CVE-2018-17145: Bitcoin Inventory Out-of-Memory Denial-of-Service Attack

Braydon Fuller and Javed Khan

September 9th, 2020

Abstract

This paper describes an easily exploitable uncontrolled memory resource consumption denial-of-service vulnerability that existed in the peer-to-peer network code of three implementations of Bitcoin and other blockchains including Litecoin, Namecoin and Decred.

1 Attack Overview

There was an uncontrolled resource consumption and out-of-memory (OOM) vulnerability that could have been easily exploited in a denial-of-service (DoS/DDoS) attack against many Bitcoin, Litecoin, Namecoin and Decred nodes by *any* other network participant. The vulnerable versions include: Bitcoin Core v0.16.0, Bitcoin Core v0.16.1, Bitcoin Knots v0.16.0, all beta versions of Bcoin up to v1.0.0-pre, all versions of Btcd up to v0.20.1-beta, Litecoin Core v0.16.0, Namecoin Core v0.16.1, and all versions of Dcrd up to v1.5.1.

The vulnerability has been patched in Bitcoin Core v0.16.2+, Bitcoin Knots v0.16.2+, Bcoin v1.0.2+, Btcd v0.21.0-beta+, Litecoin Core v0.16.2+, Namecoin v0.16.2+, and Dcrd v1.5.2+ releases. The issue does not affect earlier versions of Bitcoin Core and derivatives including v0.15 and earlier.

The vulnerability was discovered on Friday, June 22nd, 2018 by Braydon Fuller of the Bcoin protocol team at Purse. At the time of the discovery this represented more than 50% of publicly advertised Bitcoin nodes with inbound traffic [1], and likely a majority of miners and exchanges. The vulnerability in Bitcoin Core was introduced on November 15th, 2017 in pull request #10286 of Bitcoin Core and released in versions v0.16.0 and v0.16.1 of Bitcoin Core. Disclosure was made on Monday, July 9th 2018 to Bitcoin Core and Litecoin Core maintainers, and covertly patched on the following day in pull request #13622. Many forks of Bitcoin Core are *not* affected as they were forked earlier or haven't incorporated the updates. Projects such as Zcash, Bitcoin ABC, Bitcoin Gold, Bitcoin Unlimited, Bitcoin XT *did not* include the vulnerable code at the time of discovery.

On Friday, June 26th, 2020 it was discovered that the vulnerability also existed in Btcd by Javed Khan. With additional research it was also found to

affect Dcrd on Tuesday, July 7th, 2020. The vulnerability was reported to the Decred Bug Bounty program. At the time of discovery this represented 100% of the Decred nodes and 100% of nodes serving compact block filters to Bitcoin Lightning wallets.

It could be possible to escalate the severity of the denial-of-service vulnerability to contribute to a loss of funds or revenue. This could be through a loss of mining time or expenditure of electricity by shutting down nodes and delaying blocks or causing the network to temporarily partition. It could also be through disruption and delay of time sensitive contracts or prohibiting economic activity. That could affect commerce, exchanges, atomic swaps, escrows and lightning network HTLC payment channels. There has not been a known exploitation of this vulnerability in the wild.

2 Timeline

- `2018-Jun-22` - The DoS attack was discovered in Bcoin [2] by Braydon Fuller. It was discovered during code review of how the code would handle a large number of inventory messages and items. The Bcoin maintainers, including Christopher Jeffrey and Javed Khan, were securely notified via PGP email of the vulnerability with a patch included.

- `2018-Jul-02` - Before pushing fixes to Bcoin other implementations were tested for the attack and it was discovered to also DoS the latest Bitcoin Core [3] release (v0.16.1) and development branch.

- `2018-Jul-02 to 2018-Jul-08` - The attack was researched further by Braydon Fuller to evaluate the vulnerability in other versions and forks of Bitcoin Core. In this process it was discovered to only affect v0.16.1 and v0.16.0 of Bitcoin Core and Litecoin Core v0.16.0 and *not* previous versions of Bitcoin Core and Litecoin Core [4] or other forks such as Zcash or Bitcoin Cash, many others were tested and reviewed.

- `2018-Jul-09` - The description, patch and proof-of-concept of the vulnerability were timestamped in a Bitcoin block at height 531,241 in a transaction with txid:

  658d467097dc64e4e093d5bcb082139f34f570fe20bb71b57f098f5d6ebc3ce8

- `2018-Jul-09` - The description, patch and remedies were securely disclosed to Bitcoin Core and Litecoin Core maintainers, and later to several exchanges and miners.

- `2018-Jul-10` - Bcoin v1.0.0 was released with the fix, and the vulnerability was fixed in the master branch of Bitcoin Core from pull request #13622 by Matt Corallo.

