New issue

Jump to bottom

Consensus: "enterPrevote: ProposalBlock is invalid" - Error: "wrong signature" #4926

Closed njmurarka opened this issue on May 31, 2020 · 17 comments

Assignees

Labels C:consensus

njmurarka commented on May 31, 2020

Tendermint version:

Tendermint Core Semantic Version: 0.33.3
P2P Protocol Version: 7
Block Protocol Version: 10


```
"node_info": {
    "protocol_version": {
      "p2p": "7",
      "block": "10",
      "app": "0"
    },
    "id": "dbc39feecf277f59b4b16ae277e8545c54ac244a",
    "listen_addr": "tcp://0.0.0.0:26656",
    "network": "bluzelle",
    "version": "0.33.3",
    "channels": "4020212223303800",
    "moniker": "daemon-sentry-3",
    "other": {
      "tx_index": "on",
      "rpc_address": "tcp://0.0.0.0:26657"
    }
```


ABCI app:

Cosmos SDK Version: v0.38.3


```
"application_version": {
    "name": "BluzelleService",
    "server_name": "blzd",
    "client_name": "blzcli",
    "version": "0.0.0-74-ge1ee575",
    "commit": "e1ee575051ad2ea18ef22fc6bf7a6fc904612a49",
    "build_tags": "ledger,faucet,cosmos-sdk v0.38.3",
    "go": "go version go1.14.3 linux/amd64"
  }
```


Big Dipper Explorer URL:

```
http://explorer.testnet.public.bluzelle.com:3000
```

Instructions to setup a similar node (I'd suggest just setting up a sentry):

```
https://github.com/bluzelle/curium/blob/devel/docs/public/buildvalidatorsentry.md
```

Access to genesis file for chain:

```
http://a.sentry.bluzellenet.bluzelle.com:1317/genesis.json
```

Sample command to get node info:

```
curl --location --request GET 'http://a.sentry.testnet.public.bluzelle.com:1317/node_info'
```

Discord channel invite (in case you want to live chat with me... I am Neeraj one of the admins):

```
https://discord.gg/BbBZJZJ
```

Environment:

- **OS** (e.g. from /etc/os-release):

```
Distributor ID: Ubuntu
Description:  Ubuntu 18.04.4 LTS
Release:  18.04
Codename: bionic
```

- **Install tools**:

Using COSMOS SDK v0.38.3. Otherwise, not sure what else to say here.

- **Others**:

We are running a testnet chain with our CRUD database as one of the application modules, in COSMOS.

We currently (as of filing this issue) have 5 "sentries" and 3 validators. To be clear, the sentries have no voting power and are the only peers that the validators talk to (the validators can talk to each other too). Furthermore, the validators are IP firewalled to only be able to talk to the sentries and other validators. The sentries themselves keep the validator node id's private.

Sentry hostnames:

```
a.sentry.testnet.public.bluzelle.com
b.sentry.testnet.public.bluzelle.com
c.sentry.testnet.public.bluzelle.com
d.sentry.testnet.public.bluzelle.com
e.sentry.testnet.public.bluzelle.com
```

I am not listing the validator hostnames, since they are inaccessible (due to the firewall) anyways.

The validators are only listening on 26656 to validators and sentries. The sentries are listening on 26656 and 26657 and also each run the cosmos REST server, listening on 1317.

We have opened our testnet to the public. Members of the public have setup sentries and validators of their own, and are expected to use our five sentries as their P2P peers in config.toml.

What happened:

For weeks, things on our testnet had been running fine. I had dozens of members of the public running validators on it, just so these people could learn the process of setting up a validator, etc.

I needed to increase the max # of allowed validators (to something much higher than the default value of 100) in the "app_state/staking/params/max_validators" value in genesis.json. I think that this particular value is a COSMOS thing, but I wanted to mention it for context. We are not using COSMOS governance yet, so we decided to do a hard reset (ie: generate a new genesis.json and start the chain all over).

