Total Eclipse: How To Completely Isolate a Bitcoin Peer

Adja Elloh Yves-Christian, Badis Hammi, Ahmed Serhrouchni and Houda Labiod
INFRES Laboratory, Telecom Paristech, France
elloh.adja, badis.hammi, ahmed.serhrouchni, houda.labiod@telecom-paristech.fr

***Abstract*—In the cryptocurrency world, *Bitcoin* holds the first place in terms of market cap and currency price, which makes it the first target and victim of attack attempts. Indeed, there are various attacks against cryptocurrencies in general and *Bitcoin* in particular, e.g., *block withholding*, *transaction malleability* and the *Eclipse* attack. The latter, allows an attacker to completely isolate a peer and to monopolize all permanent connections from/to the victim. However, in this attack, the non permanent connections, remains non monopolized by the attacker, which can disturb the attack success. In this paper, we propose (1) a characterization of the misbehavior mechanism applied by *Bitcoin* and its weaknesses; and (2) a new method to realize the *Eclipse* attack which monopolizes all the peer's connections, even the non permanent ones, with a minimal number of IP addresses. Our characterization and attack realization, were performed on the main *Bitcoin* network and on a real client.**

***Index Terms*—Bitcoin, Blockchain, Eclipse attack, Peer-2-Peer, Security, Vulnerability.**

I. Introduction

The blockchain is a novel technology, which features make it inescapable for all existing cryptocurrencies. It represents a special public ledger, shared among peers of a network. Appending data to this ledger is possible only after a consensus among all the peers [1]. More precisely, a blockchain is a list of chained information objects. Each structured object in the chain is called block. A block is basically an information container structured according to the application above and the consensus below. Indeed, there is a network of peers under a blockchain. These peers must agree so that a block can be added to the blockchain or not. Thus, there is a need for a consensus mechanism to reach an agreement between all nodes of the network. It exists numerous consensus types and algorithms. They differ according to the cryptocurrency and to the blockchain used behind it.

Bitcoin represents the most known and used cryptocurrency. It uses the PoW as a security mechanism against *Sybil* attacks [2]. There are two types of nodes in the *Bitcoin* network: (1) nodes who participate in the creation of the blocks, called *miners* and (2) nodes that only can read the transaction blocks. It is well demonstrated and studied that a blockchain remains safe while more than 50% of its miners are honest [3]. However, this affirmation concerns only the blockchain's consensus mechanism and is not applicable to the other components of the ecosystem such as the network below. Indeed, many studies show that we can create troubles in the *Bitcoin* network without controlling more than 50% computational power and numerous attacks have been realized such as *double-spending* [4][5], *block withholding* [6], *Eclipse* [7], *mining pool* [8], *transaction malleability* [8], *selfish mining* [9] [10] and *the balance attack* [11].

One important characteristic of *Bitcoin* blockchain is its openness, all people in the world can see *Bitcoin*'s traffic without necessarily belonging to the *Bitcoin* network. The *Bitcoin* blocks and transactions transit clearly without any encryption, and only integrity protection and immutability services are performed on them. The network works over TCP connections without any well-known security protocol such as TLS [12] or IPSEC [13]. Consequently, any user can modify, drop, inject and delay *Bitcoin*'s data traffic.

Contribution

The contribution of this paper is twofold: (1) we made, through a real implementation, a deep analysis of the misbehavior detection mechanism applied by *Bitcoin* in order to characterize its reactions regarding suspicious messages and IP addresses. (2) we propose a new attack method for the *Eclipse* attack, that we call *Total Eclipse* and which aims to completely isolate a *Bitcoin* peer and to monopolize all its connections, even the non permanent ones, all within optimizing the number of used IP addresses.

This paper is organized as follows: Section II gives an global overview on *Bitcoin*'s ecosystem. Then, Section III describes the related works that treated the *Eclipse* attack. Afterwards, Section IV depicts our characterization of the punishment system of *Bitcoin*. Then, Section V details our attack method called *Total Eclipse*. Afterward, Section VI discuss and analyzes our evaluation campaign. Next, the Section VII gives some countermeasures to the discussed attack. Finally, the Section VIII concludes the paper.

