attacker in delaying the delivery of blocks, by varying the percentage of connections she intercepted. To that end, we measured the fraction of time during which the victim was uninformed of the most recently mined block. We considered our victim node to be uninformed when its view of the main chain is shorter than that of a reference node. The reference node was another Bitcoin client running the same software and the same number of peers as the victim, but without any attacker.

% intercepted connections	50%	80%	100%
% time victim node is uniformed	63.21%	81.38%	85.45%
% total vulnerable Bitcoin nodes	67.9%	38.9%	21.7%

TABLE II: 67.9% of Bitcoin nodes are vulnerable to an interception of 50% of their connections by an AS *other than their direct provider*. Such interception can cause the node to lag behind a reference node 63.21% of the time.

Delay attackers intercepting 50% of a node's connection can waste 63% of its mining power Table II illustrates the percentage of the victim's uptime, during which it was uniformed of the last mined block, considering that the attacker intercepts 100%, 80%, and 50% of its connections. Each value is the average over an attack period of *~200 hours*. To further evaluate the practicality of the attack, the table also depicts the fraction of Bitcoin nodes for which there is an AS, *in addition to their direct provider*, that intercepts 100%, 80%, and 50% of its connections.

Our results reflect the major strength of the attack, which is its effectiveness even when the adversary intercepts only a fraction of the victim's connections. Particularly, we see that an attacker can waste 63% of a node's mining power by intercepting half of its connections. Observe that, even when the attacker is intercepting all of the victim's connections, the victim eventually gets each block after a delay of 20 minutes.

Regarding the amount of vulnerable nodes to this attack in the Bitcoin topology, we found that, for 67.9% of the nodes, there is at least one AS *other than their provider* that intercepts more than 50% of their connections. For 21.7% of the nodes there is in fact an AS (other than their provider) that intercepts all their connections to other nodes. In short, 21.7% of the nodes can be isolated by an AS that is not even their provider.

B. Can powerful attackers delay the entire Bitcoin network?

Having shown that delay attacks against a single node are practical, we now quantify the network-wide effects of delaying block propagation.

Unlike partitioning attacks, we show that network-wide delay attacks (that do not utilize active hijacking) are unlikely to happen in practice. Indeed, only extremely powerful attackers such as a coalition grouping all ASes based in the US could significantly delay the Bitcoin network as a whole, increasing the orphan rate to more than 30% from the normal 1%. We also investigate how this effect changes as a function of the degree of multi-homing that pools adopt.

Methodology Unlike partition attacks, the impact of delay attacks on the network is difficult to ascertain. One would need to actually slow down the network to fully evaluate the cascading effect of delaying blocks. We therefore built a realistic event-driven simulator following the principles in [39] and used it to evaluate such effects.

Our simulator models the entire Bitcoin network and the impact of a delay attack considering the worst-case scenario

Coalition	*Realistic topology* (Section VI)	*Multihoming degree of pools*			
		1	*3*	*5*	*7*
US	23.78	38.46	18.18	6.29	4.20
DE	4.20	18.88	2.10	1.40	1.40
CN	4.90	34.27	1.40	0.70	0.70

TABLE III: Orphan rate (%) achieved by different network-wide level delay attacks performed by coalitions of *all* the ASes in a country, and considering either the topology inferred in Section VI or synthetic topologies with various degrees of pool multi-homing. The normal orphan rate is ~1%.

for the attacker. Specifically, it assumes that the communication between gateways of the same pool cannot be intercepted. Also, pools act as relay networks in that they propagate all blocks that they receive via all their gateways. Moreover, the simulator assumes that the delay attacker is only effective if she intercepts the traffic *from* a node that receives a block (essentially if she is able to perform the attack depicted in Fig. 4a). We provide further details on our simulator and how we evaluated it in Appendix A.

We ran our simulator on realistic topologies as well as on synthetic ones with higher or lower degrees of multi-homing. The realistic topology was inferred as described in Section VI. The synthetic ones were created by adding more gateways to the pools in the realistic topology until all pools reached the predefined degree of multi-homing.

