An evaluation of the security of the Bitcoin Peer-to- Peer Network

James Tapsell, Raja Naeem Akram, and Konstantinos Markantonakis

ISG-SCC, Royal Holloway, University of London, Egham, United Kingdom

Email: {James.Tapsell.2015}@live.rhul.ac.uk, {r.n.akram, k.markantonakis}@rhul.ac.uk

Abstract—**Bitcoin is a decentralised digital currency that relies on cryptography rather than trusted third parties such as central banks for its security [1]. Underpinning the operation of the currency is a peer-to-peer (P2P) network that facilitates the execution of transactions by end users, as well as the transaction confirmation process known as bitcoin mining. The security of this P2P network is vital for the currency to function and subversion of the underlying network can lead to attacks on bitcoin users including theft of bitcoins, manipulation of the mining process and denial of service (DoS). As part of this paper the network protocol and bitcoin core software are analysed, with three bitcoin message exchanges (the connection handshake, GETHEADERS/HEADERS and MEMPOOL/INV) found to be potentially vulnerable to spoofing and use in distributed denial of service (DDoS) attacks. Possible solutions to the identified weaknesses and vulnerabilities are evaluated, such as the introduction of random nonces into network messages exchanges.**

I. INTRODUCTION

Bitcoin operates as a currency through a peer-to-peer (P2P) network of nodes that execute, communicate and confirm transactions. Trust in its security is maintained by both the cryptographic elements of the system and the correct functioning of the P2P network.

A. Contributions

The key contributions of this paper are:

1) Analysis of two message exchanges in the bitcoin P2P network protocol (GETHEADERS and MEMPOOL) and their potential for spoofing and abuse in denial of service attacks.
2) Analysis of the security of hardcoded DNS seed addresses that allow nodes to first connect to the network.
3) Proposal of potential improvements to the security of the bitcoin P2P network protocol.

II. BITCOIN PEER-TO-PEER NETWORK

A. Objectives & challenges

The overall goal of bitcoin, as put forward by Nakamoto, was to enable two entities to execute a transaction without relying on a trusted third party [1]. The decentralised transmission of data (i.e. blocks and transactions) in bitcoin is carried out via a distributed peer-to-peer (P2P) network [2]

To summarise the objectives of the bitcoin network:

A key challenge with achieving these objectives is that nodes do not trust each other. Nodes must have the capability to verify information themselves without relying on a trusted third party.

TABLE I
THE OBJECTIVES OF THE BITCOIN NETWORK

Objective	Achieved by
Data transmission must be decentralised [2]	A distributed peer-to-peer network
Data storage must be decentralised [2]	A full copy of the blockchain is stored and maintained by all nodes
All blocks must be accessible to all users [1]	Blocks are broadcast to all nodes
All transactions must be accessible to all users [1]	Transactions are broadcast to all nodes

B. Network architecture

The distributed P2P network is created in a dynamic way by users of bitcoin currency [2].

Network nodes are homogeneous, with no specialised coordinating nodes and each node keeps a complete copy of the blockchain. This allows nodes to verify the validity of transactions and blocks independently without trusting each other [3].

Bitcoin nodes are identified by their IP address and operate over TCP (Transmission Control Protocol) [4], which provides a reliable channel for bitcoin messages to be transmitted between nodes (i.e. guaranteed in-order delivery and recovery from transmission errors).

However, there are no further security services beyond those provided by TCP, so bitcoin messages do not have cryptographic entity authentication or integrity protection.

Limitations of TCP: TCP is a well-established and widely studied protocol and it has been known since 1989 [5] that without any other cryptographic protection, it is trivial for an on path attacker (i.e. one that is situated on the communication path) to eavesdrop, modify, replay and fabricate TCP network packets.

An off path attacker (i.e. one that is not situated on the communication path) in a TCP/IP network can manipulate routing information to position themselves on the communications path and become an on path attacker (for example by manipulating the Routing Information Protocol [5] or Border Gateway Protocol [6]).

Furthermore off path attackers are still able to fabricate TCP packets by exploiting vulnerabilities in the use and selection of TCP sequence numbers [7], through various side channel attacks [8] [9] [10] and vulnerabilities in network implementations [11] [12].

In summary this means that, without any additional security mechanisms, an on path attacker can eavesdrop, modify,

replay and fabricate TCP messages. Also if the TCP sequence number can be compromised [5] [7] [9] [12] [9] [13], an off path attacker can fabricate messages and impersonate another network node.

As well as broadcasting transactions and blocks within TCP packets, bitcoin nodes also send TCP packets containing command messages. These command messages are used to establish and maintain connections between nodes and transfer data.

These messages are open to manipulation by network attackers as they are simply passed over TCP and do not provide any additional cryptographic protection.

1) Initial connection: To join the network for the first time a node discovers other nodes through DNS queries. The DNS names of several seed servers are hardcoded into the bitcoin core client software in the chainparams.cpp file [14].

This provides a mechanism for nodes to connect to at least one peer, which will then provide them with further active peers to connect to. In this way the hardcoded DNS seed addresses act as the trusted, authoritative source for initial peers. After that, as the node interacts on the network it builds up a local database of active peers.