II. Bitcoin overview

It exists numerous surveys and papers that describe blockchains and *Bitcoin* [14] [15]. Nonetheless, they approach the cryptographic aspect of the mechanism (e.g., PoW), and take less interest in the other aspects such as the networking functioning between the peers.

978-1-5386-8187-9/18/$31.00 ©2018 IEEE

Thus, in this section, we give a global overview of the ecosystem behind *Bitcoin*, especially, the aspects non treated by the other surveys.

A. Network Protocol

Bitcoin is an open peer-to-peer network, to which any user can connect through a *Bitcoin Client*. Each new client arriving in the network can find other peers through a *DNS Peers List*, downloaded from Bitcoin DNS Servers which addresses are hardcoded in the client software. It exists two types of addresses in the client node: (1) a DNS address list for the bootstrap of the node in the initial network connection and (2) the list of discovered nodes. The list of discovered nodes is composed of: (1) nodes announced by the DNS (trusted nodes), (2) nodes announced by themselves and (3) nodes announced by trusted nodes. Each node can at most have 125 active connections with other peers. The peers communicate through different types of Bitcoin messages. If a peer discovers one or more new blocks, it sends an *INV* message, to all its peers, containing the digest of all the blocks discovered. Each peer that receives an *INV* message, checks if the blocks corresponding to the received hashes are already held. If not, the peer sends a *GETDATA* message to request the unknown blocks to the node from which it received the *INV* message. Then, it will wait for a maximum time of 20 minutes for the response. If nothing is received within this interval, it closes the connection. If there is a response, the requested block is sent through the message *BLOCK*. The message *BLOCK* can also be sent to a peer without a precedent *INV* message. It is called in this case, unsolicited block. The following sections will describe in detail the messages structure.

In the *Bitcoin* protocol, nodes with a public IP address are free to request connections to other nodes. In the receiver side, that is called unsolicited incoming connection. In the same way, messages can be sent to other peers without a request or permission through the unsolicited connection.

B. Transactions

In *Bitcoin* ecosystem, peers exchange coins through transactions. Each transaction is hashed and signed before transmission to ensure non-repudiation and integrity properties. To be aware of its own balance, each client needs to download all the blockchain's history in order to find all the transactions in which it received and spend coins. Each new transaction is broadcasted in the whole *Bitcoin* network in order to have the validation of the network and to allow each client to have a copy.

C. Blocks

As described, the blockchain is a list of chained blocks, where each block contains the hash of the previous block, in the way to build a chain. More precisely, sets of transactions in the network are gathered in blocks and chained to the blockchain. Each block contains also a solution to a cryptographic puzzle called Proof of Work. A peer must compute the requested PoW in order to be allowed to append the block into the blockchain. The difficulty of this puzzle and the necessary computational power to execute it, determines the block-rate of the network.

The PoW is basically a solution to a cryptographic puzzle, a solution that is difficult to produce but easy to verify. It represents a hash found through brute-force that must be less than a well-known threshold called target. A block can be added permanently to the blockchain if its PoW is accepted by all network peers. The difficulty is, therefore, to find the right hash. One can observe that can be many values smaller than the target. Accordingly, it can sometimes happen that more than one solution of the cryptographic puzzle is found by different peers, almost simultaneously for the same block. This will cause a disagreement among the peers of the network. Consequently, more than one block will be added to the chain at the same level, and the blockchain will split into several branches. This issue is communally called blockchain fork.

In that case, the peers have to converge to a single branch through a mutual agreement. That is possible by choosing the longest branch, namely the branch containing the highest amount of computational work. The blocks in the shortest branch are called uncle blocks which become orphaned blocks after the abort of the shortest branch. The target is related to the hash power of the network. The difficulty of the crypto-puzzle is modified every 2016 blocks by the protocol in order to have a new block in the chain averagely every 10 minutes.

D. Bitcoin messages

As described above, peers in *Bitcoin* network communicate through different types of messages [16], established by the protocol[1]. We describe in the following, some of them. Our explanation is limited to those messages because they are the only messages related to the purpose of the paper.

