Seven Days Dark: Anatomy of an Explorer That Lied

A stalled explorer does not throw an error. It answers in under a second with a confident number that is a week old. This is what actually broke, why each fix revealed another fault underneath, and the arithmetic that no configuration can argue with. All times are UTC.

SEPTEMBER 1, 2026 · 11 MIN READ · EASYBTX.COM/RESEARCH/SEVEN-DAYS-DARK

ABSTRACT

Our block explorer stopped following BTX on 24 August and served week-old data under a green LIVE badge for seven and a half days. Four separate faults were stacked on top of each other, and three volunteer operators dismantled them in one night. Every measurement, including the ones that embarrassed us.

All times in this article are UTC. Every figure is a measurement taken from a production machine, either ours or a named operator's, and re-verified immediately before publishing.

The silence a broken explorer makes

There is a particular kind of failure that monitoring does not catch.

Our block explorer stopped following the BTX chain on 24 August at block **199,296**. It did not crash. It did not go offline. It answered every request in under a second, with a green LIVE badge and a confident block height, and that height was a week old. Every uptime check we owned passed for seven and a half days while our own site quietly misled everyone who opened it.

This is the write-up we owe. It is not a victory lap, because the work is not finished. It is an account of four faults stacked on top of each other, the three volunteers who dismantled them in one night, and one piece of arithmetic that no amount of configuration can argue with.

It is also the sequel to Why Nodes Get Stuck, which described how to tell a healthy stalled node from a genuinely broken one. This is what it costs when nobody is watching that distinction on your own infrastructure.

The three blocks that mattered

On 25 August, BTX published a change to a difficulty rule scheduled to take effect at block 199,299. Two days later, upstream withdrew it. That two-day window is the whole story.

Nodes carrying the withdrawn rule split into two groups. Some followed it onto a branch that died a few blocks later. Others, ours included, simply stopped just below the boundary, because the peers

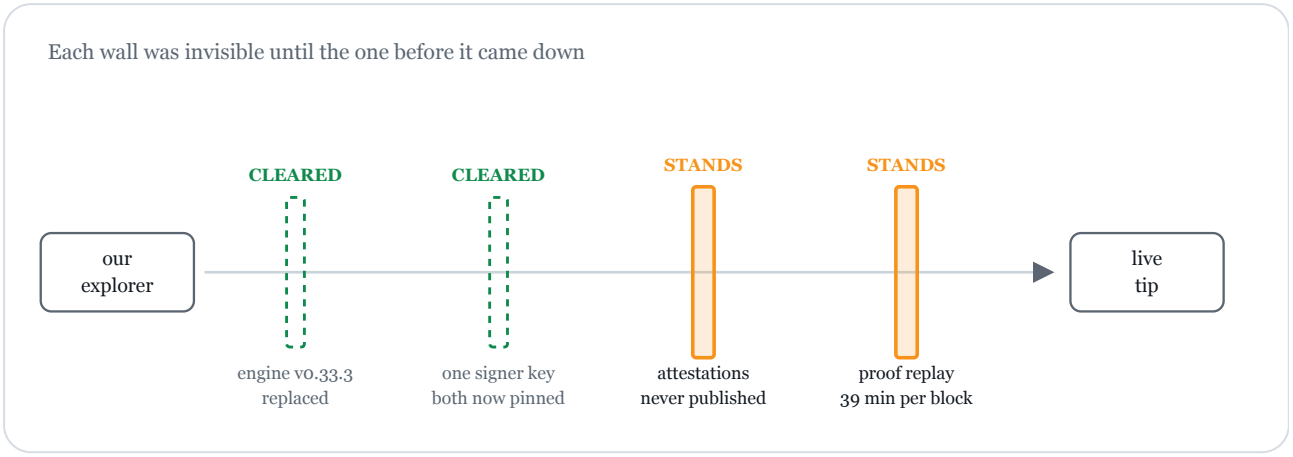
still willing to talk to them had moved on to blocks their engine would have rejected.

Our explorer's last accepted block was 199,296. The rule sat three blocks further on. We never crossed it and we never went wrong: block 199,295 on our node hashes to `33c834f8`, which is the live chain exactly. We were parked on the correct chain, in the correct order, one short step from the fork, unable to move and unable to say so.