The hardcoded DNS seed addresses are owned and managed by volunteers and have been chosen by the Bitcoin developers. Each query returns multiple IP addresses which correspond to bitcoin nodes that have high uptime. How these nodes are chosen and who manages them is not documented and raises a number of security concerns.

Were these DNS seeds to be compromised, an attacker could for example supply the addresses of their own malicious nodes to new nodes joining the network. Similar to an eclipse attack [15], this would allow an attacker to supply the victim nodes with other malicious nodes to connect to and monopolise their network connections. In this way the DNS seeds would provide an additional vector for conducting an eclipse attack.

For example, an attacker could ensure that the addresses of malicious nodes are returned from the DNS seeds by:

- Exploiting DNS protocol weaknesses such DNS cache poisoning [16] [17] to return the IP addresses of attacker controlled nodes.
- Compromising a DNS hosting account (e.g. by phishing) and changing DNS records to return the IP addresses of attacker controlled nodes.

2) Data origin authentication of DNS seeds: The aim of a DNS protocol level attack against a bitcoin DNS seed, such as a DNS cache poisoning attack, is to return a DNS response to the victim that points to nodes of the attackers choosing. The key security services that protect against this are:

1) data integrity: to detect whether data has been modified in transit [18].
2) data origin authentication: to confirm whether data came from a genuine sender [18].

The principle method for applying data integrity and data origin authentication to DNS queries is the use of DNSSEC [19]. DNSSEC is a suite of specifications to extend the DNS protocol to allow DNS records and responses to be digitally signed by the owner of the domain.

At the time of writing (August 2017) there are currently six seed addresses listed in the bitcoin software [14]:

- seed.bitcoin.sipa.be
- dnsseed.bluematt.me
- dnsseed.bitcoin.dashjr.org
- seed.bitcoinstats.com
- seed.bitcoin.jonasschnelli.ch
- seed.btc.petertodd.org

None of these addresses has DNSSEC configured and are therefore open to DNS protocol attacks such as DNS cache poisoning.

3) Control of DNS seed addresses: It is not immediately obvious to users who controls the domain names associated with the DNS seeds. Also the mechanism that chooses the nodes that each seed will return is also not documented.

A brief analysis of public WHOIS records and bitcoin code repositories was carried out to determine the likely owners of the seed domains [20]

Five out of the six domains are controlled by the primary bitcoin developers, who between them account for the vast majority of code contributions to the core bitcoin software [20]. One is an academic author and researcher of bitcoin and has contributed to several papers referenced by this paper [3] [21] [22].

Control of the DNS seeds appears to rest with six individuals, rather than a set of companies or institutions. This is perhaps a symptom of the decentralised, anarchic ideals evident in Nakamoto's original paper [1], however whilst the source code of bitcoin is freely auditable, the operation of the DNS seed addresses and P2P network is not.

There are several security concerns, not specifically aimed at the current owners of the DNS seeds, but the principle of individuals controlling DNS seed domains.

Firstly, that any individual may be influenced or may exploit their position of trust for personal gain. For example four of the six individuals are employed by Blockstream.com [23] [24] [25] [26], a private company developing blockchain based software and services.

Secondly there is a limit to the level of security any one individual can provide against a targeted attack from a highly resourced and motivated attacker. For example an attacker that compromised the DNS server or domain hosting account associated with a DNS seed could redirect all new nodes joining the network to their own malicious nodes.

Whether control of the bitcoin seed addresses places too much power in the hands of six bitcoin developers, is a question that poses both technical and philosophical considerations for bitcoin users. Much more broadly, it raises the question of who should have control of components of the bitcoin infrastructure that cannot be decentralised (such as DNS seeds) and how should those people or organisations be held accountable.

4) Establishing connections: Once a node (e.g. Node A) has learnt the IP address of another node (Node B) a connection

is established by sending a VERSION message [27] to Node B containing its software version number.

If Node B is accepting connections from this particular software version it will reply with a VERACK message [27], which includes its own software version number.

Node A will send its own VERACK message if it also is accepting connections from this software version.

This exchange allows both nodes to check each other's software versions before deciding to establish connections. Although it is not currently, this could provide a mechanism for node operators to exclude outdated software versions (with known security vulnerabilities) from participating in the network.

5) Discovering nodes: Once a node establishes a connection with another node (a peer), the node will query it for a list of network nodes that it is aware of by sending a GETADDR message [27]. The peer will reply with an ADDR message [27] containing a list of up to 1000 peers, randomly selected from the list of active peers that it is aware of.

Several academic studies [28] [29] [2] [3] and projects [30] have demonstrated that by sending GETADDR messages to each node, and subsequently sending GETADDR messages to every new node that is reported in ADDR messages, it is possible to discover all nodes currently active on the network.

This can be used to analyse the size and geographical distribution of the network [2] as well as the environments that nodes are running in, such as cloud hosting providers, private datacentres or residential connections.

