P2P Storms Vulnerability Report

By **Jacob Gadikian, Joe Bowman, Sheldon Dears, Robin Tunley, Clemens Scarpatetti**

October 8 2023

contact@notional.ventures

notional.ventures

github.com/notional-labs

Introduction

Notional security team is in the process of studying and reproducing an attack across an array of Cosmos testnets. These attacks have caused varying degrees of network distress, but the worst ones have effectively halted chains and reduced block production time by factors ranging from 6-80. A variety of weaknesses across the Cosmos stack contribute to the effectiveness of this attack, but we believe it is possible for an attacker to bring a Cosmos chain down for multiple days in which time they could leverage social media to influence prices and perform an economic attack.

While the issues that contribute to these network conditions are multifaceted and spread across the interchain stack, the attack is relatively simple to execute. It relies on filling the validator mempools, which causes network-wide congestion and performance degradation. This report will outline a brief history leading to this report, how it can be executed, the on-chain results, and our analysis of the underlying issues along with some recommendations on how they might be mediated.

History of Events

In this section we outline a short history of events that have occurred, primarily on Cosmos mainnets, whose identification have led us to our current investigation.

June 2020: Game of Zones testnet

Zaki Manian reports observing stressed network conditions during this incentivized testnet competition that are similar to conditions involved in future events and our testnet attacks.

November 2021: Sentinel Attack

In November 2021, the Sentinel blockchain experienced a validator distress attack with the following symptoms [1]:

- Unusually high bandwidth consumption
- Validators missing an uncharacteristic number of blocks
- Validators proposing invalid blocks with increased frequency

The source of the congestion issues seems to be rooted in the propagation of blocks containing transactions with the ClientUpdate [2] message within them. This resulted in blocks that were between 10 and 30 times larger (Mb as opposed to 10's of kb) than typical blocks. Jacob and the Notional team at that time did some investigating of this issue and wrote up a brief report on Github.

This incident was reported to HackerOne [3]. The response time using HackerOne was 18 days and the issue was eventually dismissed by the Informal Team. Notional received no compensation for this investigation.

May 2022: Luna Attack

During the infamous depeg of UST and the crash of the Terra blockchain, a similar set of symptoms appeared in the Terra network. In this instance, a substantial number of the Terra validators began missing an unusually high number of blocks, as can be seen in a writeup that Rarma produced [4]. The source of this paralysis seems to be that an unsually large number of small transactions flooded the network, preventing users from moving their funds and "causing a series of cascading failures".

The transactions appear to have been using the memo field as a way to coordinate blocktimes, which was also described in Rarma's article. Panic as UST depegged and LUNA's value fell added to the network instability. A summary [5] was written by Jacob and the Notional team.

August 2023: Stride

In August 2023, Stride experienced a similar attack. This time, another team was actually able to record some of the network activity in the image below:

On the left hand side in the image above, you can see the number of blockparts being gossiped through the network is orders of magnitude higher than usual. As the attack progresses, you can see the network eventually return to equilibrium, before becoming stressed again towards the far right-hand side.

September 2023 - Present: Testnet Attacks

As of late September, the Notional security team has been experimenting with ways to reproduce these network conditions using a several different attack methods. A full detailed list of these attacks is included at the end of this report, including analysis of relevant network metrics.

Execution

The attack is executed with a reasonably simple event flow. The attacker must control a moderate amount of capital, expressed in the native token of the chain they wish to attack. They will use the attack as a way to create panic around the health of the chain, and take short positions on the resulting price action.

The critical condition for this attack to be successful is that the mempool must be full. There are at least three primary methods to accomplish this:

Method 1: BananaKing

A small number of very large transactions can be broadcast to the network. An example of this can be seen in the transactions created by a user named BananaKing on the Osmosis Network [6]. This will be successful because IBC-go does not check length limits of receiver and memo fields. Notional has successfully executed attacks on testnets using this method.

Method 2: Goldilocks

An analysis performed ~110,000 blocks of the Osmosis blockchain [7] reveals a number of messages which are unusually large and can be leveraged to create network bloat. These messages can also contribute to unusually large blocks in a similar manner as in the BananaKing method, but a larger number of transactions are required. Some examples of larger-than-typical messages revealed in our analysis are:

- **/cosmwasm.wasm.v1.MsgStoreCode**: largest single message found within problematic transactions.

- **/cosmos.authz.v1beta1.MsgExec**: of particular concern since an arbitrarily large number of small messages can be batched together without transparency.

- **/ibc.core.client.v1.MsgUpdateClient:** can be batched with many MsgRecvPacket messages leading to unusually large transactions.

Method 3: Gatling Gun

A final method of filling the mempool of validators is to spam the network with a very large number of small transactions. This is the scenario that was present onchain during the Terra collapse, but to date Notional has not attempted to recreate this version of the attack.

In all cases, the transactions must have a high probability of being included in blocks. On most chains, the Goldilocks transactions and small normal transactions require gas, hence the need for initial capital. On the Cosmos Hub however, the ClientUpdate message has no gas cost. Other chains may have specific gas requirements for messages that may not have been considered here which make them more vulnerable to a particular attack method.

Once broadcast, these messages will get picked up by a proposer validator who will include them in valid blocks. Because of either the number or size of the messages, these blocks will be unusually large and when the proposer begins gossiping them to other validators through CometBFT, network performance begins to deteriorate. The primary effect of this

deterioration is a dramatic and extended reduction in blocktime, though other effects will be discussed in the Results section.

Finally, with the chain in a destabilized state, the attacker and their team can flood social media (Twitter, Telegram, Discord, etc) with panic regarding the degraded performance, with the hope of inducing downward price action. Any sudden volatility will only exacerbate the already stressed network as users begin trying to move funds. This incident can continue until a new majority of validators update to limit block size and resync states as the chain struggles to reclaim its peer-to-peer stability.