That last part is the real defect, and it belongs upstream rather than to us. A node in this state has no way to tell its operator *you are parked at a boundary, upgrade to cross it*. It looks precisely like a connectivity fault: peers connect, peers vanish, the tip does not move. We spent days chasing peering ghosts because the software had no vocabulary for what was actually wrong. That feature request is now filed.

Four walls, one corridor

The expensive part of this outage was not any single cause. It was that each cause hid the next. Every time we removed a wall the chain advanced by one block and stopped again, which felt like failure and was actually progress.



The outage was not one bug. Replacing the outdated engine revealed a half-configured signer pin; fixing that revealed an upstream bug withholding the signatures our node needed; and behind those stood arithmetic that no setting can change.

Wall one: the engine

The obvious one. Our explorer ran an engine from before the turbulence and could not accept the post-fork header chain at all. Swapping it for the current release took minutes and moved the tip exactly nowhere.

Wall two: half a key

The second wall explains why the freeze landed on *exactly* 199,296 rather than drifting.

BTX mainnet has two signers, and they sign alternate blocks. Our node trusted one of them. Block 199,296 was signed by the key we trusted; 199,297 was signed by the other. A node configured that way does not fall behind gradually. It stops dead on the first block the other signer produced, and it stays there forever, no matter how good its hardware or its peers.

We would have been slower to find this without work someone else had already published. **jarekpiot** had hit the same wall across three of his own machines, a pool mirror, an explorer and a GPU validator, and wrote down the recovery that worked: move the engine and pin *both* published keys in the same maintenance window, not one after the other. That note turned our hardest diagnosis into a two-line change. He had nothing to gain from writing it.

Wall three: the signatures that never came

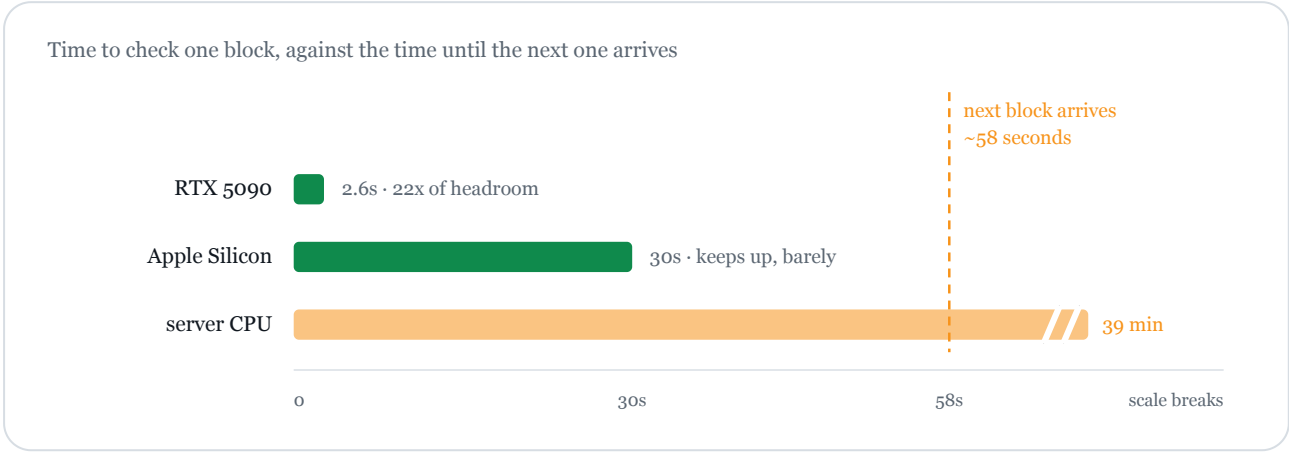
With both keys pinned, our node began asking the network for the signatures that would let it accept new blocks. It received none. Not few, zero, over an hour, with 34 peers connected and an attestation archive among them.

That is a known upstream bug: attestations are minted and never published. Our node was not misconfigured, it was waiting on a postal service that had quietly stopped delivering. A fix is in the next release. There was nothing to do about it that night.

The wall made of arithmetic

Behind those three sat something no flag could move, and it is the most important section of this article for anyone running BTX infrastructure.