In the context of security, as a node is identified by its IP address, this can in some circumstances be probabilistically linked to a physical location [31], [32]. On a broader scale, as nodes will freely report their software version in protocol handshakes (section 3.2.4), it is possible to scan the entire network for nodes running bitcoin software versions with known vulnerabilities. For example using the publicly available bitnodes.21.co project [33], at the time of writing (August 2017) there are 3 nodes running bitcoin core version 0.8.3, which is vulnerable to remote denial of service vulnerability CVE-2013-5700 [34].
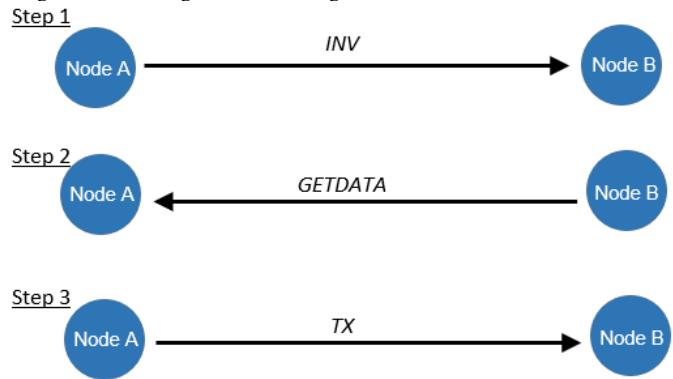
6) Transaction and block transmission: The transmission of transaction and block information to all nodes is achieved with a broadcast mechanism. Once a node learns of a new transaction or block it is forwarded on to all its neighbours (the peers the node is actively connected to). These neighbours then forward the new transaction or block on to their neighbours and the process repeats until all reachable nodes in the network have received the new transaction or block.

Node A advertises the new transaction by sending an INV message [27], which includes a SHA256 hash of the new transaction (TXID) to Node B (see figure 3.4).

If Node B is not aware of this new transaction ID (TXID) it will send a GETDATA message [27], which includes the TXID of the new transaction, to Node A.

Node A will respond by sending a TX message [27] containing the full transaction record to Node B.

Fig. 1. Advertising and transmitting a transaction between bitcoin nodes

Once Node B has successfully received the new transaction, it will validate it using its local copy of the blockchain and send INV messages to its neighbours to repeat the process.

The above process is the same for the transmission of blocks, except that the ID of the new block is sent in steps 1 and 2 (INV and GETDATA messages) and a BLOCK message [27] is used to send the block information in step 3.

The example given assumes that only one transaction or block is being broadcast, when in reality INV and GETDATA messages can contain up to 50,000 transaction or block IDs.

7) Requesting the latest blocks: Whilst a node is online and connected to the network it will receive blocks as they are broadcast across the network and the node will keep its local copy of the blockchain up to date as new blocks arrive. However whilst a node is offline new blocks will have been created which the offline node will not be aware of, so upon reconnecting to the network the node will need to obtain the missing blocks.

It will do this by sending a GETHEADERS message [27] to a node that it is connected to, which includes the block ID of the last block that the node is aware of.

For example (figure 2.3), Node A has just reconnected to Node B after being offline for some period of time. Node A will send a GETHEADERS message to Node B with a block ID of **#123**, which represents the last block in its local copy of the blockchain.

When Node B receives the GETHEADERS message it will compare the block ID **#123** with its local copy of the blockchain and if necessary reply with a HEADERS message [27], which contains the block ID's of the remaining blocks in the chain. In this example it will include block ids **#124** and **#125** in its HEADERS message.

Node A is now aware that there are two blocks, **#124** and **#125**, that it is missing.

Up to 2000 block IDs may be returned in a HEADERS message. If Node A needed to obtain more than 2000 blocks it would need to send out additional GETHEADERS messages.

The GETHEADERS/HEADERS exchange is designed to allow Node A to discover what blocks it is missing. The HEADERS message returned by Node B only contains block IDs, it does not contain the full block information. It is up

Fig. 2. Querying a peer for the latest blocks

Fig. 3. Retrieving a list of unconfirmed transactions
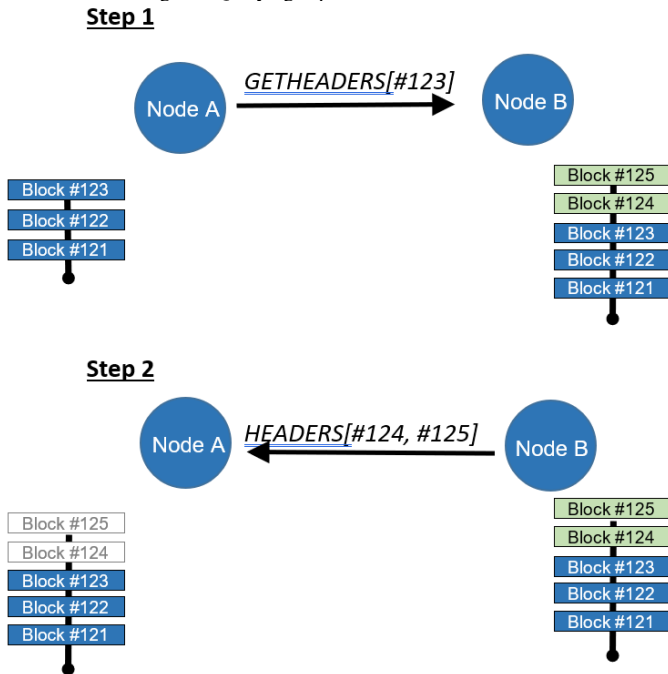
to Node A to request each block with a series of GETDATA messages (see 2.2.6) to Node B, which will then send the full information for each block.

