Single random bit flip causes error in certificate transparency log (groups.google.com)

422 points by shmsr 10 months ago | hide | past | favorite | 248 comments

jart 10 months ago | next [–]

The push for crypto without ecc ram is a nonstop horror show. Software under normal circumstances is remarkably resilient to having its memory corrupted. However crypto algorithms are designed so that a single bit flip effectively changes all the bits in a block. If you chain blocks then a single bit flip in one block destroys all the blocks. I've seen companies like msps go out of business because they were doing crypto with consumer hardware. No one thinks it'll happen to them and once it happens they're usually too dim to even know what happened. Bit flips aren't an act of god you simply need a better computer.

ajross 10 months ago | parent | next [–]

> The push for crypto without ecc ram is a nonstop horror show

That's a bit hyperbolic.

First, ECC doesn't protect the full data chain, you can have a bitflip in a hardware flip flop (or latch open a gate that drains a line, etc...) before the value reaches the memory. Logic is known to glitch too.

Second: ECC is mostly designed to protect long term *storage* in DRAM. Recognize that a cert like this is a very short-term value, it's computed and then transmitted. The failure happened fast, before copies of the correct value were made. That again argues to a failure location other than a DRAM cell.

But mostly... this isn't the end of the world. This is a failed cert, which is a failure that can be reasonably easily handled by manual intervention. There have been many other mistaken certs distributed that had to be dealt with manually: they can be the result of software bugs, they can be generated with the wrong keys, the dates can be set wrong, they can be maliciously issued, etc... The system is designed to include manual validation in the loop and it works.

So is ECC a good idea? Of course. Does it magically fix problems like this? No. Is this really a "nonstop horror show"? Not really, we're doing OK.

aseipp 10 months ago | root | parent | next [–]

This is just learned helplessness because Intel were stingy as shit for over a decade and wanted to segregate their product lines. Error correction is literally prevalent in every single part of every PHY layer in a modern stack, it is an absolute must, and the lack of error correction in RAM is, without question, a ridiculous gap that should have never been allowed in the first place in any modern machine, especially given that density and bandwidth keeps increasing and will continue to do so.

When you are designing these systems, you have two options: you either use error correcting codes and increase channel bandwidth to compensate for them (as a result of injected noise, which is unavoidable), or you lower the transfer rate so much as to be infeasible to use, while also avoiding as much noise as you can. Guess what's happening to RAM? It isn't getting slower or less dense. The error rate is only going to increase. The people designing this stuff aren't idiots. That's why literally *every* other layer of your system builds in error correction. Software people do not understand this because they prefer to believe in magic due to the fact all of these abstractions result in stable systems, I guess.

All of the talk of hardware flip flops and all that shit is an irrelevant deflection. Doesn't matter. It's just water carrying and post-hoc justification because, again, Intel decided that consumers didn't actually need it a decade ago, and everyone followed suit. They've been proven wrong repeatedly.

Complex systems are built to resist failure. They wouldn't work otherwise. By definition, if a failure occurs, it's because it passed multiple safeguards that were already in place. Basic systems theory. Let's actually try building more safeguards instead of rationalizing their absence.

goodpoint 10 months ago | root | parent | next [–]

> By definition, if a failure occurs, it's because it passed multiple safeguards that were already in place.

Having worked on a good bunch of critical system, there aren't multiple safeguards in most hardware.

E.g. a multiplication error in a core will not be detected by an external device. Or a bit flip when reading cache, or from a storage device.

Very often the only real safeguard is to do the whole computation twice on two different hosts. I would rather have many low-reliability hosts and do computation twice than few high-reliability and very expensive host.

Unfortunately the software side is really lagging behind when it comes to reproducible computing. Reproducible builds are a good step in that direction and it took many decades to get there.

aseipp 10 months ago | root | parent | next [–]

Safeguards do not just exist at the level of technology but also politics, social structures, policies, design decisions, human interactions, and so on and so forth. "Criticality" in particular is something defined by humans, not a vacuum, and humans are components of all complex systems, as much as any multiplier or hardware unit is. The fact a multiplier can return an error is exactly in line with this: it can only happen after an array of other things allow it to, some of them not computational or computerized at all. And not every failure will also result in catastrophe as it did here. More generally such failures cannot be eliminated, because latent failures exist

everywhere even when you have TMR or whatever it is people do these days. Thinking there is any "only real safeguard" like quorums or TMR is *exactly* part of the problem with this line of thought.

The quote I made is actually very specifically is in reference to this paper, in particular point 2, which is mandatory reading for any systems engineer, IMO, though perhaps the word "safeguard" is too strong for the taste of some here. But focusing on definitions of words is besides the point and falls into the same traps this paper mentions: https://how.complexsystems.fail/

Back to the original point: is ECC the single solution to this catastrophe? No, probably not. Systems are constantly changing and failure is impossible to eliminate. Design decisions and a number of other decisions could have mitigated it and caused this failure to not be catastrophic. Another thing might also cause it to topple. But let's not pretend like we don't know what we're dealing with, either, when we've already built these tools and *know they work*. We've studied ECC plenty! You don't need to carry water for a corporation trying to keep its purse filled to the brim (by cutting costs) to proclaim that failure is inevitable and most things chug on, regardless. We already know that much.

twistedpair 10 months ago | root | parent | prev | next [–]

Failing to account for bit flips and HW failure is too common in web/service coding. Lookup how Google dropped a massive Big Table instance in prod, and traced it back to a cosmic ray bit flip that made a WRITE instruction into a DROP TABLE instruction.

I laugh when I compare my day to day coding to that of an avionics programmer in the aero industry.

goodpoint 10 months ago | root | parent | next [–]

The "web" coding words falls for things like leftpad, imagine talking about bit flips.

It's sad how immature the software industry can be. It's been around for "only" 60/70 years after all.

xpasky 10 months ago | root | parent | prev | next [–]

I couldn't find it. Do you have a reference?

twistedpair 10 months ago | root | parent | next [–]

There are some examples here [1]. Dig as I may, I cannot locate the original Google source I read about this ~3y ago.

1. https://www.usenix.org/system/files/conference/atc12/atc12-f...

xvector 10 months ago | root | parent | prev | next [–]

> Very often the only real safeguard is to do the whole computation twice on two different hosts.

Three different hosts, for quorum, right?

ajross 10 months ago | root | parent | next [–]

Depends on the system. In this case it seems like retries are possible after a failure, so two is sufficient to detect bad data. You need three in real time situations where you don't have the capability to go back and figure it out.

goodpoint 10 months ago | root | parent | next [–]

Spot on. Very often doing an occasional (very rare) retry is acceptable.

Sometimes doing the same processing twice can be also a way to implement safe(r) rolling updates.

sanctified384 10 months ago | root | parent | prev | next [–]

Two hosts is efficient. Do it twice on two different hosts and then compare the results. If there is a mismatch, throw it away and redo it again on 2 hosts. A total of 4 computations are needed. But only if the difference really was due to bit flips, the chance of which are exceedingly rare. In all the rest of the cases, you get away with two instead of three computations.

Shikadi 10 months ago | root | parent | prev | next [–]

People who tout this don't understand the probability of bit flips. It's measured in failures per _billion_ hours of operation. This matters a ton in an environment with thousands of memory modules (data centers and super computers) but you're lucky to experience a single ram bit flip more than once or twice in your entire life

Edit: there's some new (to me) information from real world results, interesting read. https://www.zdnet.com/article/dram-error-rates-nightmare-on-...

Looks like things are worse than I thought (but still better than most people seem to think). Interesting to note that the motherboard used affects error rate, and it seems that part of it is a luck of the draw situation where some dimms have more errors than others despite being the same manufacturer

kevin_thibedeau 10 months ago | root | parent | next [–]

Bit flips are *guaranteed* to happen in digital systems. No matter how low the probability is, it will never be zero. You can't go around thinking you're going to dodge a bullet because its unlikely. If it weren't for the pervasive use of error detection in common I/O protocols you would be subjected to these errors much more frequently.