Due to pools multi-homing, Bitcoin (as a whole) is not vulnerable to delay attackers, even powerful ones Our results are summarized in Table III. We see that multi-homed pools considerably increase the robustness of the Bitcoin network against delay attacks. In essence, multi-homed pools act as protected relays for the whole network. Indeed, multi-homed pools have better chances of learning about blocks via at least one gateway and can also more efficiently propagate them to the rest of the network via the same gateways.

If we consider the current level of multi-homing, only powerful attackers such as a coalition containing *all* US-based ASes could effectively disrupt the network by increasing the fork rate to 23% (as comparison, the baseline fork rate is 1%). In contrast, other powerful attackers such as all China-based or all Germany-based ASes can only increase the fork rate to 5%. As such coalitions are unlikely to form in practice, we conclude that *network-wide* delay attacks do not constitute a threat for Bitcoin.

Even a small degree of multi-homing is enough to protect Bitcoin from powerful attackers. If all mining pools were single-homed, large coalitions could substantially harm the currency. The US for instance, could increase the fork rate to 38% while China and Germany could increase it to 34% and 18% respectively. Yet, the fork rate drops dramatically as the average multi-homing degree increases. This is a good news for mining pools as it shows that even a small increase in their connectivity helps tremendously in protecting them against delay attacks.

IX. Countermeasures

In this section, we present a set of countermeasures against routing attacks. We start by presenting measures that do not require any protocol change and can be partially deployed in such a way that early adopters can benefit from higher protection (Section IX-A). We then describe longer-term suggestions for both detecting and preventing routing attacks (Section IX-B).

A. Short-term measures

Increase the diversity of node connections The more connected an AS is, the harder it is to attack it. We therefore encourage Bitcoin node owners to ensure they are multi-homed. Observe that even single-homed Bitcoin nodes could benefit from extra connectivity by using one or more VPN services through encrypted tunnels so that Bitcoin traffic to and from the node go through multiple and distinct ASes. Attackers that wish to deny connectivity through the tunnel would need to either know both associated IP addresses or, alternatively, disrupt all encrypted traffic to and from nodes—making the attack highly noticeable.

Select Bitcoin peers while taking routing into account Bitcoin nodes randomly establish 8 outgoing connections. While randomness is important to avoid biased decisions, Bitcoin nodes should establish a few *extra* connections taking routing into consideration. For this, nodes could either issue a `traceroute` to each of their peers and analyze how often the same AS appears in the path or, alternatively, tap into the BGP feed of their network and select their peers based on the AS-PATH. In both cases, if the same AS appears in all paths, extra random connections should be established.

Monitor round-trip time (RTT) The RTT towards hijacked destinations increases during the attack. By monitoring the RTT towards its peers, a node could detect sudden changes and establish extra random connections as a protection mechanism.

Monitor additional statistics Nodes should deploy anomaly detection mechanisms to recognize sudden changes in: the distribution of connections, the time elapsed between a request and the corresponding answer, the simultaneous disconnections of peers, and other lower-level connection anomalies. Again, anomalies should spur the establishment of extra random connections.

Embrace churn Nodes should allow the natural churn of the network to refresh their connections. A node with disabled incoming connections or even one that is behind a NAT or a firewall will never receive a random incoming connection from the rest of the network. If the node is hijacked for a few minutes and isolated from a part of the network, it will only reconnect to the other part upon reboot or when one of its outgoing connections fails.

Use gateways in different ASes While inferring the topology we noticed that many pools were using gateways in the same AS. Hosting these gateways in different ASes would make them even more robust to routing attacks.

Prefer peers hosted in the same AS and in /24 prefixes As the traffic of nodes hosted in /24 prefixes can only partially be diverted (Section II). Hosting all nodes in such prefixes would prevent partition attacks at the cost of (~1%) increase of the total number of Internet prefixes. Alternatively, nodes could connect to a peer hosted in a /24 prefix which belongs to their provider. By doing so they maintain a stealth (intra-AS) connection with a node that is at least partially protected against hijack.

