Bitcoin Chain Width Expansion Denial-of-Service Attacks

Braydon Fuller

September 18th, 2019

Abstract

The attacks leverage unprotected resources for a denial-of-service by filling the disk and exhausting the CPU with unnecessary header and block data. This forces the node to halt operation. The attack difficulty ranges from difficult to easy. There are currently limited guards for some of the attacks that require checkpoints to be enabled. This paper describes a solution that does not require enabling or maintaining checkpoints and provides improved security.

1 Attacks

1.1 Headers Variant *with* Checkpoints

1. The attacker, with a node that is synced up to the last checkpoint, starts mining alternative blocks and artificially adjusting the block time to increase by forty minutes per block. It will initially be difficult to mine the headers. Every 2016 blocks the difficulty should adjust, increasing the rate that the headers can be generated. The maximum rate the difficulty can decrease is by a factor of four. The difficulty will eventually return to the difficulty of the genesis block.

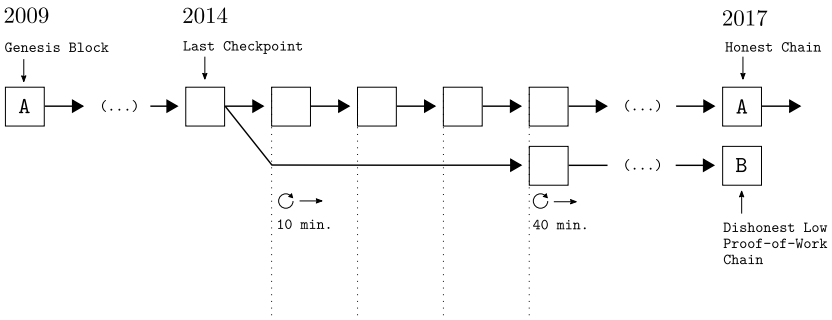

Figure 1: Difficulty Drop

2. The width of the chain is expanded by mining alternative chains from that lowered difficulty fork until the maximum number of headers have been created. The maximum number of headers within a chain is determined by the time of the target peer. This is calculated as $h = (p - l)/i$, where h equals the number of headers, p equals the peer's time, l equals the lowered difficulty time, and i equals the interval between blocks. After that window, the difficulty will adjust to become more difficult in order to create valid blocks.

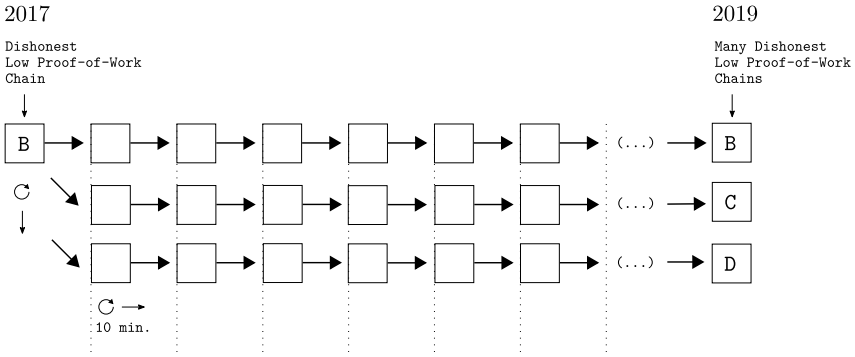

Figure 2: Width Expansion *with* Checkpoints

3. The attacker connects to the target peer, that has checkpoints enabled, and announces to the peer new block inventory via the `inv` message. There will then be a corresponding `getheaders` call from the peer, which the attacker responds with the target `header` payload. The previous process is repeated. Alternative means of transmitting the headers directly via the `headers` message would also deliver the target payload.

4. Eventually the target peer will run out of resources and halt operation.

1.2 Headers Variant *without* Checkpoints

1. The attacker, with a node that only has the genesis block, starts mining alternative blocks and artificially adjusting the block time to increase by ten minutes per block. This process is continued until the maximum number of headers have been created.