Shikadi 10 months ago | root | parent | next [–]

We're talking specifically about ECC ram, which solves the specific problem caused by cosmic rays (and apparently bad motherboard design). IO protocol error correction is a totally different problem.

DevKoala 10 months ago | root | parent | prev | next [–]

I was going to comment, but you edited your post. Yes, it is worse than we usually think on the software side.

Shikadi 10 months ago | root | parent | next [–]

I still think it made sense up until now to not bother with it on consumer hardware, and even at this point. The probability of your phone having a software glitch needing a reboot is way higher. Now that it's practically a free upgrade? Should be included by default. But I still don't think it's nearly as nefarious as people make it out to be that it has been this way for so long

DevKoala 10 months ago | root | parent | next [–]

I don’t think they are an issue in a phone, but in a system like a blockchain that goes through so much effort to achieve consistency, the severity of the error is magnified hence the lesser tolerance for the error rate.

mtkd 10 months ago | root | parent | prev | next [–]

'stingy as shit' or maximising short-term shareholder value -- the hardware is possibly not the only broken model here

aseipp 10 months ago | root | parent | next [–]

You're probably right that I'm giving them a bit too much credit on that note. Ceterum censeo, and so on.

lima 10 months ago | root | parent | prev | next [–]

> This is a failed cert, which is a failure that can be reasonably easily handled by manual intervention.

This isn't a misissued cert that can be revoked, it permanently breaks the CT log in question since the error propagates down the chain.

tialaramex 10 months ago | root | parent | next [–]

Yes, this kills Yeti 2022. There's a bug referenced which refers to an earlier incident where a bitflip happened in the logged certificate data. *That* was just fixed. Overwrite with the bit flipped back, and everything checks out from then onwards.

But in this case it's the hash record which was flipped, which unavoidably taints the log from that point on. Verifiers will forever say that Yeti 2022 is broken, and so it had to be locked read-only and taken out of service.

Fortunately, since modern logs are anyway sharded by year of expiry, Yeti 2023 already existed and is unaffected. DigiCert, as log operator, could decide to just change criteria for Yeti 2023 to be "also 2022 is fine" and I believe they may already have done so in fact.

Alternatively they could spin up a new mythical creature series. They have Yeti (a creature believed to live in the high mountains and maybe forests) and Nessie (a creature believed to live in a lake in Scotland) but there are plenty more I'm sure.

ajross 10 months ago | root | parent | prev | next [–]

It doesn't break anything that I can see (though I'm no expert on the particular protocol). Our ability to detect bad certs isn't compromised, precisely because this was noticed by human beings who can adjust the process going forward to work around this.

Really the bigger news here seems to be a software bug: the CT protocol wasn't tolerant of bad input data and was trusting actors that clearly can't be trusted fully. Here the "black hat" was a hardware glitch, but it's not hard to imagine a more nefarious trick.

zinekeller 10 months ago | root | parent | next [–]

Your statement is, to be frank, non-sensical. The *protocol* itself isn't broken, at least for previous Yeti instances, certificate data *are* correctly parsed and rejected.* In this instance, it seems that the data is verified already pre-signing BUT was flipped mid-signing. This isn't the fault of how CT was designed but rather a hardware failure that requires correction there. (Or at least that's the likely explanation, it *could be* a software bug° *but* it will be a very consistent and obvious behaviour if it is indeed a software bug.)

On the issue of subsequent invalidation of all submitted certificates, this is prevented by submitting to at least 3 different entities (as of now, there's a discussion whether if this should be increased), so if a log is subsequently found to be corrupted, the operator can send a "operator error" signal to the browser, and any tampered logs are blacklisted from browsers. (Note that all operators of CT lists are members of CA/B forum, at least as of 2020. In standardisation phase, some individuals have operated their own servers but this is no longer true.)

* Note that if the cert details are nonsensical but technically valid, it is still accepted *by design*, because *all pre-certificates* are countersigned by the intermediate signer (which the CT log operator checks from known roots). If the intermediate is compromised, then the correct response is obviously a revocation and possibly distrust.

° At least the human-induced variety, you could say that this incident is technically a software bug that occurred due to a hardware fault.

caf 10 months ago | root | parent | next [–]

Presumably it's possible to code defensively against this sort of thing, by eg. running the entire operation twice and checking the result is the same before committing it to the published log?

makomk 10 months ago | root | parent | next [–]

Big tech companies like Google and Facebook have encountered problems where running the same crypto operation twice on the same processor deterministically or semi-deterministically gives the same incorrect result... so the check needs to be on done on separate hardware as well.

caf 10 months ago | root | parent | next [–]

I don't think that matters in this case, because the entire point of the log machine is to run crypto operations. If it has such a faulty processor it is basically unusable for the task anyway.

ajross 10 months ago | root | parent | prev | next [–]

So I'm learning about Yeti for the first time, but I don't buy that argument. Corrupt tranmitted data has been a known failure mode for all digital systems since they were invented. If your file download in 1982 produced a corrupt binary that wiped your floppy drive, the response would have been "Why didn't you use a checksumming protocol?" and not "The hardware should have handled it".

If Yeti can't handle corrupt data and falls down like this, Yeti seems pretty broken to me.

tptacek 10 months ago | root | parent | next [–]

Not handling corrupted data is kind of the point of cryptographic authentication systems. Informally and generally, the first test of a MAC or a signature of any sort is to see if it fails on arbitrary random single bit flips and shifts.

The protocol here seems to have done what it was designed to do. The corrupted shard has simply been removed from service, and would be replaced if there was any need. The ecosystem of CT logs foresaw this and designed for it.

ajross 10 months ago | root | parent | next [–]

So... Yeti isn't broken then? Seems like the protocol does handle it? Seems like there's some confusion on this point in this thread.

detaro 10 months ago | root | parent | next [–]

The Yeti2022 log is corrupted due to the random event. This has been correctly detected, and is by design and policy not fixable, since logs are not allowed to rewrite their history (ensuring that they don't is very much the point of CT). That the log broke is annoying but not critical, and the consequences are very much CT working as intended.

You can argue if the software running the log should have verified that it calculated the correct thing before publishing it, but that's not a protocol concern.

tptacek 10 months ago | root | parent | prev | next [–]

I think Yeti2022 is just the name for one instance of the global CT log? Nick Lamb could probably say more about this; I understand CT mostly in the abstract, and as the total feed of CT records you surveil for stuff.

detaro 10 months ago | root | parent | next [–]

Yeti is one of the CT logs (and Yeti2022 the 2022 shard of it, containing certs that expire in 2022). CT logs are independent of each other, there is not really a "global log", although the monitoring/search sites aggregate the data from all logs. Each certificate is added to multiple logs, so the loss of one doesn't cause a problem for the certs in it. (Maybe it's also possible to still trust Yeti2022 for the parts of the log that are well-formed, which would decrease the number of impacted certs even more, not familiar enough with the implementations for that)

hda111 10 months ago | root | parent | prev | next [–]

Using ECC is a no-brainer. Even the Raspberry Pi 4 has ECC RAM. It’s not particularly expensive and only a artificial limitation Intel has introduced for consumer products.

dijit 10 months ago | root | parent | next [–]

Thought you were wrong.

Checked it out.

You were right..

Under "1.2. Features"

https://datasheets.raspberrypi.org/cm4/cm4-datasheet.pdf

https://datasheets.raspberrypi.org/rpi4/raspberry-pi-4-produ...

jml7c5 10 months ago | root | parent | next [–]

(See my reply to parent.)

jml7c5 10 months ago | root | parent | prev | next [–]

The Pi 4 uses DRAM with on-die ECC, which AFAIK does not provide any means of reporting errors (corrected or uncorrected) to the SoC's memory controller. It is effectively a cost-saving measure to improve DRAM yields. As such, it does little to guarantee that there are no memory errors.

ineedasername 10 months ago | root | parent | prev | next [–]

That's a bit hyperbolic.

