Short Paper: I Can't Believe It's Not Stake! Resource Exhaustion Attacks on PoS

Sanket Kanjalkar, Joseph Kuo, Yunqi Li, Andrew Miller

University of Illinois, Urbana Champaign

Abstract. We present a new resource exhaustion attack affecting several chain-based proof-of-stake cryptocurrencies, and in particular Qtum, a top 30 cryptocurrency by market capitalization ($952M as of 9th Aug '18). In brief, these cryptocurrencies do not adequately validate the proof-of-stake before allocating resources to data received from peers. An attacker can exploit this vulnerability, even without any stake at all, simply by connecting to a victim and sending malformed blocks, which the victim stores on disk or in RAM, eventually leading to a crash. We demonstrate and benchmark the attack through experiments attacking our own node on the Qtum main network; in our experiment we are able to fill the victim's RAM at a rate of 2MB per second, or the disk at a rate of 6MB per second. We have begun a responsible disclosure of this vulnerability to appropriate development teams. Our disclosure includes a Docker-based reproducibility kit using the Python-based test framework. This problem has gone unnoticed for several years. Although the attack can be mitigated, this appears to require giving up optimizations enjoyed by proof-of-work cryptocurrencies, underscoring the difficulty in implementing and deploying chain-based proof-of-stake.

1 Introduction

Bitcoin mining is expensive, with power consumption estimates ranging from hundreds of megawatts [8, 3] to gigawatts [9]. Naturally, there has been significant interest in reducing this cost. The main idea behind Proof-of-stake (PoS) is to move the mining competition from the physical realm to the financial realm, replacing computational mining with a random lottery based on held coins. Chain-based PoS is a minimal modification of the Bitcoin protocol with this insight. Instead of computing hash functions over an arbitrary space, we compute hash functions of each of the transaction outputs, and compare it against a difficulty threshold, weighted by the coin amount. This approach is employed by Peercoin, the first PoS currency, as well as scores of others currently in production, and is also the basis for several protocols from the research community [6, 11, 2, 4]. Because of the similarities to Bitcoin, chain-based PoS cryptocurrencies typically fork the Bitcoin codebase or some descendent thereof.

Most analysis of chain-based PoS has focused on consensus, aiming to show that properties like chain quality and chain growth are ensured in the same way as in Bitcoin (i.e., they hold when 51% of the stakeholders follow the protocol).

However, proof-of-work in Bitcoin serves a second purpose, which is to guard access to limited resources, such as its disk, bandwidth, memory, CPU. Proof-of-work is easy to check, but expensive to create, and so Bitcoin uses this as the first line of defense against junk data sent from untrusted network peers: First check the proof of work, then check everything else. This is similar to earlier (pre-Bitcoin) uses of proof-of-work, such as preventing spam and guarding access to server resources [10, 5, 7]. Recent versions of Bitcoin build further on this approach, transmitting separate data structures containing just the proofs-of-work ("headers") ahead of the actual payload ("blocks").

Unfortunately this idea from Bitcoin does not carry over properly into Proof-of-Stake. In particular, since the stake in a PoS block is found in the second transaction (the "coinstake" transaction) rather than in the header, headers-first processing is prone to attack. To explore the consequences of this insight, we examined the leading chain-based proof-of-stake cryptocurrencies, and found that five are vulnerable to a resource exhaustion attack. Roughly the attack involves sending malformed chains of invalid blocks or headers that are stored in RAM or on disk without being properly validated. We implemented and benchmarked these attacks and have begun a responsible disclosure of this issue.

2 Background

2.1 Proof of Stake

Chain-based proof-of-stake protocols can be defined in terms of two functions, a mining function M and a validation function V [4]. The validation function V takes as input a chain of blocks, and outputs 1 if and only if the chain is valid according to the application-specific rules of the cryptocurrency (i.e., all of the transaction semantics and valid proofs-of-stake). The mining function M takes as input a previous block B to build on, a coin C, and timestamp t and outputs a new block B' if it is indeed possible to mine a block, otherwise it outputs nothing. Similar to Bitcoin, nodes following the protocol attempt to extend the longest valid chain they know of.