Notional has identified two distinct roles that an attacker can assume to execute any of the three attack methods.

Public RPC Endpoint

An attacker does not need access to a full node in order to run this attack. Since all nodes (including public RPCs) have mempools - a relic of the Satoshi model for nodes - an attacker can run a script like the one found in the ‘spammy’ repository [8] and flood the RPC mempool with any of the three previously mentioned transaction structures. In this role, the attacker will need to sign for the spam transactions (requring gas). Network deterioration will begin once these transaction begin to be gossiped.

Private Node

This attack can also be executed using a private RPC node that they control. In this role, the attack will not require a script like spammy. The attacker may prefill the node’s mempool with transaction matching one of the desired methods above and keep it from gossiping this info. Later, the configuration can be adjusted, forcing the stockpiled transactions to be released all at once.

It is important to note that while the symptoms of this attack resemble those typical of a DDOS attack (as has been noted by other teams), this is a mischaracterization. A true DDOS requires multiple nodes in a coordinated effort. But, as can be seen from the roles examined above, this attack requires far less - not even a full node - and thus is far more serious.

All versions of this attack cause predictable stress to the network under seige. Noteworthy on-chain effects of this stress include, but are not limited to:

- Unusually large block times
- Full mempool
- High rate of block proposal failures
- Nodes unable to assemble complete blocks from their parts
- Full blocks containing 336 block parts
- Block size of about 20 megabytes
- Bandwidth saturation of 1Gbps pipeline in both directions

Underlying Issues & Recommendations

The issues and recommended mitigations are split across a number of repositories and scopes.

ibc-go

1. IBC MsgTransfer messages are unbounded in size

IBC MsgTransfer messages do not impose a length limitation on either Memo or Receiver fields. I understand this was raised previously, and the original design decision here was that this should be the case (certainly the for memo field) due to the desire to not place unnecessary restrictions on future IBC integrations. That said, the resulting behavior here is that it is trivial to create transactions of many kilobytes, or indeed megabytes in size, at very low cost. Neither the sender field, nor the memo field invoke computation, so the gas fee for including this transaction in a block is a function solely of the `tx_size_cost_per_byte`, which is in all probability, currently set to an unacceptably low value (this itself, is under the scope of Cosmos-SDK (w.r.t to sane defaults), and individual networks' implementations).

This issue has been used in conjunction with the issues below to execute the aforementioned attack.

That said, solely resolving this issue in isolation, very much relies on developers in future not inadvertently introducing a message type with unbounded fields. The underlying stack should additionally include additional protections to appropriately handle such transactions should they be introduced.

cometBFT

2. Inconsistent and Ineffective transaction size restrictions

In relation to (1) above, it is worth noting that cometBFT includes a size check on transactions in two places:

- In CheckTx, the transaction is checked to assert that it does not exceed the local node's max_tx_bytes, but this value is only set locally so only helps in ensuring the local node does not add an excessively large transaction to it's own mempool. This feature is ineffective in stopping a) large txs being broadcast to a node via the mempool reactor, and b) large txs being included in a block by a node that does not set this parameter to a suitable value.
- In the mempool preFilter, the transaction is checked to assert that it does not exceed the consensus param max_bytes (block size), to check the transaction can fit inside a single block. For large block sizes, this is ineffective, and a global max_tx_size would be a useful addition for chains that wish to operate with larger block sizes, but not permit excessively large transactions.
- Notably, the both of these values use the raw value of MaxBytes, but should - for correctness - use types.MaxDataBytes() to ensure the total block size remains within

the correct bounds. Failure to do so allows a malicious actor to fill the mempool with transactions of size `types.MaxDataBytes > x > MaxBytes` which can never be delivered, as ReapMaxTxBytes() which is used when creating the proposal uses the correct value.

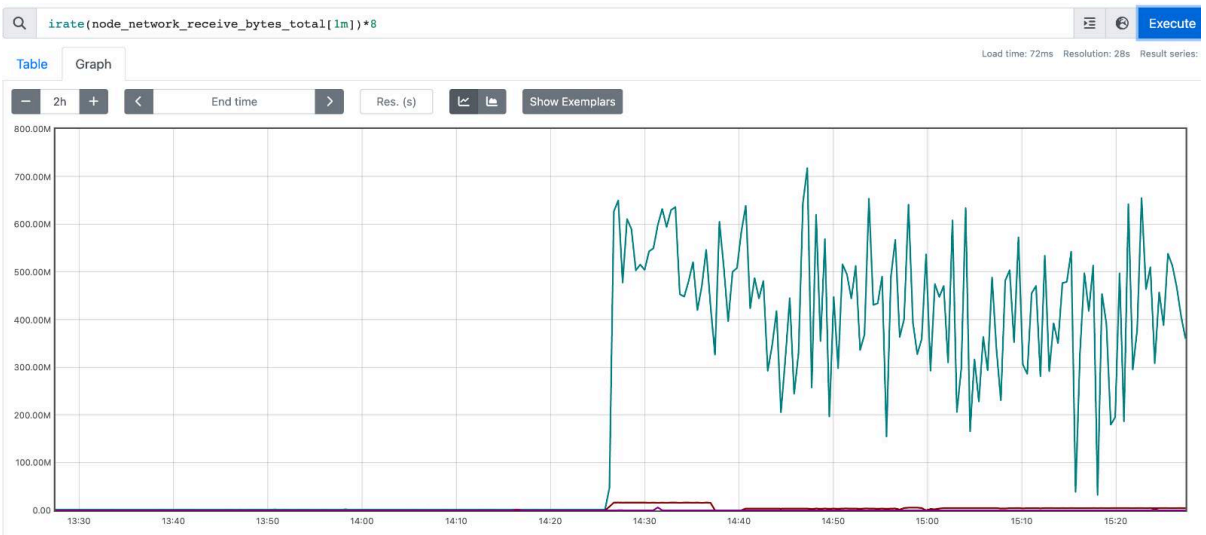
3. Default block sizes too high

This has already been covered in ASA-2023-002 and is included in here only for reference.