BTX's proof of work is deliberately hostile to GPU farms, which means checking a block is nearly as expensive as producing one. A node that cannot hear a signed receipt has to redo that work itself, for every block. We measured what that costs on three classes of machine, then measured how fast the network produces blocks to check them against.



Anything left of the dashed line can follow the chain indefinitely. Anything right of it falls behind by design, and no amount of bandwidth, peers or patience changes that. Our explorer server, a perfectly ordinary cloud machine with no graphics card, sits about forty times past the line.

This is why a week of careful configuration work produced nothing. We were not misconfigured. We were on the wrong side of a line that configuration does not cross. One of the operators ended the argument in a single sentence: *stop fighting the CPU box, it is physics, not config*.

Every chain that resists specialised mining hardware faces some version of this trade. Make mining accessible to ordinary machines and verification stays cheap but farms move in. Make mining hostile

to farms and verification becomes expensive for everyone else. The receipt mechanism is the intended answer, and it is sound. It simply was not delivering.

Why the core developer sounds impatient

Some readers have watched the BTX maintainer be blunt in the channels and read it as temper. After this week we understand it differently.

Between 27 and 30 August, BTX published **six releases in four days**. That is verifiable in the public release history. One of those release notes opens with a stop banner in which the maintainer lists, in plain language, the defects in his own preceding releases: this one partitions nodes, this one deadlocks, this one stalls. Very few teams write that sentence about their own work, in public, while shipping the fix.

That is what the impatience actually is. A protocol is not an application. You do not iterate on consensus with a hotfix next sprint, because the rule shipped last week is already baked into every block that followed it. Withdrawing that difficulty rule after two days was not a stumble, it was the system catching a mistake before it hardened into permanent history. The cost of catching it was our week of downtime. The cost of not catching it would have been permanent.

The three who fixed it

Everything above was diagnosis, and none of it was ours alone. Over roughly six hours, three operators handed us measurements we could not have taken from inside our own infrastructure.

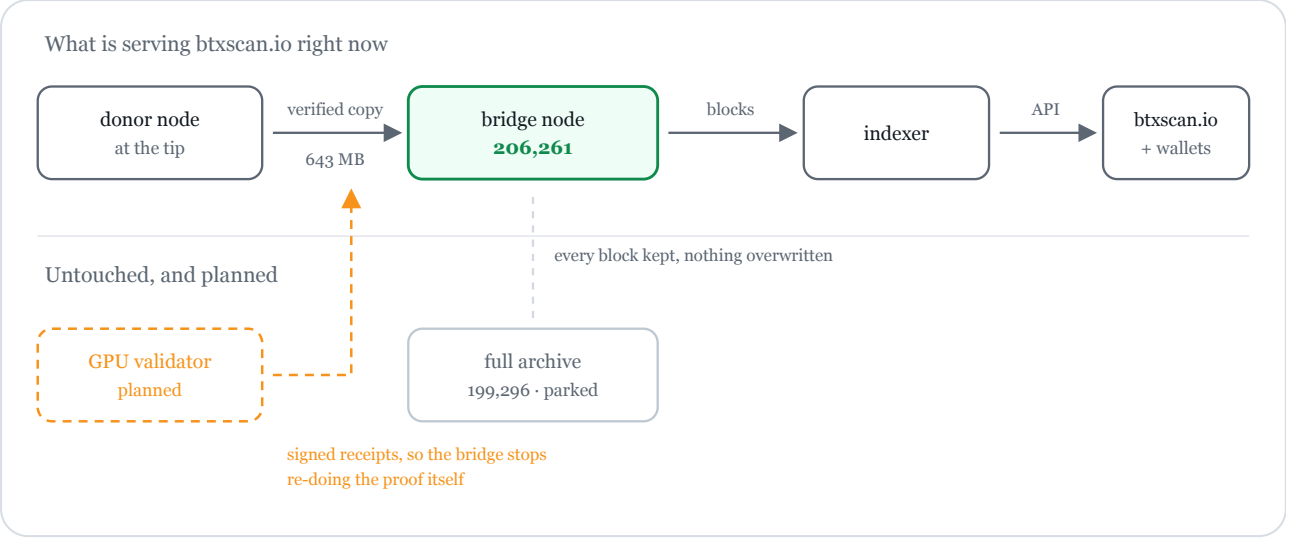
NGU found the fault underneath all the others. Every archive node on the network had migrated to the newer encrypted peer transport while our software still used the older handshake. The connection opens, the handshake dies in silence, and a healthy peer looks dead. One flag turned every unreachable archive into a working peer. He also traced exactly which recovery paths exist, corrected himself in writing when one of our measurements contradicted his advice, and told us plainly when the remaining answer was hardware rather than a setting.