As in Proof of Work, the mining function M involves comparing a hash of block data to a difficulty target. Instead of hashing the entire block, PoS introduces the notion of kernel hash, which depends mainly on the first transaction in the block, called the coinstake transaction. In more detail, the coin C spent by the coinstake transaction is hashed alongside the block's timestamp, the kernel hash of the previous block, and a few other metadata. Finally, the difficulty is weighted against the quantity of coins in C. Roughly speaking, those with more coins are proportionally more likely to be eligible to mine the next block.

2.2 Validating PoS blockchains

To mine on the largest valid block chain, as described above, a node must determine whether blocks received from peers are valid according to V. This can be expensive — in particular, it requires checking every transaction comes with correct signatures and does not double-spend any coins. Bitcoin, as well as Qtum and other PoS cryptocurrencies derived from it, have fairly complex machinery to perform this task efficiently. We give a bit more detail on how this works, focusing on the details relevant to our attack.

First of all, the node keeps track of the state of the current best chain, as well as a lookup table **pcoinsTip** of the Unspent Transaction Outputs (UTXOs) available in this chain. The **ConnectBlock()** method appends a block to the current main chain, validating each new transaction with the help of pcoinsTip.

The node maintains an in-RAM data structure **mapBlockIndex**, which represents a tree of every (valid) block header received, including the current longest chain as well as any forks. The **AcceptBlockHeader()** method performs simple checks before storing an entry in mapBlockIndex. We note here that in Bitcoin, this method checks the Proof-of-Work contained in the header, but since the coinstake transaction is not contained in the header, the analogous checking of Proof-of-Stake does not occur.

Blocks are stored on disk in append-only **block files**. Before storing on disk, the **AcceptBlock()** method first invokes AcceptBlockHeader(), and then hashes the transactions to check they match the *hashMerkleRoot* of the header. However, the transactions themselves are not checked until later.

If a chain of accepted blocks grows longer than the current main chain, then it is necessary to perform a "reorg" (Figure 1 A). This method unwinds the *pcoinsTip*, disconnecting the blocks one at a time down to the fork point, and finally connecting (and validating) the blocks in the new chain one at a time.

3 Explanation of the attack

We describe our attack scenarios from the viewpoint of an attacker node that has already formed a connection to the victim node.

Attack on RAM. We first describe the variation of our attack targeting RAM. The goal is to create fake block headers that pass *AcceptBlockHeaders()* so that the victim stores them in *mapBlockIndex.* The attack begins by picking an arbitrary fork point in the blockchain, and constructing a header that extends this block, as illustrated in Figure 1 B. Each header's *nTime* field must be strictly greater than its parent. The *hashMerkleRoot*, which ordinarily would commit to a batch of new transactions, is instead set to a garbage value. To optimize the attack, a single headers message contains a chain of the maximum number of 500 headers. To avoid being disconnected, the attacker ensures that the chain of bogus headers is strictly shorter than the current main chain; otherwise, the victim would request the corresponding blocks, disconnecting the peer after a timeout if they are not received. Detailed pseudocode is given in Algorithm 1.

Attack on Disk. Next we describe a variation of the attack that targets disk. The goal is to create a chain of fake blocks that pass *AcceptBlock()* so that the victim stores them in *mapBlockIndex* and in the blocks database. The attacker first creates a chain of blocks, starting from an arbitrary fork point in history, and reaching exactly up to the length of the main chain. Each block is filled to the maximum size with dummy transactions. The transactions have arbitrary values for their signatures and references to input transactions, as neither of these are checked during *AcceptBlock()*. Before broadcasting the blocks, the attacker first broadcasts just the headers. The reason for this is that *AcceptBlock()* discards a block unless it has been explicitly requested; by sending the headers first, once the headers chain in *mapBlockIndex* reaches the same height as the main chain, the victim requests all the blocks. Notice that full block validation is only triggered in case of reorg as shown in Figure 1. Since the attacker only sends blocks that do not exceed the length of the current chain, the victim never reorgs or validates the block. The pseudo-code for attack on disk is described in Algorithm 2.