1) *INV*: it is used by one peer to advertise other peers about its knowledge of one or more object.
2) *VERSION*: it is used by a peer to open a outgoing connection. The peer that receives this message has to respond with another *VERSION* message, containing the version of it own *Bitcoin* protocol.
3) *ADDRESS*: It is used by a peer to provide a list of addresses of known or discovered nodes.
4) *GETDATA*: it is used by a node to reply to an *INV* message in order to retrieve a specific object advertised by the peer that sent this *INV* message.
5) *BLOCK*: it is sent by a node in response to the *GETDATA* message in order to send the data block requested in the *GETDATA* essage.

[1]https://en.bitcoin.it/wiki/Main_Page

6) *PING*: it is sent by a node to request if the *TCP/IP* connection is still valid. A *PING* message without a valid response will cause the closure of the connection.
7) *PONG*: it represents a response to the *PING* message, a response that confirms the validity of the *TCP/IP* connection. A *PONG* message occurring without a precedent *PING* is ignored.

E. Miners

Each peer can compete in the consensus process to add blocks to the chain. This participation is called "mining" and the peer "miner" [17]. There are two types of miners: (1) the mining farm and (2) the mining pool. The first represents usually a unique identity with a consequent computational power which acts in the consensus process. The mining pool comprises many entities or many "weak" nodes that put together their computational power in the way to reach a high power and compete for the consensus process. In both cases, only one IP address is used. For the mining farm, it represents the machine's IP address and for the pool, it is generally the gateway's address that is used (in some cases, one pool can use numerous IP addresses).

III. Related work

There are already two methods to realize the *Eclipse* attack in a *Bitcoin* network. The first was published by Heilman *et al.* [7], which targets principally the outgoing connections of the victim while ignoring unsolicited incoming connections, since they assume that a *Bitcoin* client can have at most 9 connections. However, even if it is known that a *Bitcoin* client does not use all its available 125 connection slots at the same time, a robust attack method must also predict the uncommon cases. Indeed, if the victim receives only one unsolicited connection (using one of the 117 remaining connection slots) from a honest peer, it can break up the attack.

The second proposal, described by the same authors [18] depicts an *Eclipse* attack realized by a malicious Autonomous System (AS). In this case, the malicious AS realizes a man in the middle attack to isolate the *Bitcoin* client. Indeed, the AS filters the targeted client's connections (from/to the victim). Then, it controls all the incoming and outgoing traffic to/from the victim, allowing only its corrupted nodes to communicate with it. However, corrupting an AS represents a very difficult task, which makes the setup of this attack very difficult to reach.

IV. Characterization of the misbehavior punishment system of Bitcoin

In the following, we define a misbehaving or a malicious node, as a peer that sends false information (about transactions or blocks) to other peers. In *Bitcoin* ecosystem, there are rules that punish malicious nodes. More precisely, each *Bitcoin* node keeps a "misbehaving score" for all its connected peers. When the score reaches a determined value (the default value is set to 100), the connection with that node is closed and the node is banned for a default value of 24 hours. Indeed, every malicious behavior against a peer will increase the misbehaving score of its instigator.

The increasing value depends on the rule that the misbehaving node transgress. However, there are no details given about these values in the *Bitcoin*'s developer guide nor in the literature. Thus, we get in *Bitcoin*'s core source code and extracted the set of punishment rules related to the messages in which we are interested. The latter are depicted in Table I.

Rule	Increased value
if the size of a *INV* message is $>$ 50000 bytes	20
If the size of a *ADDR* message is $>$ 1000 bytes	20
If the size of a *GETDATA* is $>$ 50000 bytes	20
If a *GETDATA* message is replayed	1
If a *BLOCK* message is replayed	1
If a *PING* message is replayed	1

TABLE I: Punishment rules and increasing values

Table I describes only the rules related to the messages' structure. In order to push our characterization further, we realized a set of experiments in order to study how a peer behaves in the case where it is flooded with numerous messages having a valid structure. Following these experiments we noticed that:

1) The quantity of messages needed to disconnect and ban a peer is different according to messages types. Table II describes the required limit number related to each message type. For example, if a peer sends consequently 24 messages of type *PING*, the receiver peer closes the ongoing connection with this peer.
2) The misbehavior score is set to 0 each time the sender node changes the message type. In other words, if a peer A sends 23 *PING* messages to a peer B, the misbehavior score that B holds for A is 23. Then, if A sends a *GETDATA* message to B, the misbehavior score that B holds for A is reset to 0.
3) Each client maintains a misbehavior score of its directly connected nodes. This score must not exceed a certain value (set to 100 by default). The score is incremented whenever the rules of the protocol are broken by the peer. When the score's maximum allowed value is reached for a peer, the peer is disconnected and banned.
4) Each *Bitcoin* peer is identified by a couple of IP address and port number. However, two connections with the same IP address and two different port numbers, are considered by the receiver as provided by two different peers

Thus, if a malicious user wants to flood a *Bitcoin* peer, it must only change messages type frequently

while respecting not reaching the limits depicted in Table II.

Message type	max
GETDATA	24
PING	24
INV	24
BLOCK	11

TABLE II: Maximum message number before being banned

V. Towards a Total Eclipse attack

In this section, we describe first, the *Eclipse* attack and its *modus operandi*. Afterward, we describe how we exploit the *Bitcoin*'s punishment mechanism in order to enhance the method used for *Eclipse* attack.

A. Eclipse Attack

The *Eclipse* attack was already known in the area of overlay networks [19]. However, its use against *Bitcoin* was put under the spotlight by Heilman *et al.* [7]. In the latter case, it targets a *Bitcoin* peer and aims at monopolizing all its incoming and outgoing connections, in order to isolate and control the victim's view of the global network. This can lead (1) to waste the victim's computational power, (2) to make the victim compute the PoW for the attacker or (3) to realize numerous other attacks against *Bitcoin* network and clients [8].

The *Eclipse* attack is realized as follows: When a new node joins the *Bitcoin* network, it must connect to the DNS seeders which IP addresses are hard-coded in the client's source code. By connecting to the DNS seeders the new client retrieves a list of IP addresses relative to the active nodes. Then, the new peer chooses randomly, from the retrieved list, 8 IP addresses based on /16 netmask, with which it will establish a long-lived outgoing connections. Considering the 8 permanent connections, each node can accept a maximum of 125 incoming connections, including unsolicited connections (connections initiated by other peers).

One of the main features of the *Bitcoin* ecosystem is the full anonymity of its peers. The lack of authentication is the weakness that the *Eclipse* attack exploits:

1) The attacker must control numerous /16 netmask IP addresses.
2) Using the controlled addresses, the attacker initiates connections with the target. Thus, the victim adds these addresses to the list of its known peers.
3) If one or more existing long-lived connections break up; or if the miner node reboots[2]; or if the bandwidth of one established connection is under a certain threshold, then, the victim will choose randomly from the list of its known peers some IP addresses to replace the wasted long-lived connection.

The attacker will only wait until the victim chooses its controlled IP addresses, which gives him the control of the victim's view of the network. Obviously, the more are the controlled IP addresses, the more is the probability of the attack success.

B. Total Eclipse attack

As presented previously (Section V-A), the *Eclipse* attack relies on the hypothesis ensuring that "a *Bitcoin* client will not use all its 125 connections slots. But, it uses at most, only 9 connections [7]".

In this work, we present an alternative to the *Eclipse* attack as presented earlier in Section V-A. Our attack realization does not rely on a specific hypothesis. Moreover, it monopolizes the 125 victim's connection slots and not only the 8 permanent ones.

The attack method that we propose is efficient on all *Bitcoin* nodes, whether they are simple nodes, farming nodes or pools. Indeed, the nodes having high computation power, like farming nodes or pools[3], which have a more consistent connections capability, doesn't represent an exceptional case since we focus on the IP address it uses. Our attack is composed of two main steps: (1) *Eclipse* attack and (2) *Stonewall* attack.

1) Eclipse attack phase: The IP addresses known by a *Bitcoin* client are divided into two groups called *new buckets* and *tried buckets*. A *bucket* represents a list of peer addresses. The *new bucket* is the list of all the peers with which the *Bitcoin* client had never established an outgoing connexion. It is composed by 1024 small *buckets* which can accommodate each, 6 IP addresses. The *tried buckets* contain the list of IP addresses of peers with which the client have already established outgoing connexion. It contains 256 small *buckets* which can also accommodate 6 IP addresses for each one.