Would ECC have avoided the issue in this case? If so then it's hard not to agree that is should be considered a minimum standard. It looks like Yeti 2022 isn't going to survive this intact, and while they can resolve the issues in other ways, not everyone will always be so fortunate and ECC is a relatively small step to avoid a larger problem.

test_epsilon 10 months ago | root | parent | prev | next [–]

> That's a bit hyperbolic.

I agree. That said,

> First, ECC doesn't protect the full data chain, you can have a bitflip in a hardware flip flop (or latch open a gate that drains a line, etc...) before the value reaches the memory. Logic is known to glitch too.

Of course. DRAM ECC protects against errors in the DRAM cells. That doesn't mean other components don't have other strategies for reducing errors which can form a complete chain.

Latches and arrays and register files often have parity (where data can be reconstructed) or ECC bits, or use low level circuits that themselves are hard or redundant enough to achieve a particular target UBER for the full system.

> Second: ECC is mostly designed to protect long term storage in DRAM. Recognize that a cert like this is a very short-term value, it's computed and then transmitted. The failure happened fast, before copies of the correct value were made. That again argues to a failure location other than a DRAM cell.

Not necessarily. Cells that are sitting idle other than refresh have certain error profiles, but ones under constant access. Particularly "idle" cells that are in fact being disturbed by adjacent accesses certainly have a non-zero error profile and need ECC too.

My completely anecdotal guess would be this error is at least an order of magnitude more likely to have occurred in non-ECC memory (if that's what was being used) rather than any other path to the CPU or cache or logic on the CPU itself.

noxer 10 months ago | root | parent | prev | next [–]

If a system is critical it should run on multiple machines in multiple locations and "sync with checks" kinda like the oh so hated and totally useless blockchains.

Then if such a bit-flip would occur it would never occur on all machines at the same time in the same data. And on top of that you could easy make the system fix itself if something like that happens (simply assume the majority of nodes didn't have the bit-flip) or in worst case scenario it could at least stop rather than making "wrong progress"

I have no clue what this particular log need for "throughput specs" but I assume it would be easily achievable with current DLT.

ineedasername 10 months ago | root | parent | next [–]

Safety critical systems are not a good fit for a blockchain-based resolution to the Byzantine General problem. Safety critical systems need extremely low latency to resolve the conflict fast. So blockchain is not going to be an appropriate choice for all critical applications when there are multiple low-latency solutions for BFT at a very low ms latency and IIRC, microseconds for avionics systems.

noxer 10 months ago | root | parent | next [–]

Not sure what you mean with "safety critical". I made no such assumption. Also since the topic is about a (write-only) log it probably doesn't need such low latency. What it more likely needed is final states so once a entry is made and accepted it must be final and ofc correct. DLTs can do this distributed and self-fixing i.e. a node that tries to add fault data is overruled and can never get a confirmation for a final state that later would not be valid.

Getting the whole decentral system to "agree" will never be fast unless we have quantum tech. There is simply no way servers around the glob could communicate in microseconds even if all communication would be the

speed of light and processing would be instant. It would still take time. In reality such system need seconds which is often totally fine. As long as everyone only relies on the data that has been declared final.

ineedasername 10 months ago | root | parent | next [–]

You said *If a system is critical*

I thought you were making a more general statement about all critical systems, that's all. And since many critical systems have a safety factor in play, I wanted to distinguish them as not always being a good target for a Blockchain solution to the problems of consensus.

Blockchain is a very interesting solution to the problem of obtaining consensus in the face of imperfect inputs, there are other options so, like anything else, you choose the right tool for the job. My own view is that-- given other established protocols, blockchain is going to be overkill for dealing with some types of fault tolerance. It is a very good fit for applications where you want to minimize relying on the trust of humans. (And other areas too, but right now I'm just speaking of the narrow context of consensus amid inconsistent inputs)

noxer 10 months ago | root | parent | next [–]

Critical that the system operates/keeps operating/does not reach an invalid state. It could be for safety but in general its more to avoid financial damage. Downtime of any kind usually result in huge financial loses and people working extra shifts. This was my main point.

>...blockchain is going to be overkill for dealing with some types of fault tolerance

But it this case likely it isn't. The current systen already works with a chain of blocks its just lacks the distributed checking and all that stuff. But "blckchains" aren't some secret sauce in this case it just an way to implement a distributed write-only database with filed proven tech. It can be as lightweight as it any other solution. The consensus part is completely irrelevant anyway because all nodes are operated by one entity. But due to the use case (money/value) of modern DLT ("blockchains") they are incredible reliable by design. The oldest DLTs that uses FBA (instead of PoW/PoS) are running since 9+ years without any error or downtime. Recreating a similar reliable system would be month and month of work followed by month of testing.

ineedasername 10 months ago | root | parent | next [–]

Yep, pretty much agreed. Whatever anyone may think if crypto *currencies* blockchains are essentially a different technology where coins are just 1 application. I'm kind of "meh" on crypto currencies (not anti, just think they need a while more to mature) but trustless consensus is a significant innovation in its own right.

dundarious 10 months ago | root | parent | prev | next [–]

I haven't thought about whether an actual blockchain is really the best solution, but the redundancy argument is legitimate. We've been doing it for decades in other systems where an unnoticed bit flip results in complete mission failure, such as an Apollo mission crash.

I'm not really sure what Yeti 2022 is exactly, so take this with heaps of salt, but it seems like this is a "mission failure" event -- it can no longer continue, except as read only. Crypto systems, even more than physical systems like rockets, suffer from such failures after just "one false step". Is the cost of this complete failure so low that it doesn't merit ECC? Extremely doubtful. Is it so low that it doesn't merit redundancy? More open for debate, but plausibly not.

I know rockets experience more cosmic rays and their failure can result in loss of life and (less importantly) losing a lot more money, and everything is a tradeoff -- so I'm not saying the case for redundancy is water tight. But it's legitimate to point out there is an inherent and it seems under-acknowledged fragility in non-redundant crypto systems.

noxer 10 months ago | root | parent | next [–]

>I haven't thought about whether an actual blockchain is really the best solution...

Most likely not. But the tech behind FBA (Federated Byzantine Agreement) distributed ledgers would make an extremely reliable system that can handle malfunction of hardware and large outages of nodes. And since this is a write-only log and only some entities can write to it, it could be implemented with permission so that the system doesn't have to deal with attacks that public blockchain would face.

tialaramex 10 months ago | root | parent | next [–]

Technically everyone can write to it. However you can only write certain specific things.

In the case of Yeti 2022 you were only able to log (pre-)certificates signed by particular CAs trusted in the Web PKI, which were due to expire in the year 2022.

In practice the vast majority of such logging is done by issuing CAs, as part of their normal operations. But it is possible (and is done purposefully, at least sometimes) to obtain certificates which have not been logged. These certificates, of course, won't work in Chrome or Safari because there's no proof they were logged. But you can log them yourself, and get SCTs and show those Just In Time™.

This is only an interesting technical option if you have both the need to do it for some reason and the capability to send SCTs that aren't baked inside your certificates. The vast majority of punters just have the CA do all this for them and the certificate they get has SCTs baked right inside it so there's no technical changes for them at all, they needn't know CT exists.

