Mining BTX on a Mac: 30 Shares, Two Pools, and a 25x Difficulty Week

A Mac can mine BTX again. Here is exactly how fast, exactly what it earns, and every place we got it wrong on the way.

AUGUST 22, 2026 · 12 MIN READ · EASYBTX.COM/RESEARCH/MINING-BTX-ON-A-MAC

ABSTRACT

We put one Apple M5 on the BTX network for 47 hours of continuous GPU work and measured everything: episode times, share rates on two different pools, the protocol traps that reject 100% of shares silently, and a network difficulty that rose 25x in three days.

On 20 August we put one ordinary Apple silicon Mac on the BTX network and left it there. Not a benchmark, not a rigged test: a real miner, on a real pool, submitting real shares for real credit.

Forty seven hours of continuous GPU work later: **4,283 episodes completed, 30 shares accepted, 12 rejected.** Every number below is measured on that machine or read from the chain, and every one of them is reproducible.

The same week, BTX network difficulty rose **25x in 72 hours**. That turns out to be the more interesting story, and the two are connected.

What a Mac is actually doing now

Since the MatMul v4.7 fork (our earlier article), BTX proof of work is not a hash loop. It is a matrix computation the network calls an **episode**, and the difference matters more than it sounds.

A classical miner tries billions of tiny guesses per second. You can stop it at any instant and lose almost nothing. An episode is one large indivisible piece of work. On our M5 it takes a **median of 40.9 seconds**, and it is strikingly consistent: 90% finish within 42.1 seconds, 99% within 43.3 seconds. Across 4,283 of them only three ever exceeded two minutes.

You cannot pause an episode. You cannot resume one. You cannot take 60% of an episode and get 60% of a share. It either completes and produces a result, or you have nothing.

That single property is responsible for nearly every difficulty we hit.

The difficulty story: 25x in three days

Here is the chain over the last week, sampled every 100 blocks and read from block headers.

when (UTC)	height	difficulty	block interval
18 Aug 13:53	192,928	1.64e-06	46 s
19 Aug 04:13	193,728	4.32e-06	63 s
19 Aug 23:20	194,728	1.21e-05	45 s
20 Aug 14:38	195,528	2.74e-05	76 s
21 Aug 12:21	196,528	4.83e-05	87 s
21 Aug 21:53	196,928	5.23e-05	86 s

Measured precisely: difficulty at the 72 hour mark was 2.076e-06 and is now 5.231e-05. That is **25.2x**, and it is not noise. Of the 15 sample steps in that window, 14 rose.

This is the retarget doing exactly its job. BTX targets a **90 second block** (`nPowTargetSpacing = 90` in consensus). Through 17 August, in the disruption following the fork, blocks were arriving in an average of **47.6 seconds**, roughly twice as fast as intended. Difficulty had to roughly double to correct that, and then kept climbing as hashrate kept arriving.

Watch the right-hand column and you can see it working:

day	mean block interval
17 Aug	47.6 s
18 Aug	61.2 s
19 Aug	67.1 s
20 Aug	72.9 s
21 Aug	82.5 s

The chain is converging on its 90 second target from below, and at the time of writing the difficulty climb was decelerating as it got there. The last four samples moved 18.1x, 18.6x, 19.3x, 20.1x: still rising, in ever smaller steps.

Update, 22 August 12:20 UTC: the deceleration reversed

We published the paragraph above on the morning of 22 August. It was true when measured and it stopped being true within hours, which is worth showing rather than quietly editing.

Difficulty is now **8.599e-05** at height 197,621. That is **1.64x higher than when this article went up**, and **41x** the level of 72 hours before it.

The per-sample steps had shrunk to 1.015x. They then went 1.111x, 1.095x, 1.124x. The reason is in the block interval: it had reached 87 seconds, close to the 90 second target, and then fell back to between 65 and 77 seconds. More hashrate arrived overnight, so the retarget is chasing again rather than settling.