Each *Bitcoin* client, updates continuously its list of known **functional** peers. Indeed, each time, it choses randomly 64 *buckets* from the existant 1024 *new buckets*. Then, it tries to establish an unsolicited connection with each IP address of the selected group[4]. If a connexion is successfully established, the client will first breaks it up, then, it will move the tested address from *new bucket* type to *tried bucket* type.

When a client wants to establish an outgoing connection, it chooses randomly an IP address from the *tried buckets*. Thus, our goal is to fill the *new buckets* with our controlled addresses. Then, the victim will move these addresses to its *tried buckets* in order to use them after. More precisely:

1) the attacker sends a *VERSION* message to the victim;

[2]In this case it loses its 8 long-lived connections.

[3]Herein, we consider the pools that use only one IP address.

[4]$64 \times 6 = 384$ IP addresses.

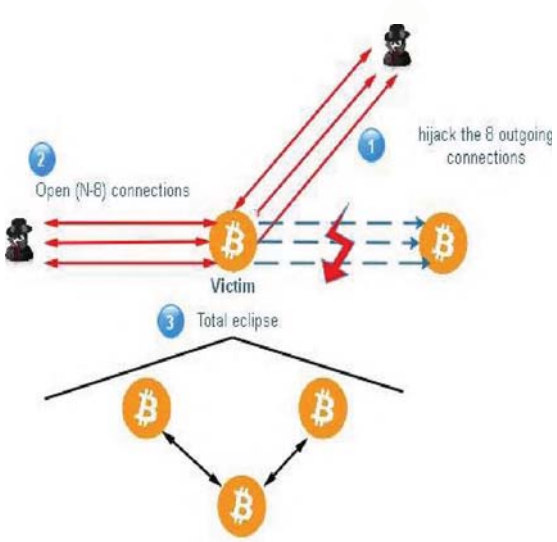

Fig. 1: Total Eclipse Attack steps

2) this way, the victim will establish a connection with the attacker and respond with another *VERSION* message;
3) once the connection is open, the attacker sends an *ADDR* message to the victim which contains a list of controlled IP addresses;
4) the victim will add the received IP addresses to its *new buckets*;
5) the victim will move the controlled IP addresses from *new* to *tried* after testing the connections establishments;
6) the attacker will wait until the victim establishes 8 long-lived connections with the controlled IPs.

In our work, we use the *Eclipse* attack as it was designed by Heilman *et al.* [7] in order to control the 8 long-lived connections. This, represents only the first step of our method. The second step consists in applying the Stonewall attack, in order to control the remaining 117 connections.

2) Stonewall attack phase: Any Internet peer node, whether it represents a *Bitcoin* node or not, can be an attacker. The first step consists in capturing the packets that will be replayed later by sniffing *Bitcoin* network related traffic. We prefer to replay existing valid messages (we use the payload of the captured messages), instead of creating new messages (payloads) for time and computation optimization, for messages like *INV*, *ADDR*, etc. but especially for *BLOCK* messages. Indeed, *INV* and *ADDR* messages are not signed but *BLOCK* messages are signed which makes their creation costly. Thus, replaying them reduces considerably the attacker's efforts. In both cases, the reaction of the victim will be the same and the received blocks will be dropped. In fact, (1) if the blocks are created they will have a valid signature but a replayed/wrong content, consequently, the block will be dropped by the victim. (2) if the messages are replayed than their content is replayed, and they are therefore dropped.