8) Collecting unconfirmed transactions: If a node is engaged in bitcoin mining activities it will need to collect and store unconfirmed transactions to include them in the block it is mining.

The collective pool of unconfirmed transactions within the network is called the 'Mempool' and miners will typically seek to gather as large a proportion of unconfirmed transactions as possible, in order to earn the most transaction fees once they discover a block.
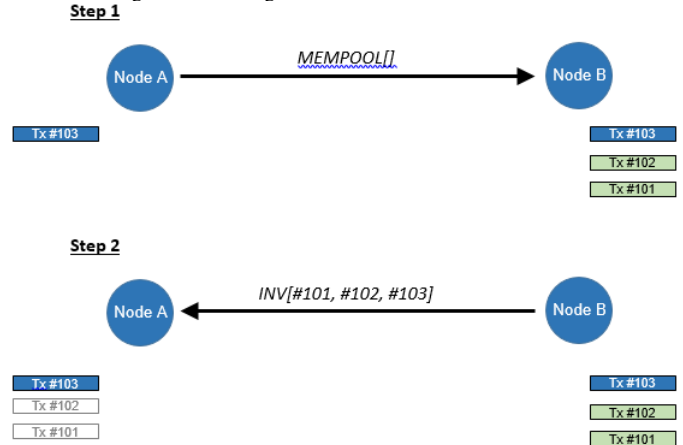
Nodes that have reconnected to the network after being offline have a method of gathering unconfirmed transactions from peers that they are connected to.

For example (figure 2.4), Node A has just reconnected to Node B after being offline for some period of time. Node A will send a MEMPOOL message [27] to Node B to request a list of unconfirmed transactions that Node B is aware of. Node B will reply with an INV message containing the transaction IDs of all unconfirmed transactions that it is aware of.

Note that Node B does not know what unconfirmed transactions Node A is already aware of, so Node B will simply send an INV message containing all the unconfirmed transactions that it is aware of.

Once Node A has learnt of the outstanding transactions (**#101** and **#102**), it will then download the full transaction information from Node B using the GETDATA/TX message exchange.

III. NETWORK BASED VULNERABILITIES AND ATTACKS

A distributed denial of service attack (DDoS) seeks to overwhelm a victim with more network traffic than the victim's network connection or computing resources can cope with [35].

One notable example of a DDoS attack is a DNS amplification attack [36]. Which in 2013 was used to generate 75Gbps of malicious traffic [37] as part of a DDoS against Spamhaus [38], an organisation that combats email spam. At the time it was the largest DDoS attack ever seen.

The two properties that make this attack successful are

1) Reflection: Responses are sent in reply to any incoming request, without authentication of the source [39].
2) Amplification: The response is much larger than the initial request, meaning that the victim receives much more data than the attacker sends out [39].

A. Possible DDoS attacks

Recall from section 2.2 that the bitcoin network protocol does not have any data origin authentication, but instead relies on the TCP sequence number to ensure that messages cannot be spoofed. However as mentioned there are numerous examples [8] [9] [10] [12] [13] where TCP sequence numbers do not provide adequate protection against spoofing.

Analysis was conducted on the bitcoin network protocol specification [40] and the bitcoin core source code [41] for message exchanges that display the potential for reflection and amplification. Such properties would indicate the potential for their exploitation in DDoS attacks against bitcoin nodes.

1) GETHEADERS/HEADERS: Recall from section 2.2.7 that upon re-joining the network a node will seek to update its local copy of the blockchain by asking its peers for the blocks that have been created whilst it was away. It will send a GETHEADERS message with the block ID of the last block it is aware of and receive back a HEADERS message containing up to 2000 block IDs.

From examining the protocol specification [27] and source code for processing an incoming GETHEADERS message [42] the standard response is to issue a HEADERS message to the listed source IP address. There are no data origin authentication

or freshness checks included in the protocol specification or any method in the processing of a GETHEADERS message to determine whether the request is genuine.

Therefore it would imply that if an attacker was able to overcome the TCP sequence number, then a forged GETHEADERS message would result in a HEADERS message being sent to the victim.

Fig. 4. GETHEADERS Reflection Attack
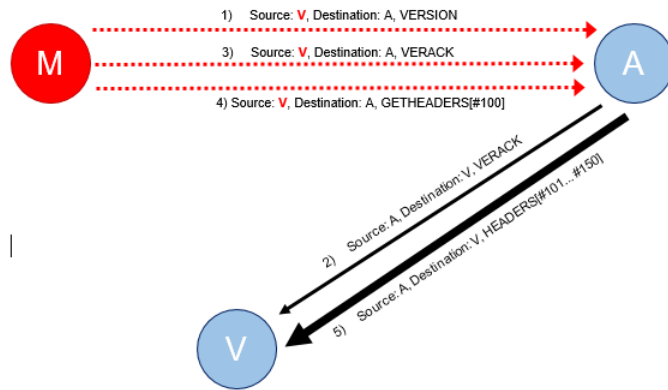
For example (figure 3.1) M wants to induce A to send a HEADERS message to victim V.

For A to accept the spoofed GETHEADERS packet, M must first trick A into establishing a connection with V. M sends a VERSION message to A with the source address spoofed as V. A sends a VERACK message to V. M sends a VERACK message to A with the source address spoofed as V.