2. The mining process is repeated again from the genesis block creating another alternative chain for as many times as necessary to create the desired target payload size. For example 5,000 alternative chains at a block height of 600,000 would give 240GB of header data (80 bytes per header).

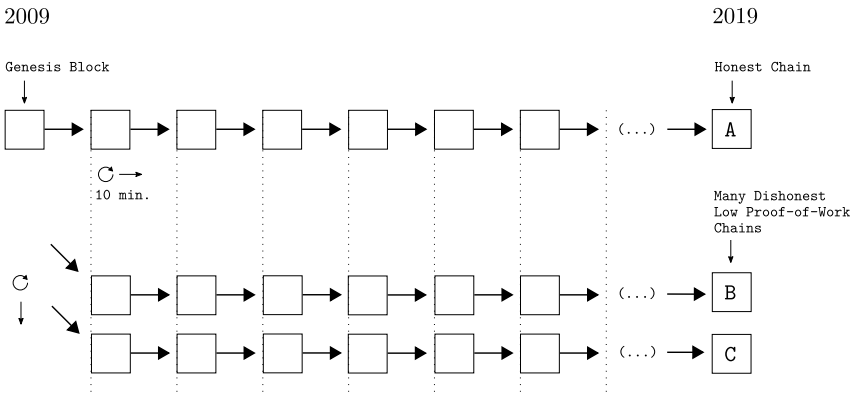

Figure 3: Width Expansion *without* Checkpoints

3. The attacker connects to the target peer, that does not have checkpoints enabled, and announces to the peer new block inventory via the `inv` message. There will then be a corresponding `getheaders` call from the peer, which the attacker responds with the target `header` payload. The previous process is repeated. Alternative means of transmitting the headers directly via the `headers` message would also deliver the target payload.

4. Eventually the target peer will run out of resources and halt operation.

1.3 Blocks Variant *without* Checkpoints

1. The attacker, with a node that only has the genesis block, starts mining alternative blocks and artificially adjusting the block time to increase by ten minutes per block.

2. Once the coinbase maturity elapses, the blocks ared filled with thousands of transactions to the maximum capacity. Blocks are mined until the chain reaches the desired target payload capacity. As there can be an indefinite number of forks and chain width, this target size can be expanded without limit.

3. The attacker connects to the target peer, that does not have checkpoints enabled, and announces to the peer new block inventory via the `inv` message. There will then be a corresponding `getdata` call from the peer, which the attacker responds with the payload blocks. The attacker makes sure that the blocks are not considered orphans, as there is a maximum number of orphans permitted. Each block needs to connect from the genesis block in the correct order. The previous process is repeated.

4. Eventually the target peer will run out of disk space and halt operation.

2 Current Attack Guards

The attack is currently guarded by preventing a fork previous to the last checkpoint. This is widely implemented in Bcoin, Btcd, and Bitcoin Core:

- In a commit added to Btcd it mentions the attack "This prevents storage of new, otherwise valid, blocks from building off of old blocks which are likely at a much easier difficulty and therefore could be used to waste cache and disk space." [1].
- There was a similar commit made to Bitcoin Core at around the same time of February 20th, 2014. It was referencing solving a fingerprinting issue, however also guards against the disk fill attack. See the commit "Don't store or send blocks forked before last checkpoint." [2].
- On October 17, 2014, Bitcoin Core switched to use headers-first synchronization that further guards against the attack. "This means that a disk-fill attack would require PoW.", as mentioned in the pull request, "Headers-first synchronization" [3].
- Upon searching on how to disable checkpoints for Bitcoin Core, there are also mentions of the guard: "But currently the role they [checkpoints] serve is to prevent low difficulty header flooding attacks, and there has been no alternative solution proposed yet (that I know of)." [4].
- And for Bcoin there is now a guard in place added in commit "blockchain: do not accept forked chain older than last checkpoint" after I had independently discovered there to be a disk-fill attack vulnerability in April, 2019 [5].