4. Highly inefficient P2P data propagation

CometBFT P2P networking uses an extremely naive algorithm for transmitting data. Every peer will receive a copy of the data. This naive routing algorithm ensures that well connected nodes both send and receive an inordinate amount of duplicate traffic. We found that during a period of heavy transaction load, a typical node with between 33-35 peers during testing, and otherwise using default CometBFT configuration values, can saturate a 1Gbps network pipe when the `max_bytes` (block size) is 22Mb in size.

Breaking this down further, we see that mempool tx gossip (chID 0x30) constitutes about 20% (56GBs over a 1 hour period; an average throughput of 133Mbps) of this traffic, and block data gossip (chID 0x21) about 80% (235GBs over a 1 hour period; average of 560Mbps).



With smaller block sizes, the situation is marginally improved. A 1Mb block size reduces data throughput under load to on channel 0x21; this is somewhat predictable due to decreased throughput in blocks. However, the reduced block data propagation throughput resulting from smaller block sizes permits mempool transaction gossip to increase to some 700Mbps of upstream capacity, while the mempool is at capacity.

Fundamentally, the existing P2P layer, and its defaults are problematic.

It does however appear that a mitigation to this is ensuring that block sizes (as per (2)), the number of peers, and P2P throttling values, are set to more appropriate values (data

suggests that a block size of 1MB -> 5MB is an acceptable range; number of peers should be maintained around the 30 mark, for a node with a 1Gbps connection, and

cosmos-sdk

5. Default tx_size_cost_per_byte is too low

As referred to in (1) above, many, if not all, networks use the default gas values for signature verification and tx_size_cost_per_byte. When transactions are unbounded in size, this default is unacceptably low. There is consensus between myself, Jacob, Zaki and Dev that this value should be at least 5x it's current value of 10 in order to appropriately price large transactions.
Cosmos Hub:

Observations

In addition to the major issues listed above, we provide here a brief summary of one particular observation we have made in the process of these tests.

Within the IBC scope, we noticed that the MsgClientUpdate transactions contain data that is not required by the ClientUpdate logic, either because it is entirely superfluous or simply inefficient. We suggest that eliminating (or more efficiently encoding) this data will make these types of tramnsactions smaller without affecting the overall functionality of ClientUpdate. Fields where we have noticed this include but are not limited to `signature/block_id_flag` and `signature/timestamp `.

Conclusion

On the current CHRS testnet there are 36 validators. The Cosmos Hub, in contrast, has 180 validators with significant geographic distribution. Given that this attack exploits the CometBFT gossip mechanism, we can expect that a similar attack performed on the Cosmos Hub would be significantly more detrimental.

It is however worth noting that existing testnets are not necessarily the best indicators of attack impact due to the deviation between testnets and mainnets in terms of size, config and topology.

Due to the pervasive nature of the issues across the Cosmos stack, all Cosmos chains are considered vulnerable to a halt or severely deprecated performance in the event a similar attack is deployed against them. But in particular, **any chain in which user funds depend on liveness** (leverage protocols, for example) **face even greater risk**.

It is the feeling of Notional that there is ample evidence to suggest that (1) this threat has been known for a considerable period of time and (2) despite our attempts to report it, no meaningful actions have been taken and (3) upon our successful reproduction on two separate testnets, our continued attempts to report the issues in alignment with its severity have been met with suspicion and hostility. We are still in the process of refining the attack, and continue to uncover underlying issues and problematic interactions. We will continue to report them as they emerge.

We have published a suite of patches to help mitigate these issues in our placid repository on Github.

References

[1]: Notional. n.d. “Attack on Sentinel.” Github. https://github.com/notional-labs/spammy/tree/main/sentinel-evidence.

[2]: Notional. n.d. “Sentinel Evidence.” Github. https://github.com/notional-labs/spammy/tree/main/sentinel-evidence.

[3]: Notional. n.d. “HackerOne Report: Sentinel Attack.” https://acrobat.adobe.com/link/review?uri=urn:aaid:scds:US:c14b2605-3497-381a-870e-510b92e33f75.

[4]: Rarma. n.d. “The Attack on Terra.” Medium. https://medium.com/@Rarma_/53cca9f849ac.

[5]: Notional. n.d. “WTF Happened to Terra.” Github. https://github.com/notional-labs/notional/blob/master/incidents/WTF%20HAPPENED%20TO%20TERRA.pdf

[6]: Osmosis. n.d. “BananaKing Transaction.” Mintscan. https://www.mintscan.io/osmosis/tx/D62F0F0354C4DEA0D9DFCA596D9BC3F2943DBA7D24818009DFD725F883088DD0

[7]: Notional. n.d. “Osmosis Block Analysis.” Github. https://github.com/notional-labs/spammy/blob/main/tests/analyze-tx-sizes/txDataAnalysis-osmosis1-50kblocks.json

[8]: Notional. n.d. “spammy.” Github. https://github.com/notional-labs/spammy/tree/main.