First, here is what I did on my OWN 5 sentries and 3 validators:

- Stopped all my sentries and validators.
- Wiped out their .blzd folders (this is the name of my "home" folder for my "blzd" daemons). Because of this, the nodes will all get new node ids and will be new "peers".
- Re-initialized each sentry and validator with "blzd init", etc... much like I always do when I setup a validator or sentry from scratch (setting up peers, etc). I had also increased the "max_num_inbound_peers" to 800 and "max_num_outbound_peers" to 200, in the [p2p] section of config.toml. This might only be anecdotal in value. I had an issue where we had too many connects to my sentries and they were dropping connections on the p2p port.
- Generated the new genesis.
- Deployed this genesis to all the sentries and validators.
- Run the necessary COSMOS commands to get the validators staked, created, etc.
- Start up all my sentries and validators, (thereby starting up the new chain from block 0).

Next, here is what I asked the people in the community to do with their validator and/or sentries:

- Run "blzd unsafe-reset-all" on all their daemons. I asked the community do this instead of wiping out the ".blzd" folder, to save them some work.
- Copy over the new genesis.json file, replacing the old genesis.json.
- Set the new peers list in the p2p section of config.toml.
- Run the necessary COSMOS commands to get the validators staked, created, etc.
- Start up their sentries and validators.

The community slowly started up their daemons.

At some point (within an hour or so, about 2300 blocks in), I started to get the error below. I was getting this on all my sentries and validators. Basically, the chain had completely crashed. I tried to restart my validators and sentries, but this was unrecoverable.

```
E[2020-05-29|03:03:32.975] enterPrevote: ProposalBlock is invalid       module=consensus height=2285 round=0 err="wrong signature (#35):
C683341000384EA00A345F9DB9608292F65EE83B51752C0A375A9FCFC2BD895E0792A0727925845DC13BA0E208C38B7B12B2218B2FE29B6D9135C53D7F253D05"
E[2020-05-29|03:03:35.128] enterPrevote: ProposalBlock is invalid       module=consensus height=2285 round=1 err="wrong signature (#35):
C683341000384EA00A345F9DB9608292F65EE83B51752C0A375A9FCFC2BD895E0792A0727925845DC13BA0E208C38B7B12B2218B2FE29B6D9135C53D7F253D05"
E[2020-05-29|03:03:37.255] enterPrevote: ProposalBlock is invalid       module=consensus height=2285 round=2 err="wrong signature (#35):
C683341000384EA00A345F9DB9608292F65EE83B51752C0A375A9FCFC2BD895E0792A0727925845DC13BA0E208C38B7B12B2218B2FE29B6D9135C53D7F253D05"
.
.
.
```

To had no choice but to "reset" the whole chain again. I stopped all my validators and sentries and this time, ONLY ran "unsafe-reset-all" on all my daemons. Of course, I also had to do some COSMOS setup again (staking, etc), but started everything again, and asked the community to again do the same steps listed above with yet a new genesis.json, etc.

Within an hour, the whole network went down again. Effectively the same error (different block, signature HASH this time):

```
.
.
.
E[2020-05-29|06:57:48.621] enterPrevote: ProposalBlock is invalid       module=consensus height=676 round=146 err="wrong signature (#8):
62A6A628CFB1F72D76C48F71A928DD628E29585DD4B861EDF3F216E77FBB0A7C492D2280B218FBA34A0751F02961C2657708711D3F212800CFE847B804F0360D
.
.
.
```

What you expected to happen:

I expect "clean" output, as so:

```
I[2020-05-30|21:53:04.286] Executed block                               module=state height=20416 validTxs=0 invalidTxs=2
I[2020-05-30|21:53:04.309] Committed state                              module=state height=20416 txs=2 appHash=80D70DC5FF062F34D3F79F15FC85CB367A5A7F9CF39B4EE6C1DC68E9F1958EA1
```

(The fact that invalidTxs is non-zero is the subject of another investigation)

Have you tried the latest version:

Not sure. I think so. Although in looking at the Tendermint Github, I see there are two minor versions available that are newer than what we have.

How to reproduce it:

I more or less explained how it came about above in the "what happened" section.

Looking at #2720, I see a similar error message, but not quite the same. But in looking at that issue, it was suggested that perhaps all the nodes did not start from the same "genesis" state. It is suggested that perhaps some node(s) have a stale "home folder" (.blzd, I presume?).