M has now established a connection with A on behalf of V and can now commence the DOS attack. M sends a GETHEADERS message to A, with the source address spoofed as V, asking for blocks with an ID greater than **#100**. A sends a HEADERS message to V with the block IDs for blocks **#101** to **#150**.

As well the potential for reflection, the GETHEADERS/HEADERS message exchange also has the potential for amplification.

The attacker must send the following messages: 1 x VERSION message with a size of 85 bytes [27]. 1 x VERACK message with a size of 24 bytes [27]. 1 x GETHEADERS message with a size of 69 bytes [27].

A total of 168 bytes.

Recall from section 3.2.9 that the resulting HEADERS message sent to the victim can contain up to 2000 block IDs, yielding a maximum message size of approximately 162,000 bytes or 158 Kilobytes. This is an increase by a factor of approximately 964.

2) MEMPOOL: Recall from section 2.2.8 that the any node can query another peer on the network for a list of all the unconfirmed transactions that it is aware of.

This is done by sending a peer a MEMPOOL message and in reply they will receive back an INV message containing a list of the transaction IDs (txID) of transactions that are unconfirmed. Recall from section 2.2.6 that an INV message can contain as many as 50,000 transaction IDs.

Again from examining the protocol specification [27] and source code for processing an incoming MEMPOOL message [43] the standard response is to issue an INV message to the listed source IP address. There are no data origin authentication or freshness checks included in the protocol specification or any method in the processing of a MEMPOOL message to determine whether the request is genuine.

Therefore it would imply that if an attacker was able to overcome the TCP sequence number, then a forged MEMPOOL message would result in an INV message being sent to the victim.

Fig. 5. MEMPOOL Reflection attack
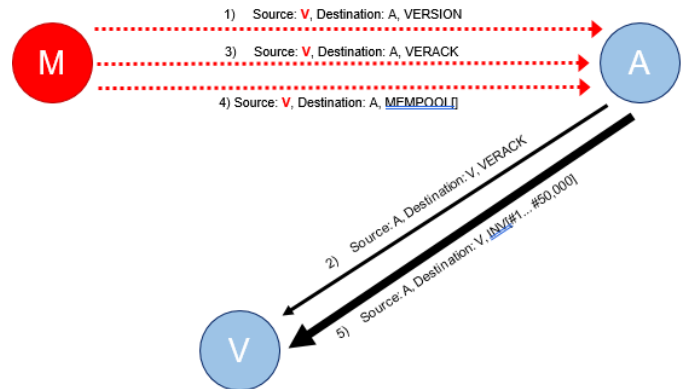
For example (figure 3.2) M wants to induce A to send an INV message to victim V.

In order for A to accept the spoofed MEMPOOL packet, M must first trick A into establishing a connection with V (section 3.2.4).

1) M sends a VERSION message to A with the source address spoofed as V.
2) A sends a VERACK message to V.
3) M sends a VERACK message to A with the source address spoofed as V.
   M has now established a connection with A on behalf of V and can now commence the DOS attack.
4) M sends a MEMPOOL message to A, with the source address spoofed as V, asking for a list of all unconfirmed transactions that A is aware of.
5) A sends an INV message to V with the txIDs for transactions **#1** to **#50,000**.

As well the potential for reflection, the MEMPOOL/INV message exchange also has the potential for amplification.

The attacker must send the following messages:
- 1 x VERSION message with a size of 85 bytes [27].
- 1 x VERACK message with a size of 24 bytes [27].
- 1 x MEMPOOL message with a size of 24 bytes [27]

A total of 133 bytes.

The size of the resulting INV message sent to the victim will vary, as although an INV message can contain up to 50,000 txIDs [27] the actual number sent will depend on the number of unconfirmed transactions that node A is aware of at the time.

The number of unconfirmed transactions in the Mempool varies quite dramatically from hour to hour, for example on 28/6/17 [44] peaking at 28,758 and dropping to 6,607 within 4 hours.

An attacker could track the size of the Mempool and time their attack to coincide with periods of high numbers of unconfirmed transactions, or indeed generate large amounts of unconfirmed transactions themselves. Therefore it is possible that an INV message may contain 50,000 transactions.

In evaluating the potential for amplification in this attack it is reasonable to assume that the resulting INV message sent to the victim will contain 50,000 txIDs. In which case, the size of the INV message will be approximately 1,800,000 bytes or 1.7 Megabytes. This is an increase by a factor of approximately 13,534.

The precise amount of data that the attacker sends and the victim receives will vary, as each message will be encapsulated in an ethernet frame (20 bytes [45]), an IP packet (36 bytes [46]) and a TCP segment (20 bytes [47]), so a total of 145 bytes is added to the size of each message.

However the potential for reflection and the scale of the amplification leads to the conclusion that the GETHEADERS and MEMPOOL message exchanges have the potential for exploitation in a denial of service attack. This hypothesis should be confirmed through experimentation

B. Comparison to other amplification vectors

While a factor of 13,534 is quite powerful, there are many other methods that can be used to amplify the size of a DDOS attack. One example is using Memcached, which has achieved a factor of 51,200 [48] in practical attacks. This also has the additional benefit that it can be created on demand, as an attacker can place large objects on the server without paying transaction fees, rather than the waiting an attacker would need to perform the same attack using bitcoin if they didn't want to pay. Memcached was used in the attack against OVH [49] which managed to hit 1.7tbps, which was helped by the number of open servers which were used.