B. Longer-term measures

Encrypt Bitcoin Communication and/or adopt MAC While encrypting Bitcoin connections would not prevent adversaries from dropping packets, it would prevent them from eavesdropping connections and modifying key messages. Alternatively, using a Message Authentication Code (MAC) to validate that the content of each message has not been changed would make delay attacks much more difficult.

Use distinct control and data channels A key problem of Bitcoin is that its traffic is easily identifiable by filtering on the default port (8333). Assuming encrypted connections, the two ends could negotiate a set of random TCP ports upon connecting to each other using the well-known port and use them to establish the actual TCP connection, on which they will exchange Bitcoin data. This would force the AS-level adversary to look at *all the traffic*, which would be too costly.

A simpler (but poorer) solution would be for Bitcoin clients to use randomized TCP port (encoded in clear with the ADDR message) as it will force the AS-level adversary to maintain state to keep track of these ports. Although a node can already run on a non-default port, such a node will receive fewer incoming connections (if any) as the default client strongly prefers peers that run on the default port.

Use UDP heartbeats A TCP connection between two Bitcoin nodes may take different forward and backward paths due to asymmetric routing, making AS-level adversaries more powerful (as they only need to see one direction, see Section V). In addition to TCP connections, Bitcoin clients could periodically send UDP messages with corroborating data (*e.g.*, with several recent block headers). These UDP messages can be used as a heartbeat that will allow nodes to discover that their connection was partially intercepted. As UDP messages do not rely on return traffic, this would enable node to realize that they are out-of-sync and establish new connections.

Request a block on multiple connections Bitcoin clients could ask multiple peers for pieces of the block. This measure would prevent misbehaving nodes from deliberately delaying the delivery of a block, simply because in such a case the client will only miss one fraction of the block, which it can request from one of its other peers.

X. Related Work

AS-level adversaries The concept of AS-level adversaries has been studied before in the context of Tor [25], [37], [22], [50], [53]. These works also illustrated the problems caused by centralization and routing attacks on a distributed system running atop the Internet. Yet, Tor and Bitcoin differ vastly in their behavior with one routing messages in an Onion-like fashion, while the other uses *random* connections to flood messages throughout the entire network. Although random graphs are usually robust to attacks, this paper shows that it is *not* the case when the network is centralized at the routing-level.

Bitcoin attacks The security of Bitcoin from network-based attacks has been relatively less explored compared to other attack scenarios. While eclipsing attacks [31], [27] have a similar impact than delay attacks when performed against a single node, they disrupt the victim's connections and assume that the attacker is directly connected to the victim (Section V). For more information about Bitcoin attacks, we refer the reader to a recent comprehensive survey on the Bitcoin protocol [18].

BGP security issues Measuring and detecting routing attacks has seen extensive research, both considering BGP hijacks [48], [54], [55] and interception attacks [16]. Similarly, there has been much work on proposing secure routing protocols that can prevent the above attacks [17], [29], [32], [40]. In contrast, our work is the first one to show the *consequences* of these attacks on cryptocurrencies.

XI. Conclusions

This paper presented the first analysis of the vulnerabilities of the Bitcoin system from a networking viewpoint. Based on real-world data, we showed that Bitcoin is heavily centralized. Leveraging this fact, we then demonstrated and quantified the disruptive impact that AS-level adversaries can have on the currency. We showed that attackers can partition the Bitcoin network by hijacking less than 100 prefixes. We also showed how AS-level adversaries can significantly delay the propagation of blocks while remaining undetected. Although these attacks are worrying, we also explained how to counter them with both short-term and long-term measures, some of which are easily deployable today.

Acknowledgments

We would like to thank Christian Decker for sharing Bitcoin data with us as well as for his valuable comments in the beginning of the project. We would also like to thank David Gugelmann for his support in one of our experiments. Finally, we are grateful to Tobias Bühler, Edgar Costa Molero, and Thomas Holterbach from ETH Zürich as well as Eleftherios Kokoris Kogias from EPFL for their helpful feedback on early drafts of this paper. Aviv Zohar is supported by the Israel Science Foundation (grant 616/13) and by a grant from the Hebrew University Cybersecurity Center.