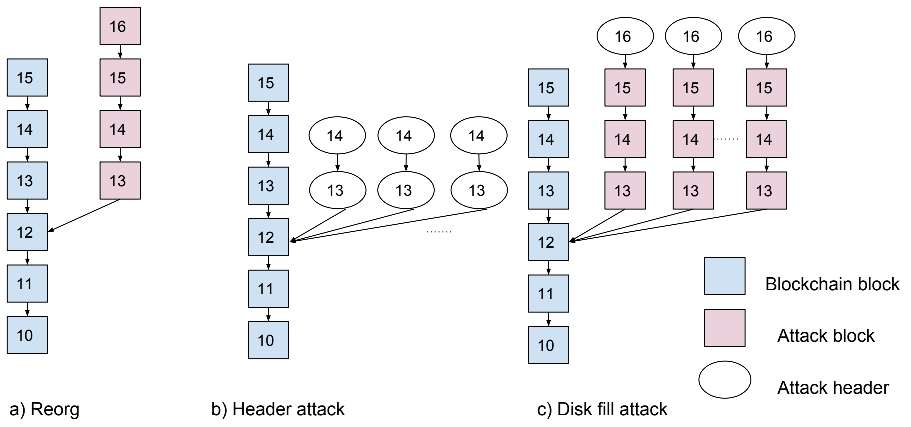

Fig. 1. Resource Exhaustion Attack

4 Evaluating the attack

4.1 Analysis of affected cryptocurrencies

To determine the impact of this vulnerability on the ecosystem, we collected a list of known cryptocurrencies from coinmarketcap.com (on Aug. 9th 2018), sorted by market cap, and filtered by chain-based PoS consensus type. We only looked at cryptocurrencies whose codebase was forked from (a descendent of) Bitcoin, i.e. in C++. We also omitted cryptocurrencies with smaller than $10M marketcap. In total we examined 26 cryptocurrencies. Next we inspected the source code and determined whether each had support for headers-based block downloads (i.e.,

Algorithm 1 RAM attack nothing at stake

```
 1: procedure RAM ATTACK(target_peer)
 2:     block ← empty
 3:     blockcount ← getblockcount()
 4:     depth ← rand(1, MAX_HEADERS_DEPTH)
 5:     pastblock_header ← getblockheader(blockcount - depth)
 6:     nTime ← pastblock_header.nTime
 7:     while target_peers.alive() do:
 8:         prevhash ← pastblock_header.hash
 9:         for d in depth do:
10:             nTime ← nTime + block_interval*d
11:             nVersion ← CURRENT_BLOCK_VERSION
12:             nBits ← get_next_difficulty_bits()
13:             merklehash, nonce ← rand32_bytes(), rand4_bytes()
14:             block_header ← block_header(nVersion, prevhash, merkleroot, nTime, nBits, nonce)
15:             send_msg_header(target_peer, block_header)
16:             prevhash ← block_header.hash
```

Algorithm 2 Disk attack nothing at stake

```
 1: procedure DISK ATTACK(target_peer)
 2:     block ← empty
 3:     blockcount ← getblockcount()
 4:     depth ← rand(1, MAX_HEADERS_DEPTH)
 5:     pastblock_header ← getblockheader(blockcount - depth)
 6:     nTime ← pastblock_header.nTime
 7:     while target_peer.alive() do:
 8:         prevhash ← pastblock_header.hash
 9:         headers, blocks ← [],[]
10:         for d in range(depth) do:
11:             nTime ← nTime + block_interval*d
12:             nVersion ← CURRENT_BLOCK_VERSION
13:             nBits ← get_next_difficulty_bits()
14:             merklehash, nonce ← rand32_bytes(), rand4_bytes()
15:             block ← block_header(nVersion, prevhash, merkleroot, nTime, nBits, nonce)
16:             for j in range(MAX_TX) do:
17:                 prev_tx, prev_index ← rand32_bytes(), 0
18:                 scriptPubKey, amount ← b" ", rand_amount
19:                 tx ← create_transaction(prev_tx, prev_index, scriptPubKey, amount)
20:                 block.append(tx)
21:             block.rehash()
22:             prevhash ← block.hash
23:             headers.append(block.header)
24:             blocks.append(block)
25:             send_msg_headers(target_peer, headers)        ▷ Wait for peer to request
    blocks
26:             send_msg_blocks(target_peer, blocks[-1])
```