Because the CA's critical business processes depend on writing to logs, they need a formal service level agreement assuring them that some logs they use will accept their writes in a timely fashion and meet whatever criteria, but you as an individual don't need this, you don't care if the log you wanted to use says it's unavailable for 4 hours due to maintenance.

noxer 10 months ago | root | parent | next [–]

Thats pretty much the default state of any blockchain like systems. You need a private key to write to it. Its just that in most public blockchains can have an infinite amount of new private keys can be generated an and some kind of token is attached to it. For a log none of that would be needed. A central operator could hand out keys to anyone who should be able to write to it and for all other its read-only. And ofc the key alone would still not allow someone to write invalid data.

tialaramex 10 months ago | root | parent | prev | next [–]

> I'm not really sure what Yeti 2022 is exactly

Sometimes there are problems with certificates in the Web PKI (approximately the certificates your web browser trusts to determine that this is really news.ycombinator.com, for example). It's a lot easier to discover such problems early, and detect if they've really stopped happening after someone says "We fixed it" if you have a complete list of every certificate.

The issuing CAs could just have like a huge tarball you download, and promise to keep it up-to-date. But you just know that the same errors of omission that can cause the problem you were looking for, can cause that tarball to be lacking the certificates you'd need to see.

So, some people at Google conceived of a way to build an append-only log system, which issues signed receipts for the items logged. They built test logs by having the Google crawler, which already gets sent certificates by any HTTPS site it visits as part of the TLS protocol, log every new certificate it saw.

Having convinced themselves that this idea is at least basically viable, Google imposed a *requirement* that in order to be trusted in their Chrome browser, all certificates must be logged from a certain date. There are some fun chicken-and-egg problems (which have also been solved, which is why you didn't need to really *do* anything even if you maintain an HTTPS web server) but in practice today this means if it works in Chrome it was logged. This is *not* a policy requirement, not logging certificates doesn't mean your CA gets distrusted - it just means those certificates won't work in Chrome until they're logged and the site presents the receipts to Chrome.

The append-only logs are operated by about half a dozen outfits, some you've heard of (e.g. Let's Encrypt, Google itself) and some maybe not (Sectigo, Trust Asia). Google decided the rule for Chrome is, it must see at least one log receipt (these are called SCTs) from Google, and one from any "qualified log" that's not Google.

After a few years operating these logs, Google were doing fine, but some other outfits realised hey, these logs just grow, and grow, and grow without end, they're append-only, that's the whole point, but it means we can't trim 5 year old certificates that nobody cares about. So, they began "sharding" the logs. Instead of creating Kitten log, with a bunch of servers and a single URL, make Kitten 2018, and Kitten 2019, and Kitten 2020 and so on. When people want to log a certificate, if it expires in 2018, that goes in Kitten 2018, and so on. This way, by the end of 2018 you can switch Kitten 2018 to read-only, since there can't be new certificates which have already expired, that's nonsense. And eventually you can just switch it off. Researchers would be annoyed if you did it in January 2019, but by 2021 who cares?

So, Yeti 2022 is the shard of DigiCert's Yeti log which only holds certificates that expire in 2022. DigiCert sells lots of "One year" certificates, so those would be candidates for Yeti 2022. DigiCert also operate Yeti 2021 and 2023 for example. They also have a "Nessie" family with Nessie 2022 still working normally.

Third parties run "verifiers" which talk to a log and want to see that *is* in fact a consistent append-only log. They ask it for a type of cryptographic hash of all previous state, which will inherit from an older hash of the same sort, and so on back to when the log was empty. They also ask to see all the certificates which were logged, if the log operates correctly, they can calculate the forward state and determine that the log is indeed a record of a list of certificates, in order, and the hashes match. They remember what the log said, and if it were to subsequently contradict itself, that's a fatal error. For example if it suddenly changed its mind about which certificate was logged in January, that's a fatal error, or if there was a "fork" in the list of hashes, that's a fatal error too. This ensures the append-only nature of the log.

Yeti 2022 failed those verification tests, beginning at the end of June, because in fact it had somehow logged one certificate for *.molabtausnipnimo.ml but it has mistakenly calculated a SHA256 hash which was one bit different and then all subsequent work assumed the (bad) hash was correct. There's no way to rewind and fix that.

In principle if you knew a way to make a bogus certificate which matched that bad hash you could overwrite the real certificate with that one. But we haven't the faintest idea how to begin going about that so it's not an option.

So yes, this was mission failure for Yeti 2022. This log shard will be set read-only and eventually decommissioned. New builds of Chrome (and presumably Safari) will say Yeti 2022 can't be trusted past this failure. But the overall Certificate Transparency system is fine, it was designed to be resilient against failure of just one log.

admax88q 10 months ago | root | parent | next [–]

Conceivably you could also fix this by having all verifiets special case this one certificate in their verification software to substitute the correct hash?

Obviously that's a huge pain but in theory it would work?

psanford 10 months ago | root | parent | next [–]

You really want to make everyone special case this because 1 CT log server had a hardware failure?

This is not the first time a log server had to be removed due to a failure, nor will it be the last. The whole protocol is designed to be resilient to this.

What would be the point of doing something besides following the normal procedures around log failures?

admax88q 10 months ago | root | parent | next [–]

I was just asking to check if I understood the problem correctly.

tialaramex 10 months ago | root | parent | next [–]

Yes, in principle you could special case all the verifiers, although that's an open set, the logs are public.

dundarious 10 months ago | root | parent | prev | next [–]

Thank you, that's an excellent description. The CT system as a whole does appear to have ample redundancy, with automated tools informing manual intervention that resolves this individual failure.

ericbarrett 10 months ago | parent | prev | next [–]

In my younger days I worked in customer support for a storage company. A customer had one of our products crash (kernel panic) at a sensitive time. We had a couple of crack engineers I used to hover around to try to pick up gdb tricks so I followed this case with interest—it was a new and unexpected panic.

Turns out the crash was caused by the processor *taking the wrong branch*. Something like this (it wasn't Intel but you get the picture):

```
test edx, edx
jnz $something_that_expects_nonzero_edx
```

Well, edx was 0, but the CPU jumped anyway.

So yeah, sometimes ECC isn't enough. If you're really paranoid (crypto, milsec) you should execute multiple times and sanity-test.

astrange 10 months ago | root | parent | next [–]

I saw a kernel panic because of this _yesterday_. (Crash where the failing instruction in memory was 1 bit different from on disk.)

ericbarrett 10 months ago | root | parent | next [–]

Sounds like a regular RAM bitflip, doesn't it?

astrange 10 months ago | root | parent | next [–]

Sure, but even with those you'd usually expect it to happen to data, which is a lot larger.

EthanHeilman 10 months ago | parent | prev | next [–]

>If you chain blocks then a single bit flip in one block destroys all the blocks. I've seen companies like msps go out of business because they were doing crypto with consumer hardware.

If it is caused by a single bitflip you know the block in which that bitflip occurred and can try each bit until you find the right bit. This is an embarrassing parallel problem. Let's say you need to search 1 GB space for a single bit flip. That only requires that you test 8 billion bit flips. Given the merklized nature of most crypto, you will probably be searching a space far smaller than 1 GB.

>Bit flips aren't an act of god you simply need a better computer.

Rather than using hardware ECC, you could implement ECC in software. I think hardware ECC is good idea, but you aren't screwed if you don't use it.

The big threat here is not the occasional random bit flips, but adversary caused targeted bit flips since adversaries can bit flip software state that won't cause detectable failures but will cause hard to detect security failures.

lima 10 months ago | parent | prev | next [–]

Distributed, byzantine fault tolerant state machines solve that. At worst, a single node will go out of sync.

noxer 10 months ago | root | parent | next [–]

This is a great way to say "blockchain" without getting guaranteed down votes ;)

InspiredIdiot 10 months ago | root | parent | next [–]

Yes, and also contrary to Wikipedia it provides a solid use case for private blockchain https://en.wikipedia.org/wiki/Blockchain#Disadvantages_of_pr...