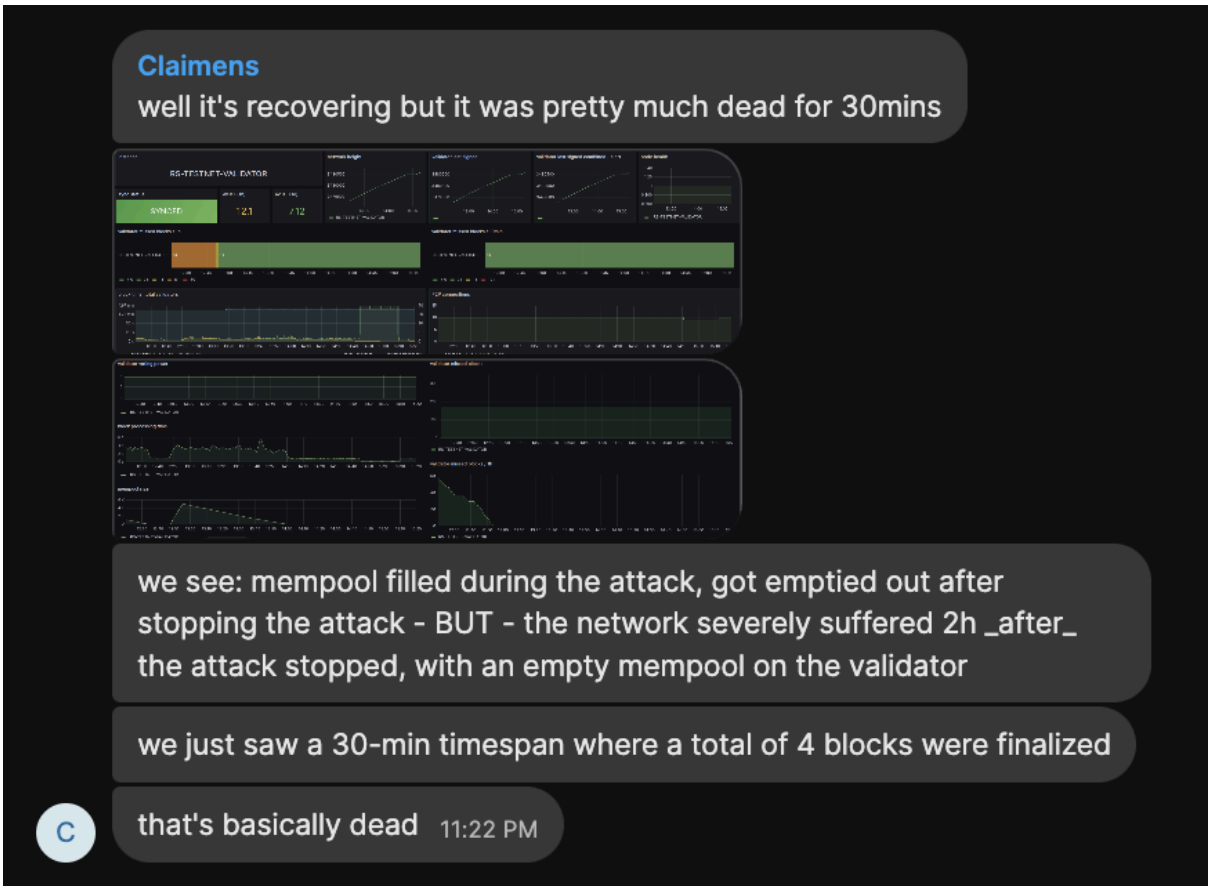
Appendix: Testnet Attack Details

Networks Tested

Many thanks to teams like Celestia, Hypha, Terraform Labs, and Injective who actively assisted us in our research

As of October 17, 2023, we continue to test the technique against networks in Cosmos exclusively with the team's explicit permission.

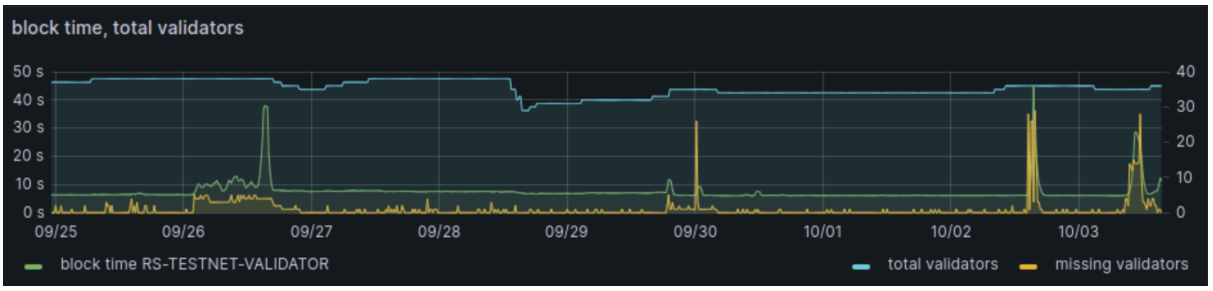
Sep 25 - Oct 3: Cosmos Hub Testnet



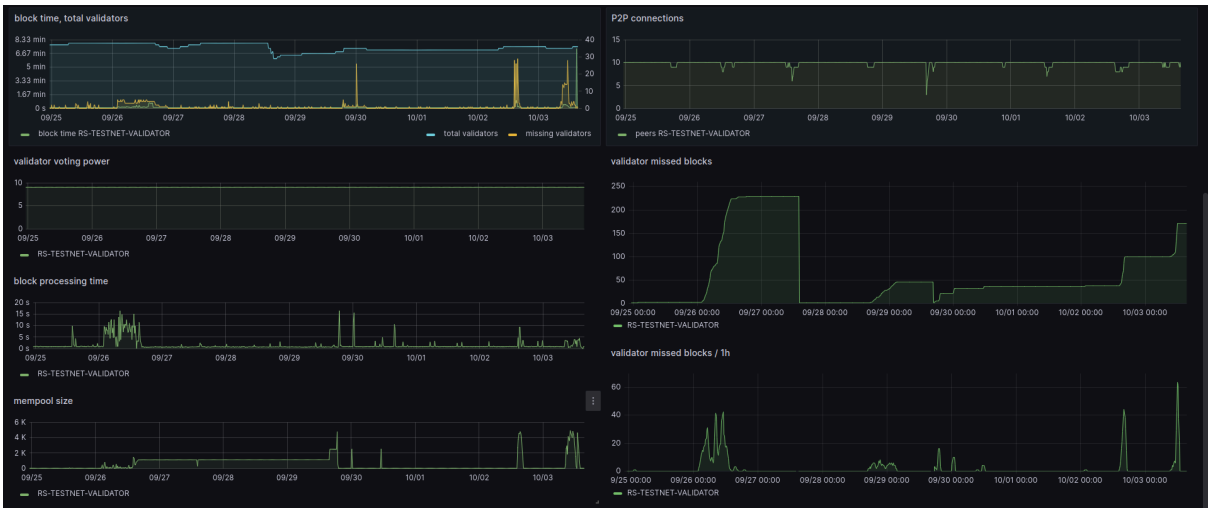
While our attempts to stress this network are on-going, in our initial batch of attacks Jacob and the Notional security team managed to successfully reproduce effects in the network activity that are consistent with the behaviour of past mainnet attacks.