References

[1] "A Next-Generation Smart Contract and Decentralized Application Platform ," https://github.com/ethereum/wiki/wiki/White-Paper.

[2] "Bitcoin Blockchain Statistics," https://blockchain.info/.

[3] "bitnodes," https://bitnodes.21.co/.

[4] "Bitnodes. Estimating the size of Bitcoin network," https://bitnodes.21.co/.

[5] "CAIDA Macroscopic Internet Topology Data Kit." https://www.caida.org/data/internet-topology-data-kit/.

[6] "Dyn Research. Pakistan hijacks YouTube." http://research.dyn.com/2008/02/pakistan-hijacks-youtube-1/.

[7] "FALCON," http://www.falcon-net.org/.

[8] "FIBRE," http://bitcoinfibre.org/.

[9] "Litecoin ," https://litecoin.org.

[10] "RIPE RIS Raw Data," https://www.ripe.net/data-tools/stats/ris/ris-raw-data.

[11] "Routeviews Prefix to AS mappings Dataset (pfx2as) for IPv4 and IPv6." https://www.caida.org/data/routing/routeviews-prefix2as.xml.

[12] "Scapy." http://www.secdev.org/projects/scapy/.

[13] "The Relay Network," http://bitcoinrelaynetwork.org/.

[14] "ZCash," https://z.cash/.

[15] A. M. Antonopoulos, "The bitcoin network," in *Mastering Bitcoin*. O'Reilly Media, Inc., 2013, ch. 6.

[16] H. Ballani, P. Francis, and X. Zhang, "A Study of Prefix Hijacking and Interception in the Internet," ser. SIGCOMM '07. New York, NY, USA: ACM, 2007, pp. 265–276.

[17] A. Boldyreva and R. Lychev, "Provable Security of S-BGP and Other Path Vector Protocols: Model, Analysis and Extensions," ser. CCS '12. New York, NY, USA: ACM, 2012, pp. 541–552.

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

[19] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese *et al.*, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 87–95, 2014.

[20] C. Decker and R. Wattenhofer, "Information propagation in the bitcoin network," in *Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on*. IEEE, 2013, pp. 1–10.

[21] ——, *Bitcoin Transaction Malleability and MtGox*. Cham: Springer International Publishing, 2014, pp. 313–326. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-11212-1_18

[22] M. Edman and P. Syverson, "As-awareness in tor path selection," in *Proceedings of the 16th ACM Conference on Computer and Communications Security*, ser. CCS '09, 2009.

[23] I. Eyal, "The miner's dilemma," in *2015 IEEE Symposium on Security and Privacy*. IEEE, 2015, pp. 89–103.

[24] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," in *Financial Cryptography and Data Security*. Springer, 2014, pp. 436–454.

[25] N. Feamster and R. Dingledine, "Location diversity in anonymity networks," in *WPES*, Washington, DC, USA, October 2004.

[26] J. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol: Analysis and applications," in *Advances in Cryptology-EUROCRYPT 2015*. Springer, 2015, pp. 281–310.

[27] A. Gervais, G. O. Karama, V. Capkun, and S. Capkun, "Is bitcoin a decentralized currency?" *IEEE security & privacy*, vol. 12, no. 3, pp. 54–60, 2014.

[28] A. Gervais, H. Ritzdorf, G. O. Karame, and S. Capkun, "Tampering with the delivery of blocks and transactions in bitcoin," in *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '15. New York, NY, USA: ACM, 2015, pp. 692–705.

[29] P. Gill, M. Schapira, and S. Goldberg, "Let the Market Drive Deployment: A Strategy for Transitioning to BGP Security," ser. SIGCOMM '11. New York, NY, USA: ACM, 2011, pp. 14–25.

[30] S. Goldberg, M. Schapira, P. Hummon, and J. Rexford, "How Secure Are Secure Interdomain Routing Protocols," in *SIGCOMM*, 2010.