- `2018-Jul-16 to 2018-Jul-17` - The patched version of Bitcoin Core v0.16.2 was tagged as release candidates rc1 and rc2.

- `2018-Jul-23` - Litecoin Core v0.16.2 was tagged as release candidate rc1 with the patch.
- `2018-Jul-29` - Bitcoin Core v0.16.2 was released with builds.
- `2018-Jul-30` - Bitcoin Knots v0.16.2 was tagged and included the patch.
- `2018-Aug-06` - Namecoin Core 0.16.2 was tagged and included the patch.
- `2018-Sep-08` - Litecoin Core v0.16.2 builds were released.
- `2018-Sep-18` - A CVE ID was requested from https://cve.mitre.org by Braydon Fuller.
- `2018-Sep-19` - The ID CVE-2018-17145 was given for the vulnerability.
- `2018-Sep-20` - A critical inflation vulnerability, CVE-2018-17144, was disclosed to the public. The vulnerable versions overlapped with releases also vulnerable to CVE-2018-17145. Many nodes upgraded during this period greatly reducing the number of nodes vulnerable to CVE-2018-17145.
- `2020-Jun-26` - Javed Khan in revisiting the attack discovered that it also applied to Btcd [5], started further research and notified Braydon Fuller.
- `2020-Jul-01` - Btcd maintainer Olaoluwa Osuntokun was securely notified via PGP email with details of vulnerability.
- `2020-Jul-03` - Btcd maintainer John C. Vernaleo and Olaoluwa Osuntokun were securely notified via PGP email with a patch to fix the vulnerability from Javed Khan.
- `2020-Jul-07` - Javed Khan opens pull request #1599 to Btcd. During further research discovers that a variant of the vulnerability also affects Dcrd [6] and reports the vulnerability to Decred Bug Bounty program.
- `2020-Jul-08` - Btcd pull request #1599 is merged and patched for the vulnerability, see commit: 875b51c9fb83e5521ef5b4f8c9c522126baa2041.
- `2020-Jul-08` - Dcrd is patched in pull request #2253.
- `2020-Aug-07` - Decred Journal (July 2020) describes the behavior of the vulnerability in reference to development for the vulnerability patch.
- `2020-Aug-27` - Dcrd v1.5.2 is released with the vulnerability patched.
- `2020-Aug-27` - Btcd v0.21.0-beta is released with the vulnerability patched.

3 Attack Technical Details

The attack can be performed by a peer rapidly sending multiple transaction `inv` messages with random hashes, one below the max at 49,999 items and never sending the corresponding `tx` data.

The attack can be accelerated with multiple peers, escalating into a DDoS, and is limited by the bandwidth capabilities to the target nodes from attacker nodes. With a 1Gbps (125 MB/s) connection it would be possible to send around 83 inv messages with 49,999 items per second, giving a maximum rate of 4,166,584 inv items per second. Memory will grow as fast as it's possible to send data to the node, until it crashes or locks up the machine in swap disk usage in several minutes.

3.1 Bcoin Details

The issue was that a `Map` used for tracking inventory from peers could grow in size without an upper-bound, aside from a time based limit that would flush the `txMap` if a peer is stalling and not responding. If enough inventory hashes can be sent before that stall timeout check, the memory can grow past memory space and crash the process.

The p2p network messages could grow the memory of `txMap` without limit via this exposed code path `handlePacket -> handleInv -> handleTXInv -> ensureTX -> getTX`:

```
getTX(peer, hashes) {
  ...
  for (const hash of hashes) {
    if (this.txMap.has(hash))
      continue;

    this.txMap.add(hash);
    peer.txMap.set(hash, now);
    ...
  }
  ...
}
```

The issue was resolved by adding a limit to the size of the peer `txMap` and removing peers that exceed the limits. This was fixed by Christopher Jeffrey in commit: 05c38853d7f50fb4ad87e28fa7b46017f78e2955

3.2 Bitcoin Core Details

The issue in Bitcoin Core was more hidden as there appears to be code that limits the growth of a similar map, see `src/net_processing.cpp` in `ProcessMessage`:

```
if (vInv.size() > MAX_INV_SZ)
{
```

```
  LOCK(cs_main);
  Misbehaving(pfrom->GetId(), 20,
              strprintf("message inv size() = %u",
              vInv.size()));
  return false;
}
```

However the actual behavior of the program was different and behaved as if there *wasn't* a limit and the memory would endlessly grow.