Algorithm 1: Stonewall attack

```
Declaration:
Ping: Packet                                   ▷ Sniffed packet
GetData: Packet
Block: Packet
Inv: Packet
C: Connection                                  ▷ an opened connection
S: List [n] Connection                         ▷ Bitcoin peer connections
n: Integer                                     ▷ number of connections (117)
V: IP                                          ▷ victim's IP
1:  function ExtractPayload(x: Packet)         ▷ Extracts payload data from Packet
2:      return DataPayload
3:  function OpenConnection(V: IP, Port: Integer)
4:      return Connection
5:  function IsClosed(V: IP, Port: Integer)    ▷ Verifies if a connection is closed
6:      return Boolean
7:  procedure Send(Pay: DataPayload, C: Connection)
    ▷ Sends the payload Pay through the connection C
8:  procedure CreateFlow                       ▷ Create flow towards the victim through different connections
9:      while j ≤ n do
10:     Port ← Random(1024 : 65535)
11:     C ← OpenConnection(V, Port)
12:     for i ← 1 To 3 do
13:         Send(ExtractPayload(Ping),C)
14:         Send(ExtractPayload(GetData),C)
15:         Send(ExtractPayload(Block),C)
16:         Send(ExtractPayload(INV),C)
        end
17:     j ← j + 1
        end
18:     while True do
        if IsClosed(V, Port) then
19:         Port ← Random(1024 : 65535)
20:         C ← OpenConnection(V, Port)
21:         Send(ExtractPayload(Ping),C)
22:         Send(ExtractPayload(GetData),C)
23:         Send(ExtractPayload(Block),C)
24:         Send(ExtractPayload(INV),C)
        end
        end
```

The second step consists in establishing connections with the victim to monopolize its 117 remaining connection slots. **To reach this goal, and according to our characterization described in Section IV, the attacker uses only one IP address**. One public IP address can open 64511 connections (from port 1024 to port 65535), which is more than sufficient to monopolize all the 117 connection slots of the victim. Indeed, the same IP with different TCP port is interpreted as different peers by the victim. To keep these connections open, the attacker replays the captured messages, while ensuring not to transgress the rules described in Table I and Table II. The attacker tries to refresh periodically the opened connection by closing and opening new connections. Algorithm 1 describes the steps of the *Stonewall* attack.

Mainly, the attacker runs many threads (n) in order

to establish with the victim, gradually, n connections (with $n = 117$). After the establishment of each connection, the attacker sends different types of data packets, different times all within ensuring not closing the connection. In Algorithm 1, we use four different data messages (*PING, GETDATA, BLOCK* and *INV*), since they are often exchanged in the network. We replay them only few times, in order to not encourage the victim to close the established connection (which may happen if numerous replays occur). The *is_closed* function checks any connection closure by the victim in order to open a new one. In this way, we maintain unavailable the victim's 117 connection slots. The Figure 1 describes the attack's steps.

VI. Evaluation and Discussion

A. Evaluation framework

In order to evaluate the impact of our attack, we recall, first, the characteristics of the *Bitcoin*'s network connections. All *Bitcoin* connections are performed through TCP. Thus, for sending any type of data to a *Bitcoin* node it is mandatory to open a TCP connection. Since there is no node authentication, any node can open connections to other nodes. It can even initiate the majority of the connections on a determinate target node.

It exists two *Bitcoin* networks in which a node can communicate: (1) the main network and (2) the test network. The main network is the realistic network where all the real transactions and functioning occur. The test network is basically for developers which want to evaluate and test their new solutions. **We have performed all our experimentations in the main network in order to have more realistic results**. Our attack does not impact the network's performances or any other *Bitcoin* peer, since the victim is a controlled peer.

We used the official *BitcoinCore 0.16.0* [20] as *Bitcoin* client. We made three weeks of traffic sniffing in order to have a large set of "replay-able" messages. Thus, we captured more than 200.000 packets from which we selected *BLOCK* (3552 bytes[5]), *PING* (32 bytes), *GETDATA* (61 bytes) and *INV*(745 bytes) messages. Maintaining sending such data does not require a hight bandwidth, which makes it possible for any malicious user, even if it owns limited ressources.

B. Attack consequences

Any *Bitcoin* node can be a victim of a *Total Eclipse* attack unless it disable unsolicited connections. However, nodes that have disabled unsolicited connections are not spared from the simple *Eclipse* attack. There are mainly two situations: (1) victims that have already opened their 8 outgoing connections and (2) nodes that are in boot mode which haven't opened their connections yet. In the first case, the attack will impact the $(n-8)$ remaining connection slots and wait for the liberation of the 8 connections (with $n = 125$). In the second case, the attack will target at the same time the 125 connection slots.