Table 1. Vulnerability analysis of chain-based PoS cryptocurrencies

Name	Market Cap (USD)	Vulnerable to RAM Attack	Vulnerable to Disk Attack	Check TxDB for Coinstake	Coordinated Disclosure	Security Process
Qtum (QTUM)	952,265,768	√	√	×	√	×
Emercoin (EMC)	110,386,208	√	√	×	√	×
Particl (PART)	47,065,433	√	√	×	√	×
NavCoin (NAV)	39,029,633	√	√	×	√	√
HTMLCOIN (HTML)	25,447,981	√	√	×	√	×
header disabled (PIVX etc.)	239,172,527	×	×	√	N/A	N/A
no header (PPC etc.)	736,472,358	×	×	√	N/A	N/A

if the AcceptBlockHeaders() method was present). By inspection, we estimated that the first 5 coins in Table 1 would be vulnerable to these attacks.

The second-from-bottom row of Table 1 is an aggregate (combined market cap) of 7 cryptocurrencies that include AcceptBlockHeader() but have disabled its functionality (i.e., do not process "headers" network messages). Likewise the bottom row is an aggregate of 14 cryptocurrencies that do not implement AcceptBlockHeader() at all.

To confirm the vulnerability, we next implemented the attacks in each of the codebases. We made use of existing test suites in Bitcoin software, specifically the "regtest" mode, which enables simulated timestamps and easy-to-create blocks, and a Python-based test node that can be extended with attacker behavior. We used Docker containers to package these tests, their dependencies, and the specific commit hash affected, into a reproducibility kit that we could easily share with developer teams as part of a vulnerability disclosure.

4.2 Benchmarking the attack

To verify that the attack works in a live network setting (and not only in the regtest mode), we conducted an attack against our own node running on Qtum's live network. We optimized our attack to benchmark how effectively it could be carried out in practice, including forming up to 10 multiple connections to the victim, and by generating the block/header payloads in a pipelined fashion while transmitting them over the connections. Our victim node had a download speed of 1825.35 Mbit/s while our attacker node had an upload speed of 49.27 Mbit/s. We were able to fill the victim's disk at a rate of 6.05 MB/s, or the victim's RAM at 2.52 MB/s. For the disk attack, the main bottleneck in our testing was the amount of bandwidth between the attacker and victim. However for the headers attack, we reached a bottleneck of computational overhead as the victim processes the headers message.

5 Coordinated Vulnerability Disclosure

Resource exhaustion attacks in cryptocurrencies are considered critical vulnerabilities.[1] Because of the ease of exploiting this, we initiated a coordinated disclosure process to give developers of all the affected codebases an opportunity to deploy mitigations.

As highlighted by a recent vulnerability affecting Bitcoin and their rival Bitcoin Cash,[2] there are not yet clearly established guidelines for disclosures involving multiple cryptocurrencies. Cryptocurrencies are decentralized, and in principle may have no one officially recognized development team [1]. However, in this instance, all five of the affected projects are clearly associated with an official website and have publicly listed contact information. We note that only one of the projects (NavCoin) has a published vulnerability disclosure process and dedicated security contact.

Cryptocurrency communities have at times been embroiled in bitter disputes. We considered that disclosing the vulnerability too widely could increase the risk that one may leak it or attack another. We note that Qtum's market cap is around 5x larger than the rest combined. However, as all five responded to our initial email, and all codebases had active development (commits on GitHub within the past week), we decided to communicate simultaneously to them all.