With the assistance of Clemens Scarpatetti from CryptoCrew, we initiated an initial sequence of over 2 dozen BannaKing-style attacks over a course of days from September 25th to October 16th. Our initial attacks indicate that while the attack can be performed from both a node or a public RPC, the effects from the node were more severe. To establish a worst-case scenario, we continued using primarily the node attack role. We found evidence of all related symptoms in our monitoring of network statistics during the attack.

Peak depreciation was achieved with a block size setting of 200kb, exactly matching the Cosmos Hub default value, as can be seen in the graph of block time below.



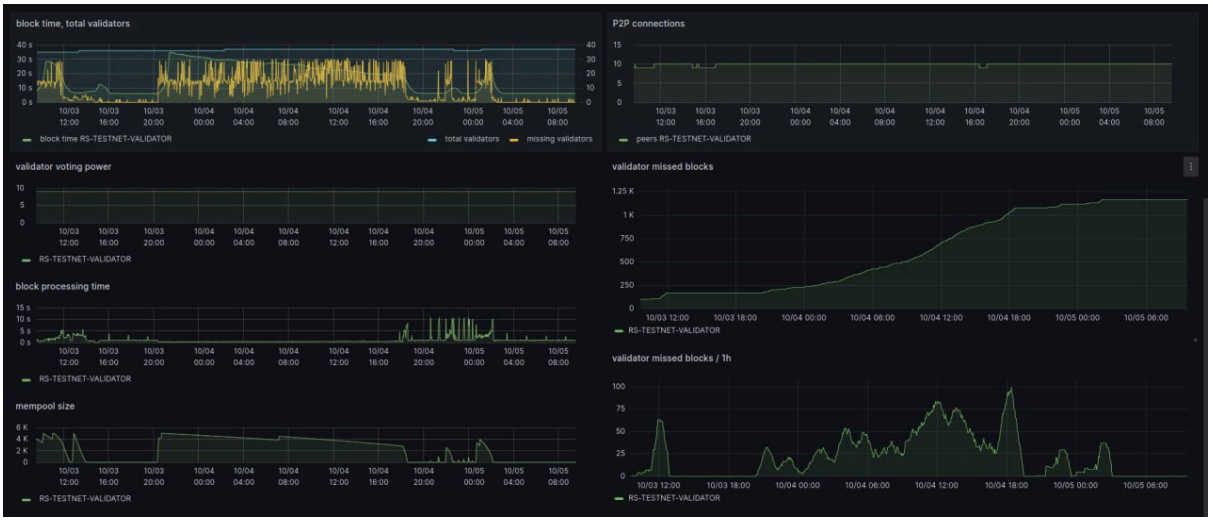
In particular, in the image below, we note from top to bottom:



Left: Block time across all validators can be seen spiking several times up to ~5 min, and peaking just below 8 min. Block processing time can be seen becoming much more volatile over the course of the attack. Mempool size experiences both a plateau far above normal, followed by violent spikes.

Right: We can see the peer connections periodically dropping out due to network stress. The absolute number of missed blocks spikes violently as the attack progresses and often plateaus far above normal. Additionally the rate of missed blocks of the attacking validator also experiences similarly violent spikes.

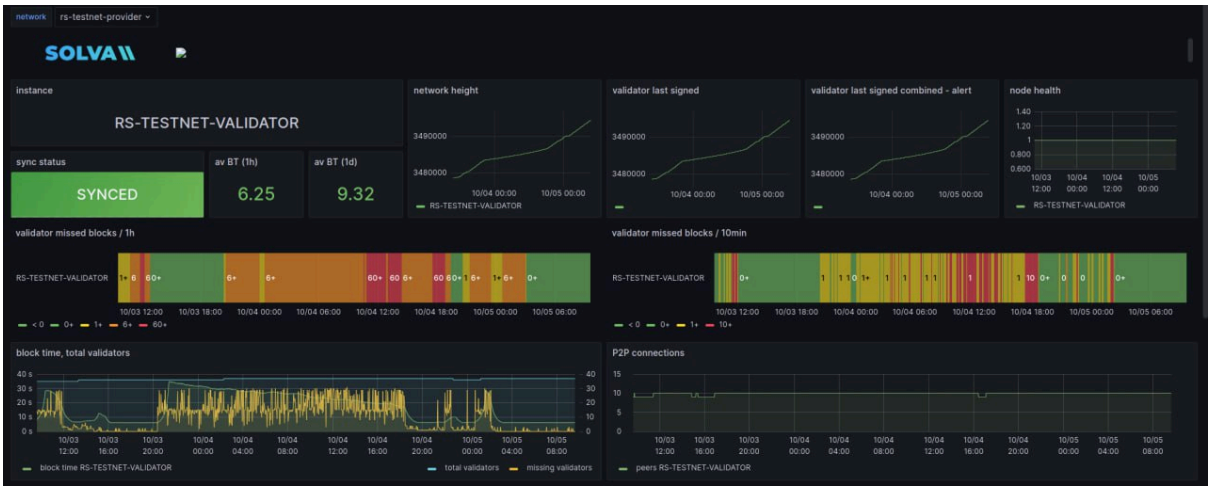
We also collected data over a 48 hour period **after** the attack was completed and found some concerning results. For example, from top to bottom, we notice:



Left: We notice extended periods of time where block times oscillate between 20 and 30 seconds - a reduction in network performance of a factor of ~4. We also no mantice spikes in block processing time. During the periods where blocktime is greatly increased, we see the mempool struggling to clear itself, further inflaming the block processing time of validators.