For our experimentations, we realized two attack campaigns: (1) the attacker belongs to the same network as the victim; and (2) the attacker belongs to a different network as the victim. In both cases, we succeeded to realize the attack and to monopolize the victim's connections. We maintained the attack for more than a week, after, we stopped it intentionally.

The Figure 2 describes the evolution of the number of occupied connection slots and their maintaining over the time, for the first 56 hours of the second attack case (the attacker belongs to the same network as the victim). We performed a monitoring each 5 hours. We can observe that 120 connections are constantly maintained. In the other hand, there are continuous variations of five connections due to the continuous closing and opening of connections with the victim.

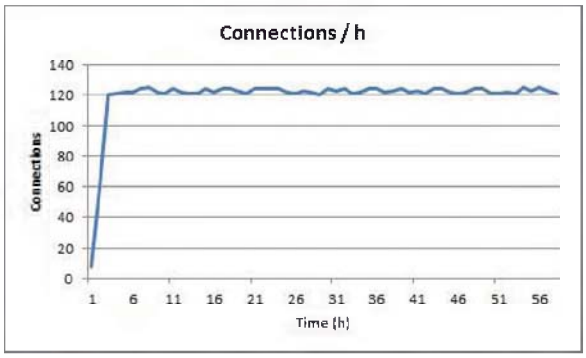

Fig. 2: Occupied connection slots evolution

The Figure 3 describes a boxplot of the time needed to perform the *Total Eclipse* attack (without considering the 8 long-lived connections) over 30 experimentations. The mean value obtained is equal to $220.62ms$. One can also note that 75% of the times values have a value lower than $270ms$.

VII. Countermeasures

We showed in this paper that *Bitcoin*'s "Misbehaving" score solution is not efficient to avoid the monopolization of client's connections. We show also how to proceed in order to completely isolate a *Bitcoin* peer, without a consequent computation power. We give in the following two proposals that can help in avoiding *Eclipse* attacks.

Unsolicited Connection We do not advise to totally disable the unsolicited connections. A client that owns only its 8 long-lived connections is more exposed to *Eclipse* attacks for long periods. Indeed, knowing that unsolicited incoming connections are unpredictable, if a client under an *Eclipse* attack receives messages from one or more unsolicited connections, it can make him aware about another vision of the network, which

[5]Only for one *BLOCK* message.

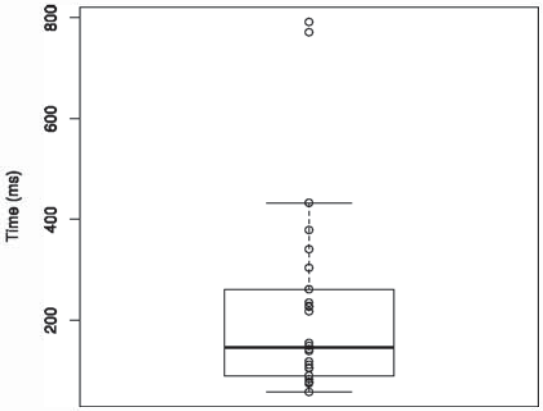

Fig. 3: boxplot of the time needed to perform the *Total Eclipse* attack

makes him disconnect ongoing connections and stop the attack. As a solution, we advertise to make a filtering of unsolicited connections, on the way to avoid that a singular IP address can open more than one connection simultaneously or successively. We are aware that this proposal does not totally solve the problem of *Eclipse* attacks, however, it makes more difficult its realization, since the attacker must control a consequent number IP addresses.

Enhancement of misbehaving detection We recommend making an optimal, reactive and fast-detection misbehaving system. The fact that the misbehaving score re-initialized after each message change represents a real weakness that we can solve with a permanent increase of score independently to the "misbehaving" type.

VIII. Conclusion

It is always affirmed that *Bitcoin* system is protected against attacks especially Denial of Service (DoS). Currently, this statement is true, but, only considering the cryptographic side of *Bitcoin* system. Besides, researches looked at how to use other *Bitcoin*'s components to perpetrate attacks.