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

[32] Y.-C. Hu, A. Perrig, and M. Sirbu, "SPV: Secure Path Vector Routing for Securing BGP," ser. SIGCOMM '04. New York, NY, USA: ACM, 2004, pp. 179–192.

[33] J. Karlin, S. Forrest, and J. Rexford, "Pretty Good BGP: Improving BGP by Cautiously Adopting Routes," in *Proceedings of the Proceedings of the 2006 IEEE International Conference on Network Protocols*, ser. ICNP '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 290–299.

[34] E. K. Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford, "Enhancing bitcoin security and performance with strong consistency via collective signing," in *25th USENIX Security Symposium (USENIX Security 16)*. Austin, TX: USENIX Association, 2016, pp. 279–296.

[35] J. A. Kroll, I. C. Davey, and E. W. Felten, "The economics of bitcoin mining, or bitcoin in the presence of adversaries." Citeseer.

[36] A. Miller, J. Litton, A. Pachulski, N. Gupta, D. Levin, N. Spring, and B. Bhattacharjee, "Discovering bitcoin's public topology and influential nodes."

[37] S. J. Murdoch and P. Zieliński, "Sampled traffic analysis by Internet-exchange-level adversaries," in *Privacy Enhancing Technologies: 7th International Symposium, PET 2007*, N. Borisov and P. Golle, Eds. Springer-Verlag, LNCS 4776, 2007, pp. 167–183.

[38] K. Nayak, S. Kumar, A. Miller, and E. Shi, "Stubborn mining: Generalizing selfish mining and combining with an eclipse attack," *IACR Cryptology ePrint Archive*, vol. 2015, p. 796, 2015.

[39] T. Neudecker, P. Andelfinger, and H. Hartenstein, "A simulation model for analysis of attacks on the bitcoin peer-to-peer network," in *IFIP/IEEE International Symposium on Internet Management*. IEEE, 2015, pp. 1327–1332.

[40] P. v. Oorschot, T. Wan, and E. Kranakis, "On interdomain routing security and pretty secure bgp (psbgp)," *ACM Trans. Inf. Syst. Secur.*, vol. 10, no. 3, Jul. 2007.

[41] A. Pilosov and T. Kapela, "Stealing The Internet. An Internet-Scale Man In The Middle Attack." DEFCON 16.

[42] Y. Rekhter and T. Li, *A Border Gateway Protocol 4 (BGP-4)*, IETF, Mar. 1995, rFC 1771.

[43] M. Rosenfeld, "Analysis of hashrate-based double spending," *arXiv preprint arXiv:1402.2009*, 2014.

[44] A. Sapirshtein, Y. Sompolinsky, and A. Zohar, "Optimal selfish mining strategies in bitcoin," *CoRR*, vol. abs/1507.06183, 2015.

[45] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," in *2014 IEEE Symposium on Security and Privacy*. IEEE, 2014, pp. 459–474.

[46] B. Schlinker, K. Zarifis, I. Cunha, N. Feamster, and E. Katz-Bassett, "Peering: An as for us," in *Proceedings of the 13th ACM Workshop on Hot Topics in Networks*, ser. HotNets-XIII. New York, NY, USA: ACM, 2014, pp. 18:1–18:7.

[47] J. Schnelli, "BIP 151: Peer-to-Peer Communication Encryption," Mar. 2016, https://github.com/bitcoin/bips/blob/master/bip-0151.mediawiki.

[48] X. Shi, Y. Xiang, Z. Wang, X. Yin, and J. Wu, "Detecting prefix hijackings in the Internet with Argus," ser. IMC '12. New York, NY, USA: ACM, 2012, pp. 15–28.

[49] Y. Sompolinsky and A. Zohar, "Secure high-rate transaction processing in bitcoin," in *Financial Cryptography and Data Security*. Springer, 2015, pp. 507–527.

[50] Y. Sun, A. Edmundson, L. Vanbever, O. Li, J. Rexford, M. Chiang, and P. Mittal, "RAPTOR: Routing attacks on privacy in TOR." in *USENIX Security*, 2015.

[51] A. Tonk, "Large scale BGP hijack out of India," 2015, http://www.bgpmon.net/large-scale-bgp-hijack-out-of-india/.