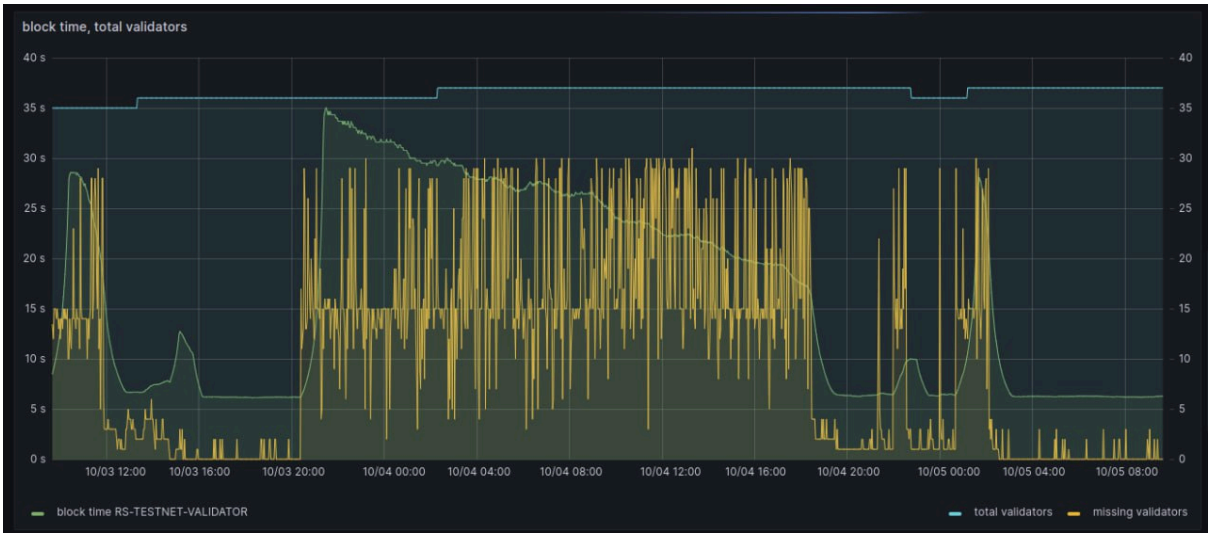
Right: While the P2P connections remain relatively consistent after the attack has completed, we see a disturbing increase in the number of missed blocks as the network continues to struggle to reclaim stabillity.

In the image below we can see slightly more detail regarding the block production. Most notably, after 48 hours, the average block time has returned to ~ 6 seconds, but the average blocktime for the last 24 hours sits around ~9.3 seconds, still on the order of 50% higher than expected.

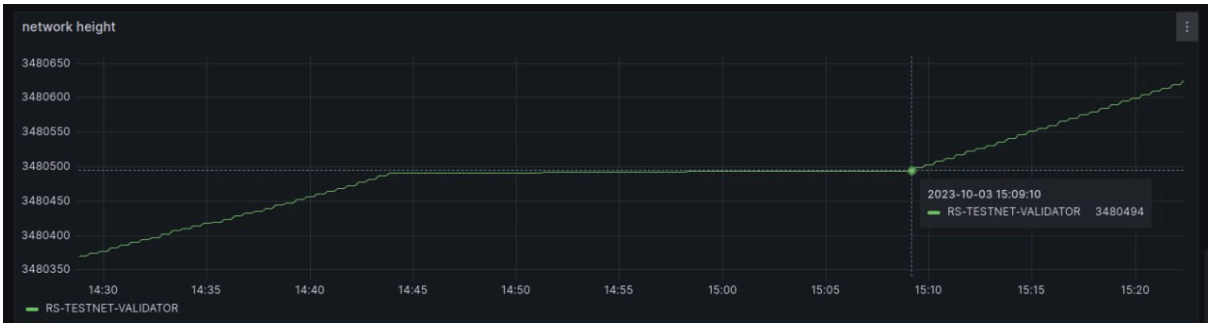


Additionally, see the network height dragging significantly this post-attack instability compared to it's baseline rate of progress during normal network conditions.

Below you can see a close up of the block time volatility in the 24 hours after the attack had been completed.



Zooming into shorter time frames, we can see a period of ~30 minutes where the network essentially halted.