Does "unsafe-reset-all" actually clear out all state including the COSMOS KV stores, app state, etc? I assume this command is sufficient to accomplish a clean slate?

Is it possible that in "resetting" my chain as I did above, some members of the public possibly forgot to run that "blzd unsafe-reset-all" command, and by missing this step, when they started their node, it had data from the previous chain left over, and this somehow brought the whole network down? If so, it is a bit scary that a single node (or even a bunch of them) could do this. It is an excellent DoS attack vector, it seems, if so.

Logs:

Listed above.

Config:

No specific changes made to Tendermint.

node command runtime flags:

This is all running from within our daemon that was built with the COSMOS SDK.

`/dump_consensus_state` output for consensus bugs

Not sure how to do this.

Anything else we need to know:

Most details given above.

I did some searching ahead of time to see if I could resolve this myself. I saw some issues related to it but they are already closed.

❤️ 6

njmurarka mentioned this issue on May 31, 2020

p2p mode: "auth failure", "ProposalBlock is invalid" & "duplicate CONN" #2720

Closed

melekes added the C:consensus label on Jun 1, 2020

melekes commented on Jun 2, 2020 — Contributor

> Does "unsafe-reset-all" actually clear out all state including the COSMOS KV stores, app state, etc? I assume this command is sufficient to accomplish a clean slate?

Yes, at least it's supposed to.

> E[2020-05-29|06:57:48.621] enterPrevote: ProposalBlock is invalid module=consensus height=676 round=146 err="wrong signature (#8)

This error basically means validator `#8` has an old genesis file (or state) => it's signature is incorrect. proposal block is considered invalid since the commit it contains has an invalid signature in it. Now, the question is how the invalid signature made it's way into the commit? we always verify signatures/votes before adding them. 2/3+ of voting power has to agree on content of commit (chainID has to be the same) in order to form a valid commit.

> Is it possible that in "resetting" my chain as I did above, some members of the public possibly forgot to run that "blzd unsafe-reset-all" command, and by missing this step, when they started their node, it had data from the previous chain left over, and this somehow brought the whole network down?

It's possible in theory, but it's not a desired behavior. We haven't seen this before.

melekes commented on Jun 2, 2020 — Contributor

@njmurarka could you please share more logs? we'll need them to understand what has happened precisely.

njmurarka commented on Jun 2, 2020 — Author

@melekes I would love to. I have the log file of output from one of my daemons. It is attached here.

blzd.log

Thank you for clarifying what that "enterPrevote: ProposalBlock is invalid module=consensus" error means.

But it really begs the question that worries me. Even if someone had an old genesis file. How could they have caused the network to go into this state?

The chain gets started with my three validators, and they have such an enormous stake that it is not possible for anybody else to even get 10% power, if even 1% power. I did this intentionally for my testnet.

So with this in mind, how can a single "rogue" node that has "bad setup" accomplish this?

I am pretty confident now that somebody simply had "old data" of some sort, and that caused this. Maybe the old genesis. Maybe it was the old kv store?

I have not attempted to replicate this, but I can tell you that it happened twice to me. Really scary if this were to happen on a mainnet.

erikgrinaker added the P:High label on Jun 3, 2020

melekes commented on Jun 5, 2020 — Contributor

Do you happen to have the `/consensus_state`, `/dump_consensus_state` output from stale nodes?

```
curl --location --request GET 'http://a.sentry.testnet.public.bluzelle.com:1317/consensus_state'
```

melekes commented on Jun 5, 2020 — Contributor

Also, it would've been great to have logs with DEBUG level

njmurarka commented on Jun 8, 2020 — Author

Unfortunately, **@melekes**, I do not have these nodes running at this point.

In fact, to be honest, I do not even know which nodes were "stale" since I am not yet clear what happened, who was stale, and what stale means here. Is this a node that has a new validator private key? An old one? Or merely a validator that has the old genesis file?

It is actually pretty scary, but I might be wrong. The reason it is scary is I am pretty confident my 3 validators (who hold 99% voting power) did everything correct, and it was one of the other validators that joined (I have 100+ other validators being run by members of the public) that might be somehow responsible. But if this is in fact true, that means that someone with < 1% voting power had a means to crash the network. I sure hope I am wrong.