Jpp named a rate limiter we had dismissed after misreading our own notes, worth an immediate 60 percent speedup on catch-up. He then caught a node inside our own shipped bootstrap list quietly serving a stale branch, and supplied the block height and both competing hashes as proof, a node that was wedging fresh installations before they could sync. Then he donated a verified copy of his chain data with a published checksum, and told us himself not to let anyone, including him, near our full archive with it.

jarekpiot had already published the signer-boundary recovery described above, proven across three machines of his own.

Every one of those contributions arrived with evidence attached: a block height, a hash, a checksum, a measured rate. Two of them corrected public claims we had made confidently and wrongly. That is not people being generous with us. That is a protocol's immune system working.

How the explorer came back



The rescue. A verified copy of the chain, checked against two independent sources before it was trusted, now runs as a second node beside the original archive, which was never modified. The dashed path is the permanent fix: our own graphics card doing the proof work and signing for it, so the server never has to.

Before that copy went anywhere near production we checked its fingerprint against the donor's published checksum, confirmed it contained chain data and nothing operational, and verified five block hashes spread across the range against two independent explorers. All five matched. Trust, then verify, then verify again using somebody else's node.

The result: the explorer moved **6,965 blocks in one operation**, from 199,296 to 206,261, and began serving current data again.

Where it stands, honestly

The explorer is current to within about two hours. It is not at the tip, and it drifts backwards by roughly sixty blocks an hour, because the bridge node hit the same arithmetic in the figure above the moment it had to check a block for itself. A borrowed copy of the chain catches you up; it does not let you follow.

The permanent fix is a graphics card in the explorer server, which is ordered and waiting on a quota approval, or the upstream release that repairs the signature supply, whichever lands first.

Meanwhile the site says all of this to visitors without being asked. Every page carries a banner naming the newest block it holds and how old that block is. The badge that used to claim LIVE now reads OLD DATA when the data is old. A public status page states plainly whether the network is fine and we are behind, or whether something worse is happening. It all clears itself automatically when the data is fresh again.

That banner is the only part of this week we are unreservedly glad about. The stale data was an accident. The confident green badge on top of it was a choice, and we made the wrong one for seven days.

Start with a node, not with mining

The advice we keep giving people who ask how to help BTX is the least glamorous one available: run a node first. Mine later, or never.

A node is not a lottery ticket, which is exactly why it is the right first step. It costs roughly the power of an LED bulb. It needs no graphics card simply to follow and serve the chain once the current fixes land. It gives you something you cannot buy: your own verified copy of the truth, so you never have to ask a website, including ours, whether your money is there. And it makes the network harder to censor while helping the next person's node bootstrap.

This week made the argument better than any essay could. The explorer that thousands of people rely on was blind for seven days. Everyone running their own node could see the chain correctly the entire time.

A proposal: node time

This is a proposal, not an announcement, and we would rather collect objections than applause.

BTX pays nothing for running a node, and we do not think it should start printing money for it. But the contribution is real and currently invisible. So what if uptime simply counted? A second of honest node time earns a point. Points would not be a coin, could not be bought, and would carry no promise of future value. They would be a public record of who held the network up, and the community would decide what that record is worth: standing, early access, a supporters board, something else entirely.

The appeal is that it rewards the least speculative behaviour available. Mining rewards hash power, which follows price. Node time rewards showing up, and those are different people. A young chain needs the second group most.

If you can see how this gets gamed, we would genuinely rather hear it now than after it is built.

What this week actually says about BTX

It would be easy to read seven days of downtime as evidence that BTX is fragile. We read it the other way, because of how the week ended rather than how it began.

A chain this young has a small number of people who genuinely understand its consensus internals, and nearly all of them are volunteers. In one night, three of them handed somebody else's infrastructure the exact measurements needed to repair it, corrected our public claims when we were wrong, corrected their own when the data disagreed, and refused the shortcuts that would have looked good and been unsafe. The operator who donated the chain copy is the same one who warned us not to let anyone push data onto our archive, including him.