3 Cost of Attack

The three variants are given a cost analysis based on mining, time and hardware costs. The cost ranges from $1.5M USD, $1K USD and $100 USD to build the payload in around 1 month to 1 day. The payload can be used against multiple target peers. As peers only relay headers that have blocks fully validated, the attack requires the necessary bandwidth to deliver the payload to each target peer. The additional necessary bandwidth costs are not included.

3.1 Headers Variant *with* Checkpoints

The checkpoint heights vary across implementations. The lowest last checkpoint is at height 295,000 (Bitcoin Core), followed by 525,000 (Bcoin) and 560,000 (Btcd). For the cost analysis we will look at Bitcoin Core as it has the lowest difficulty.

It would cost around $1,433,924 USD in mining hardware and electricity to produce 315GB of payload header data in 450 hours with ability to add additional data at a rate of 315GB per hour for $104 USD.

Mining Cost (Phase I = $1,390,650 USD)

- The last checkpoint in Bitcoin Core has a difficulty at 6,119,726,089.128147, and would require 26,284,424,481,772,170,000 hashes [6].
- With a T17 AntMiner with 40TH/s a block could be found given 657,111 seconds (or 7.6 days), and would require around 1,095 miners to equal the hash rate of 10 min blocks.
- Given that T17 uses 2,200W, it would require 739.2kWh (2,200W x 336h / 1,000) to run for two weeks per miner. Given an electric rate of $0.04 USD per kWh, it would cost $29.57 USD to run for two weeks.
- With 1,095 miners it would cost $32,376.96 USD to run for two weeks.
- 1,095 T17 miners would cost $1,390,650 USD (T17 at $1,270 USD).

Electricity Cost (Phase II = $43,169 USD):

- It would take about 2 weeks and 4 $\frac{1}{2}$ days to lower the difficulty to 1.
- That is 448 hours to lower the difficulty to 1 and 985.6 kWh * 1,095 miners at $43,169.28 USD of electricity.

Electricity Cost (Phase III = $104 USD)

- It would be a total of about 18 difficulty adjustments or 36,288 block headers, and a height at 331,288. It would be equivalent time of 144 weeks $(36,288 * 10min * 4)$, or 1,008 days, which is around January 11, 2017 as the starting checkpoint was on April 9, 2014.
- There would be a remaining 878 days to June 7th, 2019, and 126,246 block headers that could be mined at a difficulty of 1.
- To mine 2,500,000,000 additional block headers it would then require 19,803 parallel forks from that point.
- At a difficulty of 1 it takes 4,295,032,833.000015 hashes. With a T17 at 40TH/s, it could produce a block header in 0.000107376 seconds, or 3,600,000 block headers every 3,865,529.5497 milliseconds (∼hour).
- 1,095 T17 miners could produce 3,942,000,000 block headers in 1 hour and 5 minutes. The miners could produce 315GB of data with 2,609.75 kWH at a cost of $104.39 USD.

3.2 Headers Variant *without* Checkpoints

- At a difficulty of 1 it takes 4,295,032,833.000015 hashes. With a T17 at 40TH/s, it could produce a block header in 0.000107376 seconds.
- A single T17 miner could produce 315GB of data (3,942,000,000 headers) in roughly 49 days (48.990218251), at the cost of $1,270 for the miner and $103.4 for electricity at 2,585 kWh at $0.04 per kWh.
- The total to produce 315GB of data would equal $1,373.

3.3 Blocks Variant *without* Checkpoints

- Around 600,000 block headers need to be created and filled with transactions. At a difficulty of 1 it takes 4,295,032,833.000015 hashes.
- A $74.95 USD GekkoScience NewPac miner can produce up to 130Gh/s at 12W. Thus it could produce around 30 blocks per second, and around 5 hours and 30 minutes, and 0.066 kWh.
- At a rate of $0.04 kWh, it nearly equals 1 cent.
- In practice it would generate 2-5 blocks per second limited by the time to produce full blocks. This could be optimized to increase the rate, for example transactions spends are not verified and therefore would not necessarily need to be valid.
- The total cost to produce 315GB of data would be less than $100 dollars and 24 hours of time.