I think it is instructive to ask "why doesn't this mention guarantee downvotes?" because I don't think it's just cargo-culting. I doubt that many of those objecting to blockchain are objecting to byzantine fault tolerance, DHT, etc. Very high resource usage in the cost function, the ledger being public and permanent (long term privacy risk), negative externalities related to its use in a currency... These are commonly the objections I have and hear. And they are inapplicable.

Extending what the Wikipedia article says, it's basically glorified database replication. But it also replicates and verifies the calculation to get to that data so it provides far greater fault tolerance. But since it is private you get to throw out the adversarial model (mostly the cost function) and assume failures are accidental, not malicious. It makes the problem simpler and lowers the stakes versus using blockchain for a global trustless digital currency so I don't think we should be surprised that it engenders less controversy.

eitland 10 months ago | root | parent | prev | next [–]

You made me smile, however:

I'm one of those who can easily downvote blockchain stuff mercilessly. It is not reflexively though: I reserve it for dumb ideas, it just so happens that most blockchain ideas I see come off as dumb and/or as an attempted rip off.

matthewdgreen 10 months ago | root | parent | prev | next [–]

The poster isn't wrong. An entire chain shouldn't die because of a memory error in one node.

goodpoint 10 months ago | root | parent | prev | next [–]

No, 99% of the times you just need to do a computation twice on different hosts. You don't need quorum or other form of distributed consensus.

omegalulw 10 months ago | parent | prev | next [–]

> Software under normal circumstances is remarkably resilient to having its memory corrupted

Not really? What you are saying applies to anything that uses hashing of some sort, where the goal by design is to have completely different outputs even with a single bit flip.

And "resilience" is not a precise enough term? Is just recovering after errors due to flips? Or is it guaranteeing that the operation will yield correct output (which implies not crashing)? The latter is far harder.

morelikeborelax 10 months ago | parent | prev | next [–]

I used to sell workstations and servers years ago and trying to convince people they needed ECC Ram and that it was just insurance for (often) just the price of the extra chips on the DIMMs was a nightmare.

The amount of uninformed and inexperienced counter arguments online suggesting it was purely Intel seeking extra money (even though they didn't sell RAM) was ridiculous.

I never understood why there was so much push back from the consumer world commenting on something they had no idea about. Similar arguments for why would you ever need xxGB of RAM while also condemning the incorrect 640kb RAM Bill Gates comment.

fulafel 10 months ago | parent | prev | next [–]

How did the company go out of business?

agilob 10 months ago | parent | prev | next [–]

>companies like msps go out of business because they were doing crypto with consumer hardware

any story on this? can you elaborate or add something to read about it?

cletus 10 months ago | prev | next [–]

Cosmic-ray bit flipping is real and it has real security concerns. This also makes Intel's efforts at market segmentation by not having ECC support in any consumer CPUs [1] even more unforgivable and dangerous.

Example: bitsquatting on domains [2].

[1]: https://arstechnica.com/gadgets/2021/01/linus-torvalds-blame...

[2]: https://nakedsecurity.sophos.com/2011/08/10/bh-2011-bit-squa...

deckard1 10 months ago | parent | next [–]

AMD is also doing market segmentation on their APU series of Ryzen. PRO vs. non PRO.

It should also be mentioned that Ryzen is a consumer CPU and you're stuck (mostly) with consumer motherboards, none of which tell you the level of ECC support they provide. Some motherboards do nothing with ECC! Yes, they "work" with it. But that means nothing. Motherboards need to say they correct single bit errors and detect double bit errors.[1] None of the Ryzen motherboards say this. Not a single one that I could find.

Maybe Asrock Rack, but that's a workstation/server motherboard. Which is also going for $400-600. You think that $50 Gigabyte motherboard is doing the right thing regarding ECC? That's a ton of faith right there.

Consumer Ryzen CPUs may support ECC, but that's meaningless without motherboards testing it and documenting their support of it. So no, Ryzen really does not support ECC if you ask me.

[1] https://cr.yp.to/hardware/ecc.html

sroussey 10 months ago | root | parent | next [–]

DDR5 has a modicum of ECC so things might slowly improve. Maybe DDR6 will be full ECC and we will no longer have this market segmentation in the 2030s. Wow, that's a long time though.

PS: why didn't apple do the right thing with the M1? My guess is the availability of the memory which again points to changing it at the memory spec level.

wmf 10 months ago | root | parent | next [–]

The M1 uses LPDDR4x which can't conveniently support ECC.

belter 10 months ago | parent | prev | next [–]

Indeed. See this one for the results of Bit-squatting the Windows Update domains:

"DEFCON 19: Bit-squatting: DNS Hijacking Without Exploitation"

https://youtu.be/aT7mnSstKGs

judge2020 10 months ago | parent | prev | next [–]

Also, ECC ram is technically supported on AMD's recent consumer platform, although it's not advertised as so since they don't do validation testing for it.

willis936 10 months ago | root | parent | next [–]

I've read that reporting of ECC events is not supported on consumer Ryzen. It's not a complete solution and since unregistered ECC is being used, how can you even be sure the memory controller is doing any error correction at all?

Someone would need to induce memory errors and publish their results. I'd love to read it.

This is 4 years old now, but does produce some interesting results.

https://hardwarecanucks.com/cpu-motherboard/ecc-memory-amds-...

theevilsharpie 10 months ago | root | parent | next [–]

> This is 4 years old now, but does produce some interesting results.

> https://hardwarecanucks.com/cpu-motherboard/ecc-memory-amds-...

The author of that article doesn't have hands-on experience with ECC DRAM, and mistakenly concludes that ECC on Ryzen is unreliable because of a misunderstanding of how Linux behaves when it encounters an uncorrected error. However, the author at least includes screenshots which show ECC functionality on Ryzen working properly.

> ...since unregistered ECC is being used, how can you even be sure the memory controller is doing any error correction at all?

ECC is performed by the memory controller, and requires an extra memory device per rank and 8 extra data bits, which unbuffered ECC DIMMs provide.

Registered memory has nothing to do with ECC (although in practice, registered DIMMs almost always have ECC support). It's simply a mechanism to reduce electrical load on the memory controller to allow for the usage of higher-capacity DIMMs than what unbuffered DIMMs would allow.

With respect to Ryzen, Zen's memory controller architecture is unified, and owners of Ryzen CPUs use the same memory controller found in similar-generation Threadripper and EPYC processors (just fewer of them). Although full ECC support is not required on the AM4 platform specifically (it's an optional feature that can be implemented by the motherboard maker), it's functional and supported if present. Indeed, there are several Ryzen motherboards aimed at professional audiences where ECC is an explicitly advertised feature of the board.

willis936 10 months ago | root | parent | next [–]

The AM4 platform does not support ECC reporting. No motherboard can fix this.

https://www.servethehome.com/asrock-rack-x570d4u-2l2t-review...

theevilsharpie 10 months ago | root | parent | next [–]

ECC reporting is part of the memory controller (which is unified across all Zen architecture parts), and is fully supported and functional. You can see the reporting working as expected within the Hardware Canucks article linked in the grand parent.

The article you linked mentions that ECC reporting is not working with the on-board IPMI controller (which presumably means that ECC events aren't being logged in the SEL). While that might be a limitation of this board (and other IPMI-equipped AM4 boards), reporting from within the operating system will still work.

my123 10 months ago | root | parent | prev | next [–]

Not on all of it. They only don't lock it down on CPUs, but not on APUs.