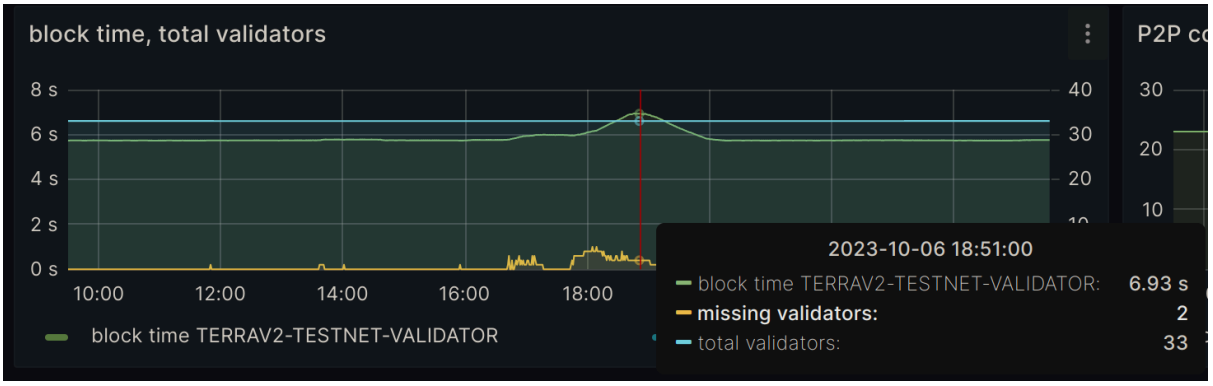


October 6 - 2023: Terra Testnet

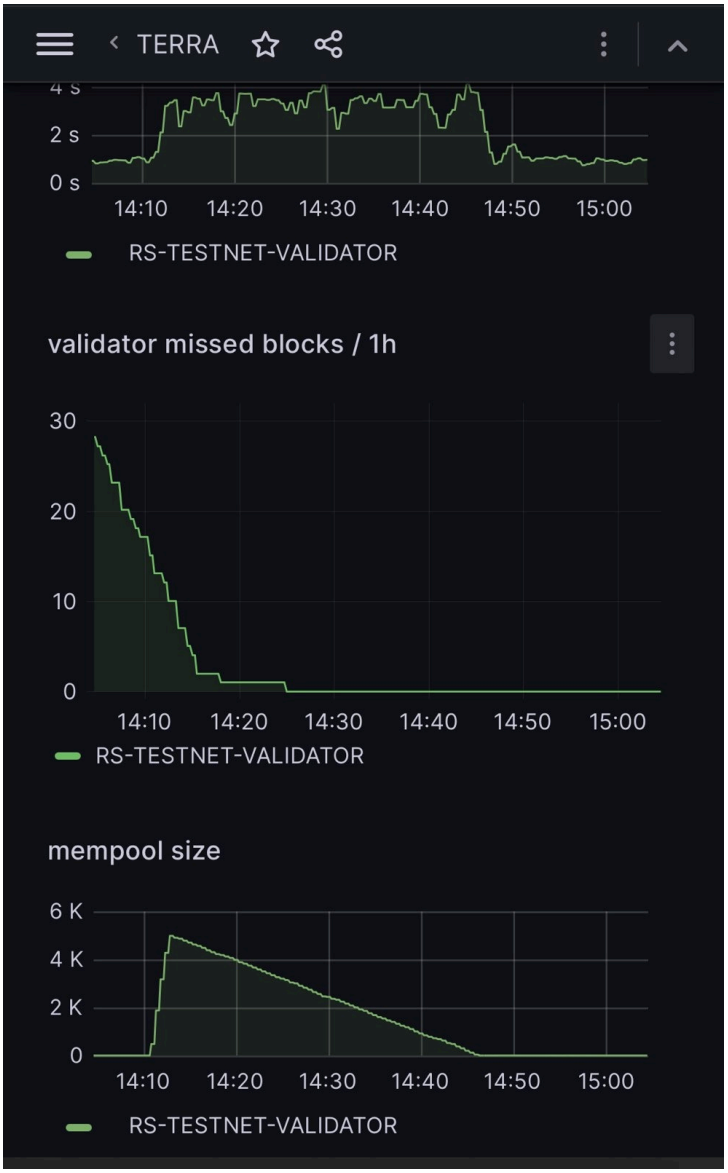
In collaboration with Emidev98 from TFL and Clemens from CryptoCrew, the attack previously launched on the CHRS testnet was replicated on the Terra network. You can see the mempool bloat in the graph below:

The network slowed exactly as described, despite the attempts of TFL to mitigate the issue with 800kb block sizes. - shortly after the test attack began, the network showed signs of deterioration leading to complaints by dev ops teams. We opted to halt the attack much sooner than we did on the CHRS testmnet, but were still able to obtain some data.

Below we can see a close up view of the attack, in which block times rise from ~5.75 seconds to a peak of ~6.95 s - a 21% deterioration in network performance. We also see the number of missing validators go from 0/33 before the attack to a peak of 5.

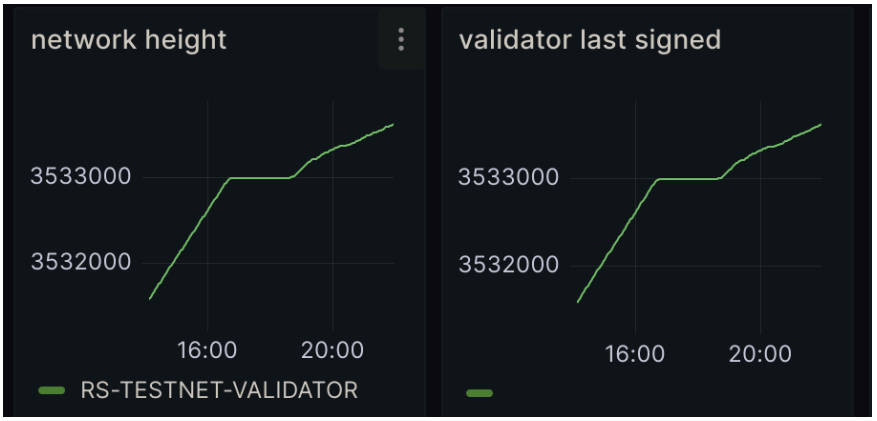


It is clear from this follow-up attack that reducing block size is an insuffcient mitigation of the aforementioned issues because it does not help with the flooding of the mempool.