I think we really need to hunt down and find an explanation for this. It puts my my ill at ease knowing this happened twice and now, it is not happening. The best theory so far is a bad validator, yet it really puts the whole point of a decentralized network (with the ensuing security) in a bad light.

I am attaching a log of the console output from "blzd" for one of the nodes. I am unclear how helpful this is, but it is better than nothing. Please scroll down to line 4,666 to see where the issue manifested.

blzd.log

melekes commented on Jun 9, 2020 — Contributor

No worries!

Looks like the only path here is to try and replicate the issue + reason through the code.

njmurarka commented on Jun 9, 2020 — Author

@melekes

We are running a "Game of Stakes" type competition right now, so I am a bit heads down. But I want to quash/explain this issue, as it is worrisome and should be to others (if it is legitimate and not something dumb on my side).

I will try to make it happen intentionally in the next few weeks. I have the ability to launch new testnets pretty quickly.

👍 3

melekes self-assigned this on Jun 10, 2020

melekes added a commit that referenced this issue on Jun 17, 2020

`more test cases for TestValidatorSet_VerifyCommit` ··· ✕ `cf18821`

melekes mentioned this issue on Jun 17, 2020

types: more test cases for TestValidatorSet_VerifyCommit #5018

Merged

melekes commented on Jun 19, 2020 — Contributor

Hasn't been able to find a reason so far.

mergify bot pushed a commit that referenced this issue on Jun 19, 2020

`types: more test cases for TestValidatorSet_VerifyCommit (#5018)` ··· ✓ `8b4a30f`

njmurarka commented on Jun 28, 2020 • edited — Author

Hello **@melekes**.

So, as a bit of fortunate news (well, not that fortunate), this "crash" has occurred again. I reached out to you on Discord but we can continue here too.

Here is some output that might look familiar:

```
I[2020-06-28|04:56:57.031] Executed block                          module=state height=385080 validTxs=0 invalidTxs=5
I[2020-06-28|04:56:57.053] Committed state                         module=state height=385080 txs=5 appHash=33BB2F44035D6A31668206B095670D81011DF8DA6EA83E73D197AE604CD34CA
I[2020-06-28|04:57:03.908] Executed block                          module=state height=385081 validTxs=0 invalidTxs=5
I[2020-06-28|04:57:03.931] Committed state                         module=state height=385081 txs=5 appHash=089451C122173385CCB904D34C7FBCC592248A6983C3201A07C04D4876DEE9A4
I[2020-06-28|04:57:10.632] Executed block                          module=state height=385082 validTxs=0 invalidTxs=5
I[2020-06-28|04:57:10.656] Committed state                         module=state height=385082 txs=5 appHash=80B48A34F8C110106607DA13AC55725D2D18EAA1D66CBA9D95FACD4CA0B21D94