Over the last 1,600 blocks, 35.3 hours, difficulty rose **2.27x** at a mean interval of 79.4 seconds.

The direction of the story is unchanged and if anything stronger: hashrate is still arriving on this chain, faster than the retarget has caught up with. The specific claim that the climb was slowing had a shelf life of about half a day, and we would rather date it than pretend we called it.

What this means for a Mac owner is less than you would think. Network difficulty governs how hard it is to find a *block*. Your pool sets its own, much easier, *share* target, and that is what decides how often you get credited. Difficulty changes what a share is ultimately worth, not how often you earn one.

Two pools, two protocols, and one trap that costs a whole afternoon

BTX has two live pools and they do not speak the same language.

Byron Bay speaks stratum, the protocol most miners already know. **btx-pool.com** uses its own JSON dialect: no request ids, no subscribe, just `hello`, `job` and `submit`. Neither is written down anywhere.

The single most expensive difference is a field name. The parent block's median time past is a required input to the proof of work: without it the solver cannot derive its seeds. Byron sends it as `parent_median_time_past`. btx-pool.com sends it as `parent_mtp`. Neither sends both.

Read the wrong one and you get `null`, the solver runs against wrong inputs, and **100% of your shares are rejected with no other symptom.** No error message, no partial failure, nothing to bisect. Just a miner that looks healthy and earns nothing.

The second difference decides whether a slow miner can earn at all. Because an episode is indivisible and takes 41 seconds against a block interval that has been between 47 and 87 seconds all week, a meaningful share of our episodes finish after the chain has moved on.

btx-pool.com credits any job on the same parent block. That is the right rule for this kind of work, and it is what makes a 41 second episode viable there.

Byron Bay raised a 60 second grace window in response to our earlier measurements, which was generous and well intentioned. We then measured whether it helped us, and found something worth reporting: **it cannot reach us, for a structural reason.** We captured Byron's own job notifications at two separate block heights and found that when it re-announces work at the same height, the message is byte for byte identical to the first one in every field except a single flag. It never issues a *new* job on the same parent. So "an older job on the same parent", the exact thing the grace window exists to forgive, is not an event that occurs on that wire. Every job change we can observe is a change of parent, which is genuinely dead work at any window width.

We have sent that finding, with the raw transcripts, to Byron's operator. It is not a complaint. It is the kind of thing you only learn by instrumenting both sides and comparing.

Where the pool's own difficulty bites

This one surprised us, and it is the number most likely to make a new user think something is broken.

Pools set a per-worker share target and adjust it automatically. Byron starts a new worker on a much easier target so the first shares arrive quickly, then tightens as it learns your speed. Split our run by which target was in force:

pool share target	episodes	accepted	one share per
eased, new worker	722	18	**40 episodes**, about 27 min
settled	2,328	11	**212 episodes**, about 2.4 h

So a new Mac sees a share every half hour or so, and then, once the pool has measured it, roughly every two and a half hours. Nothing broke. That is automatic difficulty working as designed, and the earnings per share rise to match. But if nobody tells you, the honest experience is "it worked for an hour and then it stopped", and you quit.

We got this wrong in our own release notes, which said a share takes "roughly fifty" episodes. That is true only during the honeymoon. We are correcting it.

Where we got stuck

The parts worth reading.

We shipped a hardware rule that was simply wrong. easyBTX 0.20.0 refused to mine on anything below an M5. We had reasoned our way there from the wrong file in the upstream source: a capabilities table that mentions the M5 class, rather than the resolver that actually admits a device, which reads no chip identity at all. Upstream's own published evidence includes a full speed production run on an M4 Max. We had turned away hardware that works. The fix was to stop reasoning about model names and **ask the hardware**: a half second self-qualification test that fails closed. Shipped in 0.20.1.