The fact that there are other amplification vectors available does not mean that the Bitcoin ones should not be fixed though, firstly because when the other vectors are fixed the bitcoin ones will become a reasonable vector to use. There is also the fact that memcached traffic crossing a large network should be quite rare, as the overhead would be very high, making detection easier, whereas the same attack with bitcoin would raise less suspicion from the network of a single amplifier.

IV. POSSIBLE SOLUTIONS

As explained in section 2.2.4, it is trivial to discover the versions of node software used across the network and to identify nodes running software with known vulnerabilities.

Currently the node software will still establish connections with nodes running outdated versions, however the connection handshake could provide a mechanism to exclude these nodes from the network until they are updated. A minimum version number could be used as a criteria for accepting a connection

thereby enforcing a minimum version necessary to participate in the network.

Recall from section 3.2 that there is the potential for two message exchanges (GETHEADERS/HEADERS and MEMPOOL/INV) to be used to induce a target node to send data to a victim node, potentially leading to a DDoS attack.

This is because once the TCP sequence number is defeated, the bitcoin connection handshake (figure 3.3) can be spoofed which would in turn allow the GETHEADERS or MEMPOOL messages to be spoofed. Preventing spoofing of the handshake would provide protection against subsequent message exchanges from being abused.

One method of preventing an off-path attacker from spoofing the connection handshake would be to add a random nonce to the VERACK message.

Fig. 6. Bitcoin connection handshake with a random nonce added to the VERACK messages
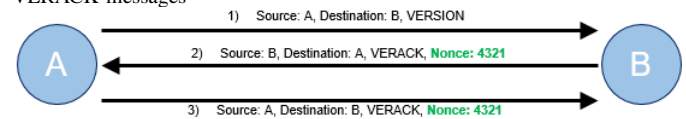
For example (figure 4.1), Node A attempts to establish a connection with Node B:
1) Node A sends a VERSION message to Node B.
2) Node B responds to Node A with a VERACK message, which includes a randomly generated nonce.
3) Node A completes the handshake by sending its own VERACK message back to Node A, including the nonce it received from Node B.
4) Node B checks that the nonce received in step 3 is correct and if so it will accept and process future messages from Node A. Otherwise Node B will ignore messages from Node A until a correct handshake is completed.

Fig. 7. An attempted spoof connection handshake with a random nonce
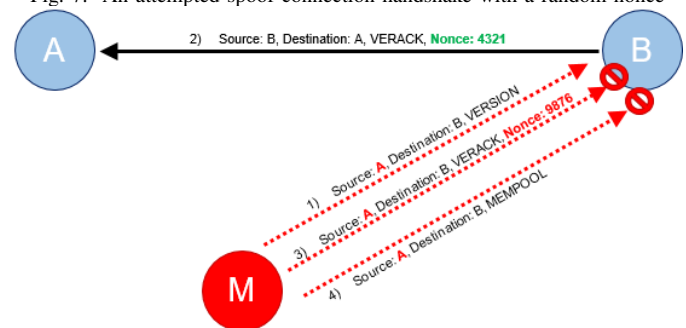
In figure 4.2 malicious Mode M tries to spoof a connection attempt from Node A with Node B.
1) Node M sends a VERSION message to Node B, changing the source address to appear that it came from Node A.
2) Node B responds to Node A with a VERACK message, which includes a randomly generated nonce. As this VERACK message is routed to Node A, Node M will not learn the value of the nonce.

3) Node M attempts to complete the handshake by sending a VERACK message to Node B, but it is forced to guess the value of the nonce generated in Step 2. Assuming that the nonce value is chosen in a secure way (explained below) it is very unlikely that the correct nonce value will be guessed.
4) Any subsequent messages received by Node B with a source address of Node A (such as a spoofed MEMPOOL message) will be rejected until a correct handshake is completed.

To ensure that a nonce value cannot feasibly be guessed, it should:

- Be randomly generated using a suitable pseudorandom number generator.
- Be sufficiently large to make brute force guessing infeasible, for example at least 32-bits (the size of the TCP sequence number [47]).

In order to detect attempted abuses of the bitcoin connection handshake the node software should also log and report incomplete connection handshakes. For example if incorrect nonces are repeatedly being given by a node this might indicate a spoof connection attempt.

V. Conclusion

To operate as a currency Bitcoin requires a P2P network to function. The operation of this P2P network is not well documented or widely studied and yet plays a crucial role in maintaining the overall security of the currency.

Users of Bitcoin rely solely on cryptography to establish trust, not on the authority of a trusted third party but on mechanisms that users can use to validate transactions themselves. Therefore this principle should also apply to communications on the P2P network – the actions and messages of other nodes should not be trusted and should be assumed to be malicious. Translating this principle into the design of bitcoin's network protocols and software does not appear to have happened.