I[2020-06-28|04:57:17.655] Executed block                          module=state height=385083 validTxs=0 invalidTxs=5
I[2020-06-28|04:57:17.660] Updates to validators                   module=state updates=00C890F67A03D76EEEA10B7AECBEE4FAD23ED19A:0
I[2020-06-28|04:57:17.688] Committed state                         module=state height=385083 txs=5 appHash=A2DE8FCB717D0898DB7A23F244B24B7455790D7114346EC5180DF300D1DD2C2B
I[2020-06-28|04:57:23.592] Executed block                          module=state height=385084 validTxs=0 invalidTxs=4
I[2020-06-28|04:57:23.616] Committed state                         module=state height=385084 txs=4 appHash=2CBEB80E0CBD3F3247DA20AF76CE96DCC533EBDF4489EB8EDEDAB789E6DD27EF
I[2020-06-28|04:57:30.333] Executed block                          module=state height=385085 validTxs=0 invalidTxs=5
```

```
I[2020-06-28|04:57:30.356] Committed state                              module=state height=385085 txs=5 appHash=398DA9C83A05C449238C129C785DA90A0D5EBED11AADBAE90A5810F239D62864

E[2020-06-28|04:57:30.975] MConnection flush failed                     module=p2p peer=f1b412c2a33bda98f42959e86602a0ca9a3ede22@54.163.188.170:26656 err="write tcp
172.26.7.15:38078->54.163.188.170:26656: write: connection reset by peer"

E[2020-06-28|04:57:30.975] Stopping peer for error                      module=p2p peer="Peer{MConn{54.163.188.170:26656} f1b412c2a33bda98f42959e86602a0ca9a3ede22 out}" err=EOF

E[2020-06-28|04:57:35.607] enterPrevote: ProposalBlock is invalid       module=consensus height=385086 round=0 err="wrong signature (#0):
42C97B86D89D56F1A07D778C4F22196F23F3F47C67EF37AC39EC06C2D8FB493AB40C99CF92811573D3FE3F61744AA45166091EB1FDBFAD02B8DA65ADDCBC160F"

E[2020-06-28|04:57:38.201] enterPrevote: ProposalBlock is invalid       module=consensus height=385086 round=1 err="wrong signature (#0):
42C97B86D89D56F1A07D778C4F22196F23F3F47C67EF37AC39EC06C2D8FB493AB40C99CF92811573D3FE3F61744AA45166091EB1FDBFAD02B8DA65ADDCBC160F"

E[2020-06-28|04:57:40.801] enterPrevote: ProposalBlock is invalid       module=consensus height=385086 round=2 err="wrong signature (#0):
42C97B86D89D56F1A07D778C4F22196F23F3F47C67EF37AC39EC06C2D8FB493AB40C99CF92811573D3FE3F61744AA45166091EB1FDBFAD02B8DA65ADDCBC160F"

E[2020-06-28|04:57:44.311] enterPrevote: ProposalBlock is invalid       module=consensus height=385086 round=3 err="wrong signature (#0):
42C97B86D89D56F1A07D778C4F22196F23F3F47C67EF37AC39EC06C2D8FB493AB40C99CF92811573D3FE3F61744AA45166091EB1FDBFAD02B8DA65ADDCBC160F"

E[2020-06-28|04:57:48.299] enterPrevote: ProposalBlock is invalid       module=consensus height=385086 round=4 err="wrong signature (#0):
42C97B86D89D56F1A07D778C4F22196F23F3F47C67EF37AC39EC06C2D8FB493AB40C99CF92811573D3FE3F61744AA45166091EB1FDBFAD02B8DA65ADDCBC160F"
```