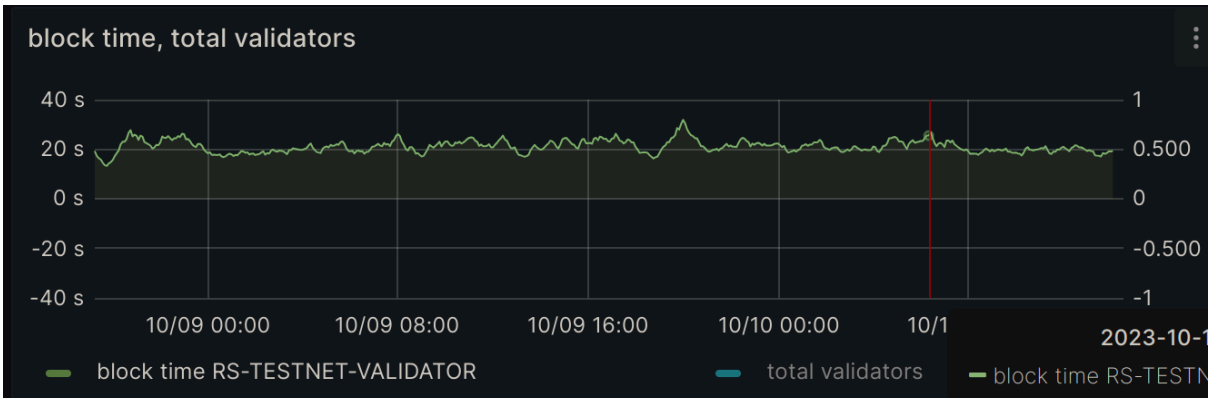


On October 9th 2023, a refined version of the attack was launched on the CHRS testnet. It was both reasonably simple and cheap to deploy. You can see in the graph below the mempool bloat during the attack:

This particular version of the attack was so devatstaing that it actually halted the network entirely for almost 2 hours:

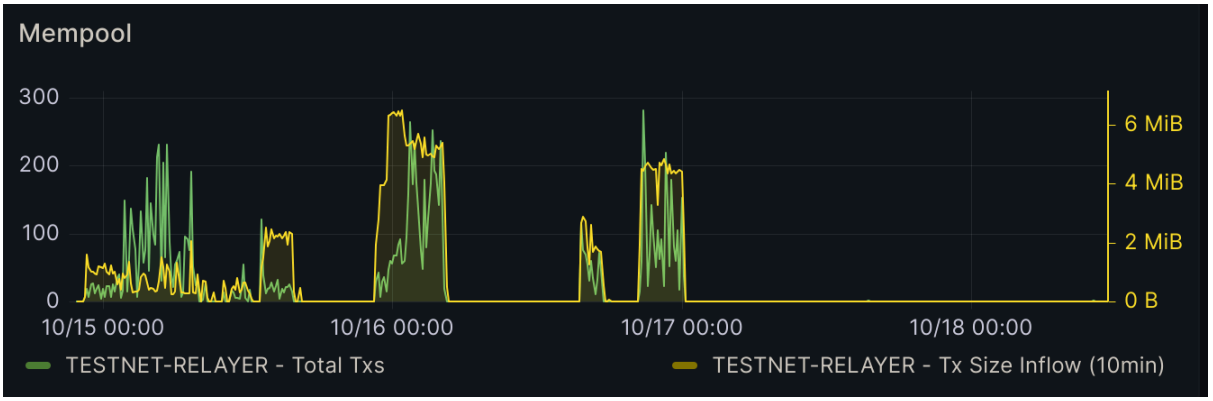


Additionally, 48 hours after the attack had finished, network congestion persisted. This resulted in block times oscillating between 15 and 32 s.

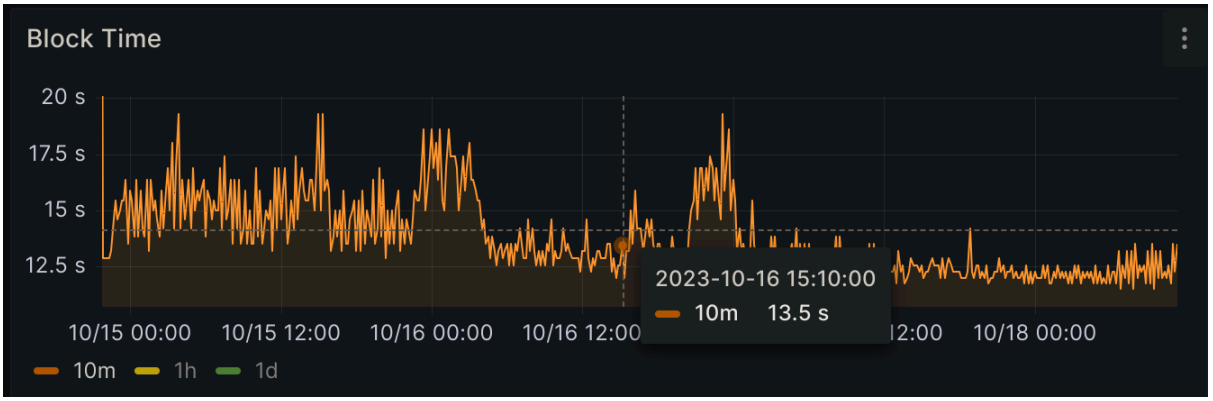


15-17 October 2023: Celestia Mocha4 Testnet

The Notional Security has also attempted a reproduction of the BanaKing style attacks on the Celestia testnet, with some very interesting findings. Below, you can see evidence of the attacks in the swelling of the mempool.



What is most interesting in this attack is thew relatively small impact on blocktimes despite this. We can see from the graph below that blocktimes did increase from their normal ~12s mark to a peak of ~19s. While this indicates a peak deprecation in performance of ~58%, the average block time over the course of the attack was closer to ~15s which is only ~25% deprecation.



Additionally, the Celestia testnet does not appear to have continued suffering from the effects of the attack after it had finished, unlike the CHRS testnet.