In conclusion, the key contribution of this project are the following recommendations:

1) Protocol hardening: The bitcoin network protocol and core software implementation requires a thorough security audit, to address potential security vulnerabilities identified in the HEADERS and MEMPOOL message exchanges. The use of network security mechanisms such as random nonces, cryptographic integrity protection and entity authentication should be further considered. For example the introduction of a random nonce during the connection handshake could provide protection against message spoofing.
2) Management of network infrastructure: Parts of bitcoin's network architecture are not decentralised (e.g. DNS seed addresses) and remain under the control of individual users (section 2.2.3). This holds the potential for conflicts of interest or insider attacks and bitcoin users need to consider how components of the bitcoin infrastructure that cannot be decentralised are governed and managed in an accountable and transparent way.

A. Further work

Several areas where further work could be carried out have been identified.

Firstly the potential for bitcoin message spoofing should be investigated through experimental analysis. In particular the potential for the use of HEADERS and MEMPOOL messages in DDoS attacks should be evaluated in a lab environment as well as the live bitcoin network.

Secondly, mechanisms for providing data integrity and entity authentication for all network nodes should be incorporated into the network protocol. The current proposals [50] only provide these services for a sub set of nodes that coordinate between themselves and cannot be used at scale across the whole network.

There is also a need to harden Bitcoin against future attacks, such as quantum ones, which may become a problem around 2027 [51]. This is the point at which quantum computes will be able to be fast enough to challenge the current generation of ASIC based miners.

References

[1] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system," http://bitcoin.org/bitcoin.pdf, 2008.
[2] Joan Antoni Donet Donet, Cristina Pérez-Solà, and Jordi Herrera-Joancomartí. The bitcoin p2p network. In Rainer Böhme, Michael Brenner, Tyler Moore, and Matthew Smith, editors, *Financial Cryptography and Data Security*, pages 87–102, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.
[3] C. Decker and R. Wattenhofer. Information propagation in the bitcoin network. In *IEEE P2P 2013 Proceedings*, pages 1–10, Sept 2013.
[4] Alex Biryukov, Dmitry Khovratovich, and Ivan Pustogarov. Deanonymisation of clients in bitcoin P2P network. *CoRR*, abs/1405.7418, 2014.
[5] S. M. Bellovin. Security problems in the tcp/ip protocol suite. *SIGCOMM Comput. Commun. Rev.*, 19(2):32–48, April 1989.
[6] Hitesh Ballani, Paul Francis, and Xinyang Zhang. A study of prefix hijacking and interception in the internet. *SIGCOMM Comput. Commun. Rev.*, 37(4):265–276, August 2007.
[7] Robert T. Morris. A weakness in the 4.2bsd unix2 tcp/ip software. 03 2018.
[8] Yossi Gilad and Amir Herzberg. Off-path attacking the web. *CoRR*, abs/1204.6623, 2012.
[9] Z. Qian and Z. M. Mao. Off-path tcp sequence number inference attack - how firewall middleboxes reduce security. In *2012 IEEE Symposium on Security and Privacy*, pages 347–361, May 2012.
[10] Zhiyun Qian, Z. Morley Mao, and Yinglian Xie. Collaborative tcp sequence number inference attack: How to crack sequence number under a second. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 593–604, New York, NY, USA, 2012. ACM.
[11] Roya Ensafi, Jong Chun Park, Deepak Kapur, and Jedidiah R. Crandall. Idle port scanning and non-interference analysis of network protocol stacks using model checking. In *Proceedings of the 19th USENIX Conference on Security*, USENIX Security'10, pages 17–17, Berkeley, CA, USA, 2010. USENIX Association.
[12] Yue Cao, Zhiyun Qian, Zhongjie Wang, Tuan Dao, Srikanth V. Krishnamurthy, and Lisa M. Marvel. Off-path TCP exploits: Global rate limit considered dangerous. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 209–225, Austin, TX, 2016. USENIX Association.
[13] Yossi Gilad, Amir Herzberg, and Haya Shulman. Off-path hacking: The illusion of challenge-response authentication. *CoRR*, abs/1305.0854, 2013.
[14] bitcoin/bitcoin. src/chainparams.cpp:132. https://github.com/bitcoin/bitcoin/blob/8222e057fe60934a57b1d8226b0e1bd071f8dac2/src/chainparams.cpp#L132.
[15] Yossi Gilad and Amir Herzberg. Off-path attacking the web. In *Proceedings of the 6th USENIX Conference on Offensive Technologies*, WOOT'12, pages 5–5, Berkeley, CA, USA, 2012. USENIX Association.

[16] Paul Vixie. Dns and bind security issues. In *Proceedings of the 5th Conference on USENIX UNIX Security Symposium - Volume 5*, SSYM'95, pages 19–19, Berkeley, CA, USA, 1995. USENIX Association.

[17] Christoph L. Schuba. Addressing weaknesses in the domain name system protocol, 1993.

[18] Keith M Martin. *Everyday Cryptography: Fundamental Principles and Applications*. Oxford University Press, Oxford, 2012.

[19] D. Atkins and R. Austein. Threat analysis of the domain name system (dns). RFC 3833, RFC Editor, 2004.

[20] various. bitcoin/bitcoin. https://github.com/bitcoin/bitcoin/graphs/contributors.

[21] Christian Decker and Roger Wattenhofer. Bitcoin transaction malleability and mtgox. *CoRR*, abs/1403.6676, 2014.

[22] T. Bamert, C. Decker, L. Elsen, R. Wattenhofer, and S. Welten. Have a snack, pay with bitcoins. In *IEEE P2P 2013 Proceedings*, pages 1–5, Sept 2013.