4 Solution

The solution is three parts; download blocks *only* for header chains that exceed the current best validated chain, limit the chain width and total number of alternative header chains, and limit the rate of header messages. This solution does not require that checkpoints configuration be enabled and has improved security.

4.1 Headers-first Synchronization

Full blocks should *only* be downloaded and stored on disk if part of a header chain with greater proof-of-work than the current best validated chain. This is commonly known as headers-first synchronization. It was added to Bitcoin Core in 2014 [7]. The synchronization works by requesting all headers from a loader peer up until recent blocks, and then headers are requested from multiple peers as well as full block data. If there is an issue with the loader peer, the loader peer will be rotated. Multiple blocks are requested at a time from multiple peers within a window ahead of the fully validated chain. Blocks are then added to

the chain as they become available in sequence. This method requires that there be proof-of-work for blocks to be downloaded. It increases the cost of a disk-fill attack from using full blocks to only being able to use headers.

4.2 Limiting Header Chain Width

The total chain width should be limited with a maximum number of possible candidate alternative header chains. The alternative chains with the least amount of proof-of-work should be pruned to maintain the maximum.

The headers variant attack relies on expanding the width of the header chain rather than the height. It is because there is a limited number of headers that can be created that is roughly equal to the current height of the chain. Headers are 80 bytes so 10 alternative chains with height of 600,000 would only be 480MB.

The cost of the attack would require `n` times the current hash-rate to produce `n` times the current growth of disk usage for headers and blocks (1-4MB every 10 minutes). The amount of disk space necessary can be calculated given a maximum growth rate. The disk resources are now protected, however there remains an ability to exhaust CPU resources via the header messages.

4.3 Header Message Rate Limiting

The rate that header messages are received should be limited to reserve CPU resources used for processing valid blocks and forks. This prevents the CPU and resources becoming exhausted by pruning the alternative chains and headers. This is done in two parts; limit the rate of unrequested headers and throttle the `getheaders` requests based on the proof-of-work of the received headers.

It's necessary to be able to accept unrequested header messages; it is how blocks are broadcast over the network. This is enabled via the `sendheaders` message that is sent to a peer. There are also unrequested headers sent to a peer in the event of a chain reorganization, there is a maximum number of headers that are sent in that case.

The total number of unrequested headers messages is therefore predictable to calculate over a period of time. The peer should rate limit the number of unrequested headers to provide the necessary rate for those cases, however preventing the case that such headers would be considered misbehavior. This provides the peer to be able to control the timing of any further headers messages via `getheaders`.

It is necessary to make a subsequent call to `getheaders` after receiving a full set of headers. It provides the ability to request the full header chain from peers. The headers can be from any point in the chain all the way back to the genesis block. There can also be duplicate headers in a response to a `getheaders` call that are necessary to be handled. This is because the locator sent to the remote peer includes hashes from the tip to the genesis, skipping many in-between. The remote peer will send headers from the common hash in that subset, and therefore there may be some overlap.

The probability that there will be a deep chain reorganization is extremely low. This is detailed in the Bitcoin Whitepaper in section 11, titled "Calculations" [8]. The probability that less than 50 percent of the hash-rate produces a greater proof-of-work chain decreases exponentially the greater the depth in height.

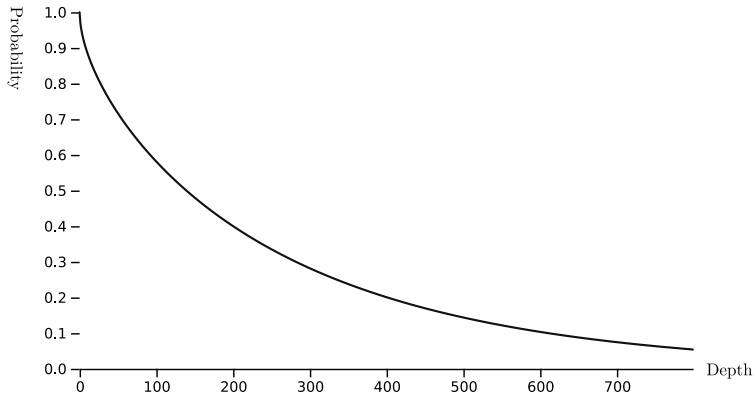

Figure 4: Probability to generate greater proof-of-work chain with 48% hashrate.