Bitcoin relies on a peer-to-peer network, which makes the network's characteristics and weaknesses as potential attack vectors. In this paper we made an experimental campaign that aims at characterizing the punishment and ban system used by *Bitcoin* against malicious nodes. Following this characterization, we exploit the discovered weaknesses to propose an enhanced *Eclipse attack* that we call *Total Eclipse attack* which targets the total isolation of a *Bitcoin* peer, realizing a of denial of service on the victim.

In this work, we proposed a set of countermeasures that aim to detect and avoid *Eclipse* attacks. Thus, for our future works we plan: (1) to implement these countermeasures and (2) to study and evaluate their effectiveness.

References

[1] Mohamed Tahar Hammi, Badis Hammi, Patrick Bellot, and Ahmed Serhrouchni. Bubbles of Trust: A decentralized blockchain-based authentication system for IoT. *Computers & Security*, 78:126–142, 2018.

[2] John R Douceur. The sybil attack. In *International Workshop on Peer-to-Peer Systems*, pages 251–260. Springer, 2002.

[3] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. page 12, 2008.

[4] Ghassan Karame, Elli Androulaki, and Srdjan Capkun. Two bitcoins at the price of one? double-spending attacks on fast payments in bitcoin. *IACR Cryptology ePrint Archive*, 2012(248), 2012.

[5] Meni Rosenfeld. Analysis of hashrate-based double spending. *arXiv preprint arXiv:1402.2009*, 2014.

[6] Samiran Bag, Sushmita Ruj, and Kouichi Sakurai. Bitcoin block withholding attack: Analysis and mitigation. *IEEE Transactions on Information Forensics and Security*, 12(8):1967–1978, 2017.

[7] Ethan Heilman, Alison Kendler, Aviv Zohar, and Sharon Goldberg. Eclipse attacks on bitcoin's peer-to-peer network. In *USENIX Security Symposium*, pages 129–144, 2015.

[8] Mauro Conti, Chhagan Lal, Sushmita Ruj, et al. A survey on security and privacy issues of bitcoin. *arXiv preprint arXiv:1706.00916*, 2017.

[9] Johannes Göbel, Holger Paul Keeler, Anthony E Krzesinski, and Peter G Taylor. Bitcoin blockchain dynamics: The selfish-mine strategy in the presence of propagation delay. *Performance Evaluation*, 104:23–41, 2016.

[10] Kartik Nayak, Srijan Kumar, Andrew Miller, and Elaine Shi. Stubborn mining: Generalizing selfish mining and combining with an eclipse attack. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pages 305–320. IEEE, 2016.

[11] Christopher Natoli and Vincent Gramoli. The balance attack or why forkable blockchains are ill-suited for consortium. In *Dependable Systems and Networks (DSN), 2017 47th Annual IEEE/IFIP International Conference on*, pages 579–590. IEEE, 2017.

[12] T. Dierks and E. Rescorla. RFC 5246: The Transport Layer Security (TLS) Protocol Version 1.2. *Internet Engineering Task Force (IETF) , August*, 2008.

[13] S. Frankel and S. Krishnan. RFC 6071: IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap. *Internet Engineering Task Force (IETF) , August*, 2011.

[14] Shaikshakeel Ahamad, Madhusoodhnan Nair, and Biju Varghese. A survey on crypto currencies. In *4th International Conference on Advances in Computer Science, AETACS*, pages 42–48. Citeseer, 2013.

[15] Zibin Zheng, Shaoan Xie, Hong-Ning Dai, and Huaimin Wang. Blockchain challenges and opportunities: A survey. *Work Pap.–2016*, page 25, 2016.

[16] Bitcoin developer reference. url=https://bitcoin.org/en/developer-reference, 2018.

[17] Andreas M Antonopoulos. *Mastering Bitcoin: unlocking digital cryptocurrencies*. " O'Reilly Media, Inc.", 2014.

[18] Maria Apostolaki, Aviv Zohar, and Laurent Vanbever. Hijacking bitcoin: Routing attacks on cryptocurrencies. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pages 375–392. IEEE, 2017.

[19] Atul Singh et al. Eclipse attacks on overlay networks: Threats and defenses. In *In IEEE INFOCOM*. Citeseer, 2006.

[20] Download bitcoin core. latest version: 0.16.0. url=https://bitcoin.org/en/download, 2018.