[52] ——, "Massive route leak causes Internet slowdown," 2015, http://www.bgpmon.net/massive-route-leak-cause-internet-slowdown/.

[53] L. Vanbever, O. Li, J. Rexford, and P. Mittal, "Anonymity on quicksand: Using BGP to compromise TOR," in *ACM HotNets*, 2014.

[54] Z. Zhang, Y. Zhang, Y. C. Hu, and Z. M. Mao, "Practical defenses against BGP prefix hijacking," ser. CoNEXT '07. New York, NY, USA: ACM, 2007.

[55] Z. Zhang, Y. Zhang, Y. C. Hu, Z. M. Mao, and R. Bush, "iSPY: Detecting IP prefix hijacking on my own," *IEEE/ACM Trans. Netw.*, vol. 18, no. 6, pp. 1815–1828, Dec. 2010.

APPENDIX

A. Bitcoin event-driven simulation

In this section, we provide more details on the simulator we used in Section VIII-B.

Inputs The simulator takes as input a realistic topology and some synthetic topologies with higher or lower degree of multi-homing. Each of the topologies includes the list of IP addresses running Bitcoin nodes, the IPs of the gateways and the hash rate associated with each mining pool, along with the forwarding paths among all pairs of IPs. The realistic topology was inferred as described in Section VI. The synthetic ones were created by adding more gateways to the pools in the realistic topology until they all reached the predefined degree of multi-homing.

Model The simulator models each Bitcoin node as an independent thread which reacts to events according to the Bitcoin protocol. Whenever a node communicates with another, the simulator adds a delay which is proportional to the number of ASes present on the forwarding path between the two nodes. Each nodes initializes 8 connections following the default client implementation. Blocks are generated at intervals drawn from an exponential distribution, with an expected rate of one block every 10 minutes. The probability that a specific pool succeeds in mining a block directly depends on its mining power. We consider that the gateways of a pool form a clique and communicate with each other with links of zero delay which an attacker cannot intercept. Concretely, this means that whenever a block is produced by a pool, it is simultaneously propagated from all the gateways of this pool. Although this choice makes the attacker less effective, we assume that this is the default behavior of a benign pool.

Attack In the simulation, an AS can effectively delay the delivery of a block between two nodes if and only if she intercepts the traffic from the potential recipient to the sender, essentially if she is able to perform the attack depicted in Fig. 4a. An adversary that intercepts the opposite direction is considered unable to delay blocks, during the simulation. Although this choice makes the attacker less effective, it avoids a dependency between the orphan rate and time. Indeed, these attackers lose their power through time, as the connections they intercept are dropped after the first effective block delay, and possibly replaced with connections that are not intercepted by the attacker.

Validation We verified our simulator by comparing the orphan rate as well as the median propagation delay computed with those of the real network. We found that both of them are within the limits of the actual Bitcoin network [20].

Methodology We evaluated the impact of delay attacks considering the three most powerful country-based coalition (US, China and Germany) as measured by the percentage of traffic all their ASes[3] see.

[3]We map an AS to a country based on the country it is registered in.

Isolated mining power	*Minimum # prefixes*	*Median # prefixes*	*# Feasible Partitions*
6%	2	86	20
7%	7	72	23
8%	32	69	14
30%	83	83	1
39%	32	51	11
40%	37	80	8
41%	44	55	3
45%	34	41	5
46%	78	78	1
47%	39	39	1

TABLE IV: This table lists all partitions that can be created based on our inferred topology. The leftmost column indicates the portion of mining power contained within the isolated set and the rightmost the number of different combinations of pools that could form it.

We run the simulation 20 times for each set of parameters and consider a new random Bitcoin topology during each run. In each run, 144 blocks are created, which is equivalent to a day's worth of block production (assuming an average creation rate of one block per 10 minutes). We evaluated the impact of delay attacks by measuring the orphan rate, different adversaries can cause.

B. Possible Partitions

Table IV shows a complete list of all the possible sets of nodes P with the mining power that can be isolated. A summary of it was included and discussed in Section VII.