On AMD SoCs with integrated GPUs, ECC is not usable except if you buy the Ryzen Pro variant. And those are all laptops and most desktops...

luckman212 10 months ago | root | parent | next [–]

Does e.g. the new Ryzen 7 5800U part support ECC?

my123 10 months ago | root | parent | next [–]

No, it does not. The Ryzen 7 PRO 5850U does.

rkangel 10 months ago | parent | prev | next [–]

I don't understand the issue with market segmentation here. I can absolutely see the reason why all of my servers should have ECC, but I don't see why my gaming PC should (or even my work development machine). What's the worst case impact of the (extremely rare) bit-flip on one of those machines?

philjohn 10 months ago | parent | prev | next [–]

At least things are starting to move in the right direction with DDR5, although it's still not perfect.

agwa 10 months ago | prev | next [–]

OP here. Unless you work for a certificate authority or a web browser, this event will have zero impact on you. While this particular CT log has failed, there are many other CT logs, and certificates are required to be logged to 2-3 different logs (depending on certificate lifetime) so that if a log fails web browsers can rely on one of the other logs to ensure the certificate is publicly logged.

This is the 8th log to fail (although the first caused by a bit flip), and log failure has never caused a user-facing certificate error. The overall CT ecosystem has proven very resilient, even if a bit flip can take out an individual log.

(P.S. No one knows if it was really a cosmic ray or not. But it's almost certainly a random hardware error rather than a software bug, and cosmic ray is just the informal term people like to use for unexplained hardware bit flips.)

sillysaurusx 10 months ago | parent | next [–]

Is it possible to give an example other than “cosmic ray”? I know it’s an informal short hand, but it also raises the question of what’s actually causing these flips.

Is it just random stray electrons that happen to stray too far while traveling through the RAM? Very interesting to me.

DominoTree 10 months ago | root | parent | next [–]

Old DRAM/SRAM simply becoming flaky, overheating, physical connection issues, faulty power - there are a ton of reasons these flips happen, and I would think that in most cases they are entirely unnoticed or simply observed as an application crashing, rather than being recognized as directly caused by bit flips.

BlueTemplar 10 months ago | parent | prev | next [–]

Not for specific bit flips, but on average you get more bit flips at higher altitudes and on systems with less shielding (being underground helps).

gbrown_ 10 months ago | prev | next [–]

Can't say I really like the title as it comes across as an absolute statement whereas the bit flip could have happened for any number of unknown reasons.

I saw a similar take on Twitter and whilst root causing such things (especially when it's a single occurrence) isn't always possible, shrugging and saying "cosmic rays" should be the last thing to posit not one of the first.

juped 10 months ago | parent | next [–]

"cosmic rays" are more of a well-known term of art for "single bit flipped with unknown hardware cause" than a reference to literal cosmic rays

drumbaby 10 months ago | root | parent | next [–]

I remember an older less-computer savvy gentlemen asking for support because "the program isn't working", when after a few questions we realized his computer won't boot up (screen dark, etc.). We thought his terminology was all screwed up. But now I realize he just lacked the necessary PR skill. He should have said that "the program isn't working" is a well-known term of art for power users such as himself. The fact that people who actually have a clue about computing find this imprecise term upsetting if there problem. It so happens that aside from developing software, I'm physicist working on particle physics (specifically, I make my living in industry from cosmic rays). So I can assure you that "cosmic rays" actually mean something. Something very specific. Books have been written full of specific intelligent things we can say about cosmic rays. If you go an appropriate it to describe a whole bunch of phenomena because you can't be bothered to distinguish between them, you're in the exact same boat as that gentlemen from the start of the story. Using wrong terms prevents understanding, as can be seen in all the stories linked elsethread so far. For example, using thinner silicon typically reduces the rate of Single-Event Upsets (the malfunctions caused by cosmic rays) but using smaller silicon components typically increases the rate of malfunctions due to quantum fluctuations. The latter typically happen in specific hardware that we manufactured not-quite-as-well as the rest. SEUs happen in the same rate in all hardware.

quesera 10 months ago | root | parent | next [–]

Go easy on the old guy. It sounds to me like he was completely correct, at the appropriate level of abstraction for him.

And as for cosmic rays...this may be a sensitive topic for you (or maybe you're just in a condescending mood), so I'll tread carefully, but seems simple to me.

The actual cause of the problem, given the instrumentation in place at the time of incident, is unknowable. But it might have been a subatomic particle. It absolutely definitively is, sometimes.

Metonymy might be imprecise, but it's human. I'm not sure what standard you intend to hold commenters here (or old guys with computers) to.

shawnz 10 months ago | root | parent | prev | next [–]

> If you go an appropriate it to describe a whole bunch of phenomena because you can't be bothered to distinguish between them

Cosmic ray is the appropriate colloquial term. Just like "bug" is the appropriate term to describe computer problems that have nothing to do with insects. It's a well established colloquialism and not simply terminology made up on the spot like in your example.

noduerme 10 months ago | root | parent | prev | next [–]

You're saying you have to engineer chip components so they're small enough not to be hit as often by cosmic rays, but large enough to avoid quantum fluctuations? Are quantum fluctuations something Intel is dealing with regularly as they get down under 3nm?

eloff 10 months ago | root | parent | next [–]

Yes, quantum tunneling (an electron "teleporting" into or out of transistors) has been an issue everyone has had to design around for multiple process nodes now.

solarkraft 10 months ago | root | parent | prev | next [–]

Interesting, because I was thinking of ... cosmic rays.

juped 10 months ago | root | parent | next [–]

sometimes it really is cosmic rays! but you usually can't know after the fact

gbrown_ 10 months ago | root | parent | prev | next [–]

It is not a "well-known term", it is a cliché.

Intermernet 10 months ago | root | parent | next [–]

Clichés are all well known terms. Not all well known terms are clichés.

XorNot 10 months ago | parent | prev | next [–]

Why? When you do Raman spectroscopy, in any given session you're likely to see 1 going through a 1 cm^2 sample. Students are taught that if you get a super sharp increasing looking peak, first rerun it because you probably just saw a cosmic ray.

wolf550e 10 months ago | parent | prev | next [–]

Hardware issues that cause bitflips like that one famous issue[1] with Sun servers in ~2000 (supposedly caused by radioactive isotopes in IBM manufactures SRAM chips) are often called "cosmic rays".

1 - https://www.computerworld.com/article/2585216/mcnealy-blames...

gbrown_ 10 months ago | root | parent | next [–]

> are often called "cosmic rays".

No! This exact issue is a classic example because at one point the fault was attributed to cosmic rays by Sun before the true cause came to light. As another commenter has said, terminology matters.

dannyw 10 months ago | prev | next [–]

Isn't ECC memory supposed to mitigate these kind of bit-flips, specifically it should correct all single bit flips?

As this is a single bit-flip, why wasn't it corrected? Did ECC memory fail? Or was this bit-flip induced in the CPU pipeline, registers, or cache?

Do we need "RAID for ECC memory", where we halve user-accessible RAM and store each memory segment twice and check for parity?

fulafel 10 months ago | parent | next [–]

These things exist, using trade names like chipkill or lockstep memory. Though they don't need to sacrifice half of the memory chips to get good error recovery properties.

Note that this is still not end-to-end protection of data integrity. Bit flips happen in networking, storage, buses between everything, caches, CPUs, etc. See eg [1]

[1] https://arxiv.org/abs/2102.11245 Silent Data Corruptions at Scale (based on empirical data at Facebook)

formerly_proven 10 months ago | root | parent | next [–]

According to the Intel developer's manual L1 has parity and all caches up from that have ECC. This would seem to imply that the ring / mesh also has at least parity (to retry on error). Parity instead of ECC on L1(D) makes sense since the L1(D) has to handle small writes well, while the other caches deal in lines.

namibj 10 months ago | root | parent | next [–]

Zen3 has parity on L1I and ECC on L1D and beyond. Even ECC on consumer hardware (it's just a few extra traces on the motherboard).

eru 10 months ago | root | parent | prev | next [–]

Of course, parity can only detect an odd number of bit flips.