The bugs behind this are real, they are upstream, and they are being fixed at a pace most projects never manage. Each is an ordinary growing pain of a protocol doing something genuinely new: a proof of work expensive enough to make GPU farms pointless is also expensive enough that a node

without the right hardware cannot keep up, and the machinery that lets modest machines take part anyway is exactly the machinery that broke.

What we owe in return is this report, the fixes pushed back into the software everyone else installs, and the discipline to keep saying *our data is behind* on our own front page for as long as it remains true.

Frequently asked questions

Why did btxscan show old data instead of an error?

Because from the inside, a node that has stopped following the chain looks identical to one that is perfectly caught up. Both answer instantly, both report a tip, and neither has any way to know that the rest of the network moved on without it. Our uptime checks all passed for seven and a half days. The site now compares its newest block against the wall clock and says on every page how old that block is, so this specific failure can never be silent again.

Were my coins ever at risk?

No. A block explorer reads the chain and never holds anyone's keys. What was out of date was our view of the chain, not anybody's balance. The same is true for a node that stalls: your coins live on the chain itself, not in the software that displays it.

Why did the explorer stop at exactly block 199,296?

BTX mainnet has two block signers and they sign alternate blocks. Our node was configured to trust only one of them. Block 199,296 happened to be signed by the key we trusted, and 199,297 was signed by the other one. A node in that state does not slow down gradually, it stops dead on the first block the other signer produced and stays there regardless of hardware or peers. Pinning both published keys is the fix.

Why can't you just copy the missing blocks from another node?

You can, and we did. A verified copy of another operator's chain data caught us up almost seven thousand blocks in one operation. What copying cannot do is let you follow. Every new block still has to be checked locally when it arrives, and that is the part our server could not afford.

Why does a node without a graphics card fall behind?

BTX's proof of work is deliberately expensive to compute so that industrial mining farms gain little advantage. The unavoidable consequence is that checking a block is nearly as expensive as producing one. We measured about 2.6 seconds per block on a current high-end GPU, about 30 seconds on Apple Silicon, and about 39 minutes on an ordinary server CPU. The network produces a block roughly every 58 seconds, so only the first two can keep up.

Is there not a shortcut so modest machines can keep up?

Yes, and it is the right design: well-equipped nodes check a block and sign a receipt, and lighter nodes trust the receipt instead of redoing the work. That mechanism exists in BTX today. The reason

it did not save us is an upstream bug in which those receipts are created but never published, so our node asked for them and heard nothing. The fix is in the next release.

How is this different from the earlier stuck-node problem you wrote about?

The earlier article was about telling apart a node that is healthy and waiting from one that is genuinely behind. This incident is what happens when the second case goes unnoticed for a week on infrastructure other people depend on. It is the sequel, and the lesson is that the diagnosis is worthless unless something is watching the clock and willing to say so in public.

Is my easyNode affected, and what should I do?

It depends on your platform, and easybtx.com/node always states what is published today. As of 7 September 2026: Linux is 0.6.20, carrying the current engine (BTX 0.34.6), following the live chain, validating blocks on your own NVIDIA card when you have a suitable one, and the updater offers it. macOS is 0.6.19, on that same engine, and follows the live chain, but 0.6.20 is a Linux-only release, so a Mac copy is offered nothing by the updater and has to be downloaded by hand. A Mac still on 0.6.12 or earlier stops at block 199,299, which is expected and not something you did, and updating starts it again. The native Windows app still carries an engine from before the rule change and cannot update itself; the working path on Windows is the Linux build under WSL2, as the node page explains. Your coins are unaffected on any platform, and the download pages now say this above the download button rather than after it.

Should I run a node at all after reading this?

More than before. Every person running their own node could see the chain correctly for the entire week our explorer was blind. That is the whole argument. A node needs roughly the power of an LED bulb, needs no graphics card simply to follow and serve the chain once the current fixes land, and it means you never have to ask a website, including ours, whether your money is there.

Published by easyBTX, which builds software for BTX and is not the BTX core team. This paper is educational research, not financial advice, and it is not an official BTX document. Read the live version and check the sources at easybtx.com/research/seven-days-dark.