C. Inferring pool gateways

We summarize here two ways hijacking can be used to reveal the gateways of pools.

1) The attacker may hijack the pool's stratum servers to discover connections that they establish and thus reveal the gateways they connect to. Connections between the pool's stratum server and its gateways are done via bitcoind's RPC access, and can be easily distinguished.
2) The attacker may hijack the relay network to discover the connections used by large mining pools. The relay network has indeed a public set of six IPs that pools connect to. By hijacking these, one may learn the IPs of various gateways. Specifically, it will not be hard to identify a pool by observing the blocks each publishes to the relay network.

D. Increase partition persistence

Our observations in Section VII-D reveal concrete ways for the attacker to make her partition lasts longer. Intuitively, the attacker needs to suppress the effect of churn in order to keep the victim nodes isolated. The following three observations help in this regard:

Observation 1: A smaller set of nodes will be easier to isolate for extended periods. The smaller the set of nodes is, the more time will pass before any isolated node restarts. Similarly, if the set of nodes is small, external nodes will connect to the isolated set with lower probability.

Observation 2: All incoming connection slots can be occupied by connections from attacker nodes. Bitcoin nodes typically limit the number of incoming connections they accept. The attacker may use several nodes to aggressively connect to the isolated nodes, and maintain these connections even after BGP routing is restored. Connections initiated by external nodes would then be rejected by the isolated nodes (as all slots of connections remain occupied).

Observation 3: Outgoing connections can be biased via a traditional eclipse attack [31]. Once an isolated node reboots, it will try to establish connections to nodes in its peer lists. These can be biased to contain attacker nodes, other isolated nodes, and IP addresses that do not actually lead to Bitcoin nodes. This is done by aggressively advertising these IPs to the victim nodes. The connections they form upon rebooting will then be biased towards those that maintain the partition.

E. Frequently Asked Questions

Can bitcoin relays bridge the partition? While Bitcoin relays [13] improve Bitcoin performance, they do not improve Bitcoin security against routing attacks. Indeed, as the IPs of the relays are publicly available, relays are also vulnerable to both hijacking and passive attacks. As such, they can neither bridge the partition, nor act as a protected relay when an actual network attack takes place. Note that routing attacks are completely different from DDoS attacks, which relays are most likely protected against. Routing attacks are much more complicated to deal with as they constitute a human-driven process depending highly on the provider of the victim rather than the victim itself. As an illustration, the mitigation of a hijack could include the provider of the attacker disconnecting her or her announcements being filtered globally, after the hijack has been detected by the provider of the victim.

Can NATed nodes bridge a partition? All nodes behind a NAT act as a simple node that doesn't accept connections. Indeed, by hijacking the corresponding public IP, the attacker receives traffic destined to all of them. In Section IV, we explained why simple nodes cannot bridge the partition, the same applies for NATed nodes.

Can Bitcoin routing alternatives solve the attacks presented in this paper? Similar to the Bitcoin relays, existing proposals could indeed speed up the transmission of blocks by tackling with natural limitation of the current Internet such as delay and packet loss. For instance, FIBRE [8] uses Forward Error Correction (FEC) and compression to speed up the block transmission. In parallel, Falcon [7], uses a technique called cut-through routing to achieve fast block transmission in the presence of increased network delays. However, neither of the aforementioned approaches address routing attacks in the form of BGP hijacks or traffic eavesdropping.

F. Ethical considerations

Although we performed both of the attacks we describe against nodes that were connected to the Bitcoin network we did not disturb their normal operations. Regarding the hijack experiment, we advertised and hijacked a prefix that was assigned to us by the Transit Portal (TP) project [46]. No other traffic was influenced by our announcements. Also, the isolated nodes were experimental nodes we ran ourselves. Actual Bitcoin nodes that happened to be connected to these were not affected, they just had one of their connections occupied (as if they were connected to a supernode). Regarding the delay experiment, the victims were again our own Bitcoin clients. While they were indeed connected to nodes in the real network, those were not affected as we only modified packets towards our victim nodes.