A working Mac displayed a hard zero. Because an episode is one work unit every 41 seconds, the honest rate is about 0.028 per second. The speed readout had no scale below whole units, so it rounded that to `0`, on a machine that was mining perfectly and having shares accepted. There is nothing more alarming a miner can show you. It now reads in episodes per minute, and shows a count of episodes attempted, because between shares that is the only thing on screen that moves.

A test that guarded against exactly this had never once run. We keep a check that downloads the real mining engine and verifies we can still read its output, written precisely because our own tests only ever proved we were self consistent. It failed on its first step, every time, from the day it was written: 28 runs, 28 failures, never reaching the engine. Two small bugs in the check itself. When we fixed it, it immediately found a real problem: the engine had started adding colour codes to the status line it prints, and our parser read a coloured field as a *missing* field, silently holding the previous value. Block height and share counters could sit frozen indefinitely with nothing logged.

The lesson generalises past mining. A guard nobody has watched succeed is not a guard. And a test written against a string you typed yourself proves only that you are consistent with yourself.

We got ourselves temporarily banned from a pool. Our reconnect logic backed off when a connection *failed*, but not when a pool accepted the socket and then closed it. That is not a rare case, it is what rate limiting looks like, and we answered it with 35 reconnects in a few seconds. Byron's edge refused us for about two and a half hours, entirely fairly. easyBTX now waits at least 60 seconds with jitter and doubles from there, matching the operator's published policy.

So: is it worth mining BTX on your Mac?

Honestly, as participation rather than income. One laptop against a network whose difficulty just rose 25x in three days is a small contributor, and we are not going to pretend otherwise.

What we can now say precisely, which nobody could a week ago:

- A qualifying Mac **does** solve BTX proof of work, at a median of 40.9 seconds per episode, with 99% of episodes inside 43.3 seconds
- Shares **are** accepted and credited: 30 of them, from one machine
- The gate is a **measurement**, not a model name, so more Macs qualify than we originally believed
- The app now tells you the truth about what it is doing, including when the answer is "nothing yet, and here is the counter proving it is trying"

The mining path is real. That was the open question, and it is closed.

Everything here is measured on one M5 running easyBTX 0.21.1 against Byron Bay between 20 and 22 August 2026, and on BTX block headers read from the public chain over the same period.

Frequently asked questions

Can a Mac mine BTX?

Yes. Since easyBTX 0.20.1 an Apple silicon Mac can solve the MatMul v4.7 proof of work and have shares credited by a pool. We have 30 accepted shares from one M5 across 47 hours of continuous GPU work.

Which Macs can mine?

Any Mac whose GPU passes a half-second self-qualification test on macOS 26.4 or newer. easyBTX asks the hardware directly rather than judging by the chip's name. We originally shipped a rule that refused anything below an M5, and that rule was wrong.

How fast is it?

One work unit, called an episode, takes a median of 40.9 seconds on an M5 and is remarkably consistent: 99% of episodes finish within 43.3 seconds.

How often does a Mac get a share?

It depends entirely on the target your pool sets. On Byron Bay's eased new-worker target we averaged one accepted share per 40 episodes, about 27 minutes. Once the pool's automatic difficulty settled, it became one per 212 episodes, about 2.4 hours.

Why did BTX difficulty rise so much this week?

The chain targets a 90 second block. After the post-fork disruption blocks were arriving in about 47 seconds, far too fast, so the retarget raised difficulty roughly 25x over 72 hours. Block intervals moved to about 82 seconds, close to target, and the climb is now slowing.

Does higher difficulty mean my Mac earns less?

It does not change how often your pool credits you a share, because a pool sets its own share target. It does change what the pool earns per block found, which flows through to what a share is worth.

Which pool should a Mac use?

easyBTX defaults to Byron Bay on macOS and that is the recommended choice today. It is the only live pool the Mac build can currently talk to.

Is mining on a laptop worth it?

Treat it as participation, not income. One Mac is a small contributor. What we measured is that it works, it is honest about what it is doing, and it costs you nothing but electricity to find out.