The issue was within an easy to overlook and unguarded call to `GetMainSignals().Inventory(inv.hash)` in `ProcessMessage` of `src/net_processing.cpp`, that would grow the size of `m_callbacks_pending` at a faster rate than it is processed in `SingleThreadedSchedulerClient` that is used for tracking events. This is the code at the source of the vulnerability, see `src/validationinterface.cpp`:

```
void CMainSignals::Inventory(const uint256 &hash) {
  m_internals->m_schedulerClient.AddToProcessQueue([hash, this] {
    m_internals->Inventory(hash);
  });
}
```

The issue was resolved by removing the vulnerable code as it was no-longer needed in pull request #13622 by Matt Corallo, see details in commit: beef7ec4be725beea870a2da510d2817487601ec

3.3 Btcd Details

The issue in Btcd takes more time to exploit compared to Bitcoin Core and Bcoin and can be under 5 minutes to over 10 minutes from a single peer.

Btcd stores the received transaction inventory items in `peer.knownInventory` (`peer/peer.go`) which is an MRU map (`mruInventoryMap`) that is supposed to be limited by size (e.g. 1000). Unfortunately, the instance of `peer.knownInventory` uses the MRU map in a way that exposes a bug which makes the map vulnerable to memory leaks.

In detail, `mruInventoryMap.Add` takes a pointer to `wire.InvVect` which is later modified by `SyncManager.handleInvMsg`. In `netsync/manager.go`:

```
if peer.IsWitnessEnabled() {
    iv.Type = wire.InvTypeWitnessTx
}
```

When `mruInventory.Add` tries to evict the LRU item from `invList`, it no longer matches the `invMap` key, since the object has been changed. The eviction will always fail since `lru.Type != iv.Type`. Because of this the memory usage keeps climbing. In `peer/mruinvmap.go`:

```
if uint(len(m.invMap))+1 > m.limit {
    node := m.invList.Back()
```

```
    lru := node.Value.(*wire.InvVect)

    // Evict least recently used item.
    delete(m.invMap, *lru)
    ...
}
```

In addition to `mruInventoryMap`, two other maps store received inventory hashes as keys for `peerSyncState`: `requestedTxns` and `requestedBlocks`. These maps should be bounded to prevent memory leaks.

The issue in Btcd was resolved by switching to a fixed LRU implementation and switching the usage of the maps to use the updated version. This was completed in pull request #1599 by Javed Khan on July 7th, see commit: 4b3f7f3c7a490151801c0aaf117befeae1c6bc1b.

3.4 Dcrd Details

Similar to the Btcd vulnerability, however it relates specifically with a subset of the two unbounded maps in `blockmanager.go`.

Both `peerSyncState` properties `requestedTxns` and `requestedBlocks` are unbounded and only cleared when the corresponding inventory item is resolved. If the remote peer returns `notfound` for the inventory items, the hash is *not* cleared from the maps' keys. A malicious peer may spam non-existent inventory items until the host runs out of memory.

Stall detection is *not* triggered in this case because the malicious peer never stalls, as every `getblocks` is returned an `inv` and every `getdata` is returned a `notfound`.

A very straightforward fix is to simply bound the maps `requestedTxns` and `requestedBlocks` in `blockmanager.go`. This issue in Dcrd has been resolved in pull request #2253 by David Hill on July 8th, 2020, see commits:

- 0b3133113523972c3c2cc080058b2933b53e2609
- ef1530cfbc42e1b9583241923de16a2603bf5cdc
- 6187eebff62006c4bd71866391603e37a7e1654b
- 347b0b9376a8ae68af75b890cfb162131cea568a

4 Acknowledgments

- Braydon Fuller for finding the vulnerability in Bcoin and Bitcoin Core and doing the initial research.
- Matt Corallo, Wladimir J. van der Laan and maintainers for patching the vulnerability in Bitcoin Core.
- Christopher Jeffrey for patching the vulnerability in Bcoin.

- Purse team: Andrew Lee and Buck Perley for assisting in disclosure.
- Luke Dashjr for patching Bitcoin Knots.
- Adrian Gallagher and maintainers for patching the vulnerability in Litecoin Core.
- Jeremy Rand, Daniel Kraft and maintainers for patching Namecoin Core.
- Javed Khan for researching and patching the vulnerability in Btcd and Dcrd.
- Thejaswi Puthraya for helping debug the vulnerability in Btcd and Dcrd.
- John C. Vernaleo and maintainers for patching Btcd.
- David Hill, Dave Collins and maintainers for patching Dcrd.

5 References

1. `https://bitnodes.io`
2. `https://bcoin.io`
3. `https://bitcoincore.org`
4. `https://litecoin.org`
5. `https://github.com/btcsuite/btcd`
6. `https://decred.org`