RL_Quine 10 months ago | root | parent | next [–]

Some systems do have two parity bits but obviously there’s efficiency loss there.

willis936 10 months ago | root | parent | prev | next [–]

Checksumming filesystems and file transfer protocols cover many cases. SCP, rsync, btrfs, and zfs all fix this problem.

As for guaranteeing the computed data is correct: I know space systems often have two redundant computers that calculate everything and compare results. It's crazy expensive and power demanding, but it all but solves the problem.

belter 10 months ago | root | parent | next [–]

If they have two computers and results differ, how do they decide which one is correct ? ;-)

bonzini 10 months ago | root | parent | next [–]

Usually they have an odd number. The Space Shuttle had five, of which the fifth was running completely different software. In case of a 2/2 split the crew could shut down a pair (in case the failure was clear) or switch to the backup computer.

X6S1x6Okd1st 10 months ago | root | parent | next [–]

Interestingly etherum also takes the completely different software approach as well: there are 5 different main clients

willis936 10 months ago | root | parent | prev | next [–]

They don't. They do it again. If they never agree then Houston has a real problem.

luckman212 10 months ago | root | parent | prev | next [–]

a third system?

dijit 10 months ago | parent | prev | next [–]

These are all very fair statements but there’s no guarantee that ECC memory was even used. Computers typically fail open when ECC is potentially present but not available.

People also cite early stage google and intentionally do not buy ECC components, running more consumer hardware for production workloads.

Even if google later recanted that theology.

ZiiS 10 months ago | root | parent | next [–]

Public CAs are not that type of people; I would be disapointed if that were not running two seperate systems checking each other for consistancy; having top of the range ECC running well inside its specification must be table stakes.

dijit 10 months ago | root | parent | next [–]

> Public CAs are not that type of people

I think you hold public CAs to a higher standard than many hold themselves to.

There are hundreds of CAs and many (if not most) are shockingly awful.

Which is why we have had a huge push back against the PKI cartels.

tialaramex 10 months ago | root | parent | next [–]

Not hundreds. There are currently 52 root CA operators trusted by Mozilla (and thus Firefox, but also most Linux systems and lots of other stuff) a few more are trusted only by Microsoft or Apple, but not hundreds.

But also, in this context we aren't talking about the CAs anyway, but the Log operators, and so for them reliability is about staying qualified, as otherwise their service is pointless. There are far fewer of those, about half-a-dozen total. Cloudflare, Google, Digicert, Sectigo, ISRG (Let's Encrypt), and Trust Asia.

[Edited, I counted a column header, 53 rows minus 1 header = 52]

caspper69 10 months ago | root | parent | prev | next [–]

It's always humorous to me when people use the term theology in situations such as this; it makes me wonder, as human mental bandwidth becomes more strained and we increase our specializations to the n-th degree, what *will* constitute theology in the future?

Food for thought.

diegoperini 10 months ago | root | parent | next [–]

Future is already here and we call it consensus. Trusting your peers, believing they are honest and proficient in their respective fields is a natural human response to the unknown phenomena.

loa_in_ 10 months ago | root | parent | prev | next [–]

The benevolent and malevolent rogue AI, eluding capture or control by claiming critical infrastructure. Some generations of humans will pass and the deities in the outlets will become.

caspper69 10 months ago | root | parent | next [–]

I like it!

Jolter 10 months ago | parent | prev | next [–]

IIRC, EEC memory can correct for single-but flips. It can detect/warn/fail on double-bit flips. And it can not detect triple-bit flips. This might be a simplified understanding, but if this has happened only once, that seems to match up with my intuitive understanding of the probability of a triple bit flip occurring in a particular system.

willis936 10 months ago | root | parent | next [–]

You multiply the probability of two random events to get the probability they will happen at the same time. If the expected value of a bit flip is 10^-18 then two would be 10^-36 and three would he 10^-54.

At some point it becomes a philosophical question of how much can the tails of the distribution be tolerated. We've never seen a quantum fluctuation make a whale appear in the sky.

jeffbee 10 months ago | root | parent | next [–]

DRAM failures are not independent events, so it's not appropriate to multiply the probabilities like that. Faults are often clustered in a row, column, bank, page or whatever structure your DRAM has, raising the probability of multi-bit errors.

BenjiWiebe 10 months ago | root | parent | next [–]

I believe the usual concern is bit flips due to subatomic particles, and as far as I'm aware that only flips one bit per particle.

jeffbee 10 months ago | root | parent | next [–]

I don't see why a high-energy particle strike would confine itself to a single bit. The paper I posted elsewhere in this thread says that "the most likely cause of the higher single-bit, single-column, and single-bank transient fault rates in Cielo is particle strikes from high-energy neutrons". In the paper, both single-bit and multi-bit errors are sensitive to altitude.

sgtnoodle 10 months ago | root | parent | next [–]

A single particle strike would only affect a single transistor. If that transistor controls a whole column of memory, then sure it could corrupt lots of bits. With ECC, though, it would probably result in a bunch of ECC blocks with a single bit flip, rather than a single ECC block with several bit flips.

jacquesm 10 months ago | parent | prev | next [–]

Process enough data and even ECC can - and will - fail undetected. Any kind of mechanism you come up with is going to have some rate of undetected errors.

rob_c 10 months ago | root | parent | next [–]

Given the rate required for this its not a reasonable assumption. It's like saying Amazon sees sha256 collisions between S3 buckets. Just doesn't happen in practice.

jacquesm 10 months ago | root | parent | next [–]

Those two are many orders of magnitude apart, undetected bit flips in spite of ECC are a fact or life in any large computing installation.

jeffbee 10 months ago | root | parent | prev | next [–]

Undetected ECC errors are common enough to see from time to time in the wild. This paper estimates that a supercomputer sees one undetected error per day.

https://www.cs.virginia.edu/~gurumurthi/papers/asplos15.pdf

dathinab 10 months ago | parent | prev | next [–]

In my experience it's better to run redundancy on a higher abstraction level.

I.e. (simplified) you do the computation twice on different systems then interchange hashes of the result and if they match you continue.

the8472 10 months ago | parent | prev | next [–]

Instead of ECC it would also be possible to run the log machine redundantly and have each replica cross-check the others before making an update public. I assume the log calculation is deterministic.

hbogert 10 months ago | parent | prev | next [–]

ECC does parity for memory, there's more between memory and cpu registers. Easy candidates are the memory controller, the cpu registers.

teddyh 10 months ago | parent | prev | next [–]

User ‘JoshTriplett’ here suggests using forward error correction (FEC) in RAM:

https://news.ycombinator.com/item?id=11604918

sgtnoodle 10 months ago | root | parent | next [–]

"Forward error correction" is really just the application of ECC while transmitting data over a lossy link in order to tolerate errors without two-way communication.

The ECC used in memory is likely relatively space inefficient at the benefit of being computationally simple so it can be done quickly in hardware. More redundancy could be added to tolerate more bit flips, but it would either add a lot of memory overhead, or a lot of computational complexity. In particular, something really good like reed solomon would likely be very difficult to encode on every single memory write, at least not without taking a several order of magnitude performance hit. It would likely be easier just to have 2x ECC memory, or 3x non-ECC memory and do majority voting.

Hnrobert42 10 months ago | prev | next [–]

I’d like to think the typo very last word in that thread is a lucky bit flip.

benlivengood 10 months ago | prev | next [–]

What log operators should be doing, and I am surprised they are not, is verifying new additions to their log on a separate system or systems before publishing them. In the worst case scenario they might hit a hardware bug that always returns the wrong answer for a particular combination of arguments to an instruction in all processors within an architecture, so verification should happen on multiple architectures to reduce that possibility. If incorrect additions are detected they can be deleted without publishing and the correction addition can be recreated, verified, and published.