It appears to be the exact same problem.

When I restart a node that has "crashed" (presumably with the output above), I get the following:

```
blzd start
I[2020-06-28|10:20:48.228] starting ABCI with Tendermint                module=main
I[2020-06-28|10:20:48.241] Module setup                                 module=main bluzelle_crud=true

deleting 385022, 385022demo-d1272187875103293441
panic: Failed to reconstruct LastCommit: Failed to verify vote with ChainID bluzelle and PubKey PubKeyEd25519{3FD21AAD1FAAF2AF272D4C8B005F19B8F93B6C3C34A44E7FDAF07DD51EE54F56}:
invalid signature

goroutine 1 [running]:
github.com/tendermint/tendermint/types.CommitToVoteSet(0xc004d18790, 0x8, 0xc00354c800, 0xc00544fa40, 0x0)
/go/pkg/mod/github.com/tendermint/tendermint@v0.33.3/types/block.go:594 +0x496
github.com/tendermint/tendermint/consensus.(*State).reconstructLastCommit(0xc000096000, 0xa, 0x0, 0xc004d18780, 0x6, 0xc004d18790, 0x8, 0x5e03d, 0xc0059d8ae0, 0x20, ...)
/go/pkg/mod/github.com/tendermint/tendermint@v0.33.3/consensus/state.go:493 +0x73
github.com/tendermint/tendermint/consensus.NewState(0xc00017a5a0, 0xa, 0x0, 0xc004d18780, 0x6, 0xc004d18790, 0x8, 0x5e03d, 0xc0059d8ae0, 0x20, ...)
/go/pkg/mod/github.com/tendermint/tendermint@v0.33.3/consensus/state.go:177 +0x502
github.com/tendermint/tendermint/node.createConsensusReactor(0xc0000da8c0, 0xa, 0x0, 0xc004d18780, 0x6, 0xc004d18790, 0x8, 0x5e03d, 0xc0059d8ae0, 0x20, ...)
/go/pkg/mod/github.com/tendermint/tendermint@v0.33.3/node/node.go:388 +0x170
github.com/tendermint/tendermint/node.NewNode(0xc0000da8c0, 0x1608900, 0xc000030820, 0xc000e2cb30, 0x15edfc0, 0xc000085240, 0xc000e2cd40, 0x140b500, 0xc000e2cd50, 0x160ee40, ...)
/go/pkg/mod/github.com/tendermint/tendermint@v0.33.3/node/node.go:661 +0x985
github.com/cosmos/cosmos-sdk/server.startInProcess(0xc000dab480, 0x140bd28, 0x1d, 0x0, 0x0)
/go/pkg/mod/github.com/cosmos/cosmos-sdk@v0.38.3/server/start.go:157 +0x4c1
github.com/cosmos/cosmos-sdk/server.StartCmd.func1(0xc000dd6840, 0x1f17138, 0x0, 0x0, 0x0, 0x0)
/go/pkg/mod/github.com/cosmos/cosmos-sdk@v0.38.3/server/start.go:67 +0xb4
github.com/spf13/cobra.(*Command).execute(0xc000dd6840, 0x1f17138, 0x0, 0x0, 0xc000dd6840, 0x1f17138)
/go/pkg/mod/github.com/spf13/cobra@v0.0.6/command.go:840 +0x453
github.com/spf13/cobra.(*Command).ExecuteC(0xc0000c5340, 0x2, 0xc000dab820, 0x12a7fda)
/go/pkg/mod/github.com/spf13/cobra@v0.0.6/command.go:945 +0x317
github.com/spf13/cobra.(*Command).Execute(...)
/go/pkg/mod/github.com/spf13/cobra@v0.0.6/command.go:885
github.com/tendermint/tendermint/libs/cli.Executor.Execute(0xc0000c5340, 0x140c150, 0x2, 0xc000d7f390)
/go/pkg/mod/github.com/tendermint/tendermint@v0.33.3/libs/cli/setup.go:89 +0x3c
main.main()
/go/src/github.com/bluzelle/curium/cmd/blzd/main.go:84 +0x777
bash-5.0#
```

I have kept my nodes running so we can resolve this matter. It is quite worrisome. I don't want to make assumptions. It is possible this is a result of the code we have in the COSMOS layer.

That having been said, it seems like this is a genuine Tendermint issue. I don't see how any sort of code in the COSMOS layer should be able to cause this, even if we intentionally wanted it to be so.

It is worrisome because we want to run a Mainnet soon. But having a crash like this makes that an unwise choice till we positively REPRODUCE this issue, and then resolve it. I obviously cannot sign off on a mainnet with an issue like this that is outstanding.

Thoughts? Please advise what you need from me.

Thanks.

ebuchman commented on Jun 28, 2020 • edited — Collaborator

Thanks for the reports on this. Can you please provide output from the `/consensus_state` and `/dump_consensus_state` RPC endpoints?

Would it be possible for us to get remote access to the RPC end points of running nodes so we can dig in a bit ourselves?

ebuchman commented on Jun 28, 2020 • edited — Collaborator

Also to clarify - do these livelocks only happen after a genesis upgrade? And are you changing the chain-id after genesis upgrade? Technically once a chain-id has been used it shouldn't be used again in a genesis restart because validator signatures from the previous chain could be considered misbehaviour (eg. a signature for the old block 5 could conflict with a signature for the new block 5 from the same validator and could be seen as evidence, caused that validator to be slashed!). I'm not sure that's relevant in this case to the infinite "invalid proposal block" but it's something to keep in mind in general.

ebuchman mentioned this issue on Jun 28, 2020

.github/issue_template: Update `/dump_consensus_state` request. #5060

Merged

njmurarka commented on Jun 28, 2020 • edited — Author

> Also to clarify - do these livelocks only happen after a genesis upgrade? And are you changing the chain-id after genesis upgrade? Technically once a chain-id has been used it shouldn't be used again in a genesis restart because validator signatures from the previous chain could be considered misbehaviour (eg. a signature for the old block 5 could conflict with a signature for the new block 5 from the same validator and could be seen as evidence, caused that validator to be slashed!). I'm not sure that's relevant in this case to the infinite "invalid proposal block" but it's something to keep in mind in general.