However, because of difficulty adjustments, it is highly probable if the initial loaded header chain was dishonest and did not increase in proof-of-work difficulty producing a very low proof-of-work chain. It is therefore necessary for a peer to be able accept headers all the way from the genesis block, even if the current header chain height is very high. However, as the accumulative proof-of-work increases on the fully validated chain, the probability of a deep reorganization decreases exponentially. In that case, header messages from the great depths of the chain are highly improbable to lead to a reorganization, and there is much less priority to be considered and requested.

Subsequent calls of `getheaders` should be throttled based on proof-of-work. Headers that have less work than the current validated chain tip should have the next `getheaders` call delayed. The greater the difference in proof-of-work the more delay should be added. There should be a maximum delay defined, and an inverse exponential function ($x^{\frac{1}{2}}$) should be used to calculate the delay given an input of the difference in proof-of-work. This slows down such requests from exhausting CPU resources, and provides the necessary CPU resources for handling headers that are likely to lead to a reorganization.

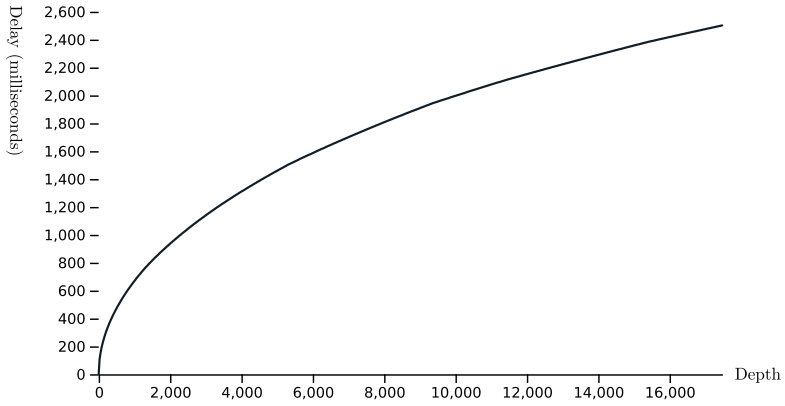

Figure 5: Delay for `getheaders` given the difference in chainwork from height 594000 of mainnet (max of 5s).

5 Implementation

At the time of this writing, the above proposed set of solutions have been implemented in the following implementations:

- Headers-first Synchronization: Bitcoin Core and Bcoin
- Limiting Header Chain Width: Bcoin
- Header Message Rate Limiting: Bcoin

Note: The solution to not accept forks previous to the last checkpoint has been implemented in Bitcoin Core, Bcoin and Btcd. However the last checkpoint across these implementations are at different heights.

6 Acknowledgments

- Matthew Zipkin for discussion of headers-first synchronization and review of this paper.
- Mark Tyneway and Boyma Fahnbulleh for review of this paper.

7 References

1. https://github.com/btcsuite/btcd/commit/50b6e10b5781772b0f6a506535f575d81a95dc7a
2. https://github.com/bitcoin/bitcoin/pull/2910/commits/d8b4b49667f3eaf5ac16c218aaba2136ece907d8
3. https://github.com/bitcoin/bitcoin/pull/4468
4. https://bitcoin.stackexchange.com/questions/1797/what-are-checkpoints
5. https://github.com/bcoin-org/bcoin/commit/b68207610c57139d409ca7958f9c8386f9b8002e
6. https://en.bitcoin.it/wiki=Difficulty
7. https://github.com/bitcoin/bitcoin/pull/4468
8. https://nakamotoinstitute.org/bitcoin/