6 Mitigations

We propose some easy to implement mitigations for the affected currencies. We can Checkpoint every K blocks so that the node does not accept forking blocks more than K blocks deep. Another such mitigation might include Disabling headers support and use TxDb check to determine if they have seen the coinstake transaction. This mitigation, although not perfect atleast requires the adversary to have some stake in the past. Lastly, we propose to UTXO snapshot every K blocks and perform validation of all blocks by rolling the pcoinsTip struct from the closest snaphot to the fork point.

7 Discussion and conclusion

We show a resource exhaustion attack that can be carried out by a malicious peer without any stake in the currency and without any privileged network position. We found only a small number of the seventy chain-based PoS cryptocurrencies we considered to be vulnerable; however, weighted by market cap, this is more

[1] https://en.bitcoin.it/wiki/Common_Vulnerabilities_and_Exposures
[2] https://bitcoincore.org/en/2018/09/20/notice/

than half. The affected projects were all forked from a relatively recent Bitcoin version (version 0.10.0 or later, released February 2015) that incorporates the "headers first" feature, while those based on earlier versions of Bitcoin code are not vulnerable. We suspect the most likely outcome of our report is that the affected cryptocurrencies will downgrade to adopt behavior like the others.

However, we observe that even the non-vulnerable cryptocurrencies do not correctly implement the idealized protocol described in the research literature [6, 11, 2, 4]. In particular, the coinstake transactions in accepted blocks are validated against the coins database associated with the current main chain — even if the new block in question is on a fork from the main chain. In other words, the validation function V_p is not applied deterministically, but instead approximated. We plan to explore the consequences of this in future work.

The insights behind our attack are related to, but distinct from, the "nothing-at-stake" problem. This refers to the observation that in chain-based PoS, stakeholders are not penalized for mining on blocks on a conflicting fork. Our resource exhaustion attack is different in that the attacker need not have ever been a stakeholder at all. However, both cases highlight the difficulties in adapting designs ideas from proof-of-work into the proof-of-stake setting.

References

1. Azouvi, S., Maller, M., Meiklejohn, S.: Egalitarian society or benevolent dictatorship: The state of cryptocurrency governance. In: 22nd International Conference on Financial Cryptography and Data Security (2018)
2. Bentov, I., Gabizon, A., Mizrahi, A.: Cryptocurrencies without proof of work. In: International Conference on Financial Cryptography and Data Security. pp. 142–157. Springer (2016)
3. Böhme, R., Christin, N., Edelman, B., Moore, T.: Bitcoin: Economics, technology, and governance. Journal of Economic Perspectives **29**(2), 213–38 (2015)
4. Brown-Cohen, J., Narayanan, A., Psomas, C.A., Weinberg, S.M.: Formal barriers to longest-chain proof-of-stake protocols. arXiv preprint arXiv:1809.06528 (2018)
5. Dwork, C., Naor, M.: Pricing via processing or combatting junk mail. In: Annual International Cryptology Conference. pp. 139–147. Springer (1992)
6. Fan, L., Zhou, H.S.: A scalable proof-of-stake blockchain in the open setting (or, how to mimic nakamoto's design via proof-of-stake). Cryptology ePrint Archive, Report 2017/656 (2017), https://eprint.iacr.org/2017/656
7. Juels, A., Brainard, J.G.: Client puzzles: A cryptographic countermeasure against connection depletion attacks. In: NDSS. vol. 99, pp. 151–165 (1999)
8. Narayanan, A., Bonneau, J., Felten, E., Miller, A., Goldfeder, S.: Bitcoin and cryptocurrency technologies: a comprehensive introduction. Princeton University Press (2016)
9. O'Dwyer, K., Malone, D.: Bitcoin mining and its energy footprint. In: IET Conference Proceedings. The Institution of Engineering & Technology (2014)
10. Parno, B., Wendlandt, D., Shi, E., Perrig, A., Maggs, B., Hu, Y.C.: Portcullis: protecting connection setup from denial-of-capability attacks. ACM SIGCOMM Computer Communication Review **37**(4), 289–300 (2007)
11. Pass, R., Shi, E.: Fruitchains: A fair blockchain. In: Proceedings of the ACM Symposium on Principles of Distributed Computing. pp. 315–324. ACM (2017)