Real quick replies to keep the momentum... we did not change or "upgrade" the genesis. I am not 100% sure what a "genesis" upgrade actually is, although I am 70% confident of my guess. We did nothing to status quo. Network was running for 3+ weeks and then boom -- this occurred!

No change to the chain-id. No upgrade. No changes other than validators getting jailed, getting unbonded, and new validators trying to join. All natural, "organic" actions that are part of any Tendermint/COSMOS chain.

Can you clarify what you mean when you say that a chain-id should not be re-used? This is a potentially a critical guideline. Indeed we have reset our network a few times and have re-used the same chain-id. That having been said, it scares the **** out of me that a network can crash like this. It contradicts the whole "high availability" value proposition of a blockchain, to have a "bug" like this. Even if a single validator has "bad DB data" (validator signatures from the previous chain), it makes no sense that this alone could cause the entire network to crash this way, does it? I mean really... if 99.9% of the voting power is ok with things and some validator shows up with bogus/antiquated data, how can that alone cause the whole network to crash like this? I sincerely hope I am wrong about this guess. And even if I am not, it would be great to resolve this. As it stands, our testnet is on its knees... stalled. I don't know how to recover it. Some nodes are running but no new blocks are forthcoming.

I will get on providing answers to your request for output from those two RPC endpoints immediately.

This having been said... I and some members of the community are very alarmed this has happened a few times... just to be clear, Bluzelle runs the three validators that collectively hold 97%+ of the voting power. So it is disturbing to think that some arbitrary validator out there, with "bad data" could cause this. It is a pretty good DoS attack vector, at a minimum.

Thanks so very much, btw. Appreciate the quick feedback. I think that if this is truly an issue in Tendermint (I am 60% confident it is), it behooves us to resolve it quickly.

njmurarka commented on Jun 28, 2020 • edited — Author

> Thanks for the reports on this. Can you please provide output from the `/consensus_state` and `/dump_consensus_state` RPC endpoints?
>
> Would it be possible for us to get remote access to the RPC end points of running nodes so we can dig in a bit ourselves?

Consensus State (link and file attached):

http://dev-backup.testnet.public.bluzelle.com:26657/consensus_state

consensus_state.txt

Dump Consensus State (link and file attached):

http://dev-backup.testnet.public.bluzelle.com:26657/dump_consensus_state

dump_consensus_state.txt

Can you please confirm you have access at http://dev-backup.testnet.public.bluzelle.com:26657 to the endpoint you are asking for?

What else can I provide to assist?

Thanks.

ebuchman commented on Jun 28, 2020 — Collaborator

Thanks, this is super helpful. Looks like we've identified the problem and managed to replicate this issue. Will publish a fix ASAP.

👍 2

njmurarka commented on Jun 29, 2020 • edited — Author

> Thanks, this is super helpful. Looks like we've identified the problem and managed to replicate this issue. Will publish a fix ASAP.

Thanks. That's great!

Please do share here what the problem was, how you replicated, and how it was resolved? I'd love to see the discussion on this, if it was public.

Will we be able to "save" our existing testnet with this fix?

ebuchman commented on Jun 29, 2020 — Collaborator

Yes we will share details on all of that once the fix is released.

And we will try to provide a script that should allow you to save your existing testnet. Thanks for your patience!

👍 1

tessr pushed a commit that referenced this issue on Jul 2, 2020

`consensus: Do not allow signatures for a wrong block in commits` ··· `8ccfdb9`

tessr closed this as completed in `480b995` on Jul 2, 2020

---

melekes reopened this on Jul 2, 2020

melekes closed this as completed on Jul 3, 2020

---

mergify (bot) pushed a commit that referenced this issue on Jul 6, 2020

`.github/issue_template: Update /dump_consensus_state request. (#5060)` ··· ✓ `bccc7fb`

sergio-mena mentioned this issue on Oct 18

docs: clarify `BlockIDFlag` variants #9590

Merged

3 tasks

Assignees

melekes

Labels

C:consensus

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

4 participants