[23] Blockstream - open hash contractor. https://blockstream.com/team/luke-dashjr/. (Accessed on 2017-07-14).

[24] Blockstream - infrastructure tech engineer. https://blockstream.com/team/christian-decker/. (Accessed on 2017-07-14).

[25] Blockstream - infrastructure tech engineer. https://blockstream.com/team/pieter-wuille/. (Accessed on 2017-07-14).

[26] Blockstream - technical advisor. https://blockstream.com/team/matt-corallo/. (Accessed on 2017-07-14).

[27] Protocol documentation. https://en.bitcoin.it/wiki/Protocol_documentation. (Accessed on 2017-06-11).

[28] Andrew Miller, James Litton, Andrew Pachulski, Neal S. Gupta, Dave Levin, Neil Spring, and Bobby Bhattacharjee. Discovering bitcoin's public topology and influential nodes. 2015.

[29] Sebastian Feld, Mirco Schönfeld, and Martin Werner. Analyzing the deployment of bitcoin's p2p network under an as-level perspective. *Procedia Computer Science*, 32:1121 – 1126, 2014. The 5th International Conference on Ambient Systems, Networks and Technologies (ANT-2014), the 4th International Conference on Sustainable Energy Information Technology (SEIT-2014).

[30] Global bitcoin nodes distribution - bitnodes. https://bitnodes.earn.com/. (Accessed on 2017-02-25).

[31] Zi Hu, John Heidemann, and Yuri Pradkin. Towards geolocation of millions of ip addresses. In *Proceedings of the 2012 Internet Measurement Conference*, IMC '12, pages 123–130, New York, NY, USA, 2012. ACM.

[32] Ingmar Poese, Steve Uhlig, Mohamed Ali Kaafar, Benoit Donnet, and Bamba Gueye. Ip geolocation databases: Unreliable? *SIGCOMM Comput. Commun. Rev.*, 41(2):53–56, April 2011.

[33] Network snapshot - bitnodes. https://bitnodes.earn.com/nodes/?q=Satoshi:0.8.3. (Accessed on 2017-06-24).

[34] Nvd - cve-2013-5700. https://nvd.nist.gov/vuln/detail/CVE-2013-5700. (Accessed on 2017-06-24).

[35] Us-cert: Advisory (ca-1998-01) smurf ip denial -of-service attacks. https://www.cert.org/historical/advisories/CA-1998-01.cfm. (Accessed on 2017-07-15).

[36] Us-cert: Alert (ta13-088a) dns amplification attacks. https://www.us-cert.gov/ncas/alerts/TA13-088A. (Accessed on 2017-07-15).

[37] The ddos that knocked spamhaus offline (and how we mitigated it). https://blog.cloudflare.com/the-ddos-that-knocked-spamhaus-offline-and-ho. (Accessed on 2017-07-15).

[38] Spamhaus. https://www.spamhaus.org/. (Accessed on 2017-07-15).

[39] Vern Paxson. An analysis of using reflectors for distributed denial-of-service attacks. *SIGCOMM Comput. Commun. Rev.*, 31(3):38–47, July 2001.

[40] Bitcoin wiki: Bitcoind. https://en.bitcoin.it/wiki/Bitcoind. (Accessed on 2017-04-17).

[41] Bitcoin core code repository. https://github.com/bitcoin/bitcoin. (Accessed on 2017-01-25).

[42] bitcoin/bitcoin. src/net_processing.cpp:2085. https://github.com/bitcoin/bitcoin/blob/d42a4fe5aaae60f33a89bde78f21820abefce922/src/net_processing.cpp#L2085.

[43] bitcoin/bitcoin. src/net_processing.cpp:2712. https://github.com/bitcoin/bitcoin/blob/d42a4fe5aaae60f33a89bde78f21820abefce922/src/net_processing.cpp#L2712.

[44] Blockhain.info - mempool transaction count. https://blockchain.info/charts/mempool-count?timespan=24h. (Accessed on 2017-06-28).

[45] Ieee standard for ethernet. *IEEE Std 802.3-2012 (Revision to IEEE Std 802.3-2008)*, pages 1–3747, Dec 2012.

[46] J. Postel. Internet protocol. RFC 791, RFC Editor, 1981.

[47] V. Cerf and R. Kahn. A protocol for packet network intercommunication. *IEEE Transactions on Communications*, 22(5):637–648, May 1974.

[48] Memcrashed - major amplification attacks from udp port 11211. https://blog.cloudflare.com/memcrashed-major-amplification-attacks-from-port-11211/. (Accessed on 04/11/2018).

[49] 1.7 tbps ddos attack — memcached udp reflections set new record. https://thehackernews.com/2018/03/ddos-attack-memcached.html. (Accessed on 04/11/2018).

[50] bitcoin/bips. bip-0150.mediawiki:-1. https://github.com/bitcoin/bips/blob/f1485fdb5f1875bf96d5ecdcfa27b275ac45ef6a/bip-0150.mediawiki#L-1.

[51] Divesh Aggarwal, Gavin K. Brennen, Troy Lee, Miklos Santha, and Marco Tomamichel. Quantum attacks on bitcoin, and how to protect against them, 2017.