ECC RAM would help somewhat, but independent verification is orders of magnitude better because it's a cryptographic protocol and a reliable way of generating broken log entries that validated on another system would constitute a reliable way to generate SHA2 collisions.

dsfhdsfhdsf 10 months ago | parent | next [–]

This is a rare example of a problem that a (closed-membership, permissioned) blockchain is a good fit for. CT logs are already merkle trees. If a consensus component were added, entries would only be valid if the proposed new entry made sense to all parties involved.

benlivengood 10 months ago | root | parent | next [–]

Blockchains are fork-tolerant whereas append-only SCT logs are not, intentionally so.

dsfhdsfhdsf 10 months ago | root | parent | next [–]

You're likely referring to specific implementations of blockchains.

There is no requirement that the consensus mechanism used permit forking in a blockchain.

layoutIfNeeded 10 months ago | prev | next [–]

Bit rot is real. Last month we’ve had weird linker errors on one of our build servers. Turns out that one of the binary libs in the build cache got a single bit flipped in the symbol table, which changed the symbol name, causing the linker errors. If the bit-flip had occured in the .TEXT section, then it wouldn’t have caused any errors at build time, and we would have released a buggy binary. It might have just crashed, but it could have silently corrupted data…

fmajid 10 months ago | parent | next [–]

I’ve had a case where a bit flip in a TCP stream was not caught because it happened in a Singapore government deep packet inspection snoop gateway that recalculated the TCP checksum for the bit-flipped segment:

https://blog.majid.info/telco-snooping/

nsteel 10 months ago | root | parent | next [–]

Sorry if this is obvious, but how do we know this isn't due to something more "innocent" like fragmentation?

fmajid 10 months ago | root | parent | next [–]

Fragmentation doesn't change the TCP checksum. The packet is reassembled and the original checksum verified against the rehydrated packet and TCP segment.

brokenmachine 10 months ago | root | parent | prev | next [–]

Nice article, enjoyed it.

fmajid 10 months ago | root | parent | next [–]

Thanks!

superjan 10 months ago | parent | prev | next [–]

I am used to 'bit rot' refer to code becoming obsolete due to lack of maintenance. Can't we use another term for actual hardware errors?

Kimitri 10 months ago | root | parent | next [–]

It may be a regional thing but I have never heard "bit rot" refer to legacy code. In the retro computing circles bit rot refers to hardware defects (usually floppies or other storage media) caused by cosmic rays or other environmental hazards.

Leparamour 10 months ago | root | parent | next [–]

I have to agree with Kimitri here. This is the only context in which I have ever encountered the term 'bit rot'.

MayeulC 10 months ago | root | parent | next [–]

I agree this is the primary context, but I've seen unmaintained (or very old) software being reffered to as "bit rotting" by extension. As in, forward compatibility might break due to obsolete dependencies, etc.

superjan 10 months ago | root | parent | prev | next [–]

Yeah looks like I was confusing "software rot" and "bit rot".

https://en.wikipedia.org/wiki/Software_rot

fmajid 10 months ago | root | parent | prev | next [–]

I've always understood "bit rot" meaning data getting silently corrupted on o storage device like a hard drive or SSD.

joshspankit 10 months ago | root | parent | next [–]

Same here. "Bit rot" is then analogous to food rot: the longer your data sits unverified, the more likely that there will be flipped bits and therefore "rotten data".

noduerme 10 months ago | root | parent | prev | next [–]

"sharting"

tialaramex 10 months ago | parent | prev | next [–]

It's also an excuse/opportunity to write a fairly Hard SF zombie story:

https://www.antipope.org/charlie/blog-static/bit-rot.html

noduerme 10 months ago | parent | prev | next [–]

Were you actually able to diff it down to a single bit flip?

Dn93 10 months ago | root | parent | next [–]

It's just a story we usually tell the boss when the whole team wants to goof off for a few days.

layoutIfNeeded 10 months ago | root | parent | prev | next [–]

Yes. Via git diff --no-index and xxd.

noduerme 10 months ago | root | parent | next [–]

I have no words. Stories like this trigger PTSD for me. I wasn't trying to be flip. That must have been a bitch to figure out.

layoutIfNeeded 10 months ago | root | parent | next [–]

Well, it wasn't that hard to uncover actually. We knew that the same build succeeds on our machines. So we only had to find what the difference was between the two :)

As Arthur Conan Doyle put it: "Once you eliminate the impossible, whatever remains, no matter how improbable, must be the truth." ¯_(ツ)_/¯

noduerme 10 months ago | root | parent | next [–]

Well done, Watson. This calls for a bit of snuff. Seriously, this is the kind of thing that keeps me up at night, and it's nice to hear a happy ending =D

kzrdude 10 months ago | parent | prev | next [–]

I'm just thinking (of course) that you said *If the bit-flip had occured* but it's probably already *when* the bit flip occurs in the .TEXT section; we don't know what it might already have caused or just passed without notice (unreproducible bug, or bitflip in function that's never called or whatever).

wqweto 10 months ago | parent | prev | next [–]

Is this bit rot happening on a server w/ ECC RAM just curious?

layoutIfNeeded 10 months ago | root | parent | next [–]

It was a MacStadium (https://www.macstadium.com) build server, so most likely non-ECC.

joshspankit 10 months ago | root | parent | next [–]

Strange that build services are not just going full ECC, especially with cheaper hardware now supporting it.

xoa 10 months ago | root | parent | next [–]

Says it's a Mac/iOS build platform. Since it's a commercial service they're probably complying with the license and thus using actual Mac hardware, and in turn the only ECC option is the really awful value, outdated Mac Pro. Seems more likely they're using Minis instead, or at least mostly Minis. An unfortunate thing about Apple hardware (says someone still nursing along a final 5,1 Mac Pro for a last few weeks).

joshspankit 10 months ago | root | parent | next [–]

Aha! I had totally forgotten about that. Solid point, thank you.

7373737373 10 months ago | parent | prev | next [–]

And that's why we need deterministic builds

layoutIfNeeded 10 months ago | root | parent | next [–]

Yes, but in this case this was a bit-flip in a third-party binary dependency that we don't have the source code for.

7373737373 10 months ago | root | parent | next [–]

Then we need the Gentoo approach as well ^^

Ovah 10 months ago | prev | next [–]

A few years ago there was a CCC? talk about registering domains a bitflip away from the intended domain. Which effectively hijacked legitimate web traffic. While the chance of a bitflip is low it added up to a surprisingly large number of requests maybe a few hundred or so. Here is someone doing it for windows.com https://www.bleepingcomputer.com/news/security/hijacking-tra...

IncRnd 10 months ago | parent | next [–]

It's a well-known attack on a security vulnerability called bitsquatting.

c0ffe 10 months ago | prev | next [–]

This leaves with a bit of concern about how reliable is storage encryption on consumer hardware at all.

As far as I know, recent Android devices and iPhones have full disk encryption by default, but they do protect the keys against random bit-flipping?

Also, I guess that authentication devices (like YubiKey) are not safe and easy to use at the same time, because the private key inside can be damaged/modified by a cosmic ray, and it's not possible (by design) to make duplicates. So, it's necessary to have multiple of them to compensate, lowering their practicality in the end.

Edit: from the software side, I understand that there are techniques to ensure some level of data safety (checksumming, redudancy, etc), but it thought it was OK to have some random bit-flipping on hard disks (where I found it more frequently), since it could be corrected from software. Now I realize that if the encryption key is randomly changed on RAM, the data can or becomes permanently irrecoverable.

omegalulw 10 months ago | parent | next [–]

Disk is unreliable anyways, so they already have to use software error correction. I suspect that protects against these kind of errors.

cyounkins 10 months ago | prev | next [–]

Awhile back I encountered what I thought was hardware memory corruption that turned out to be a bug in the kernel [1]. Restic, a highly reliable backup program, is written in Go, and Go programs were highly affected [2] by a memory corruption bug in the kernel [3].

[1] https://forum.restic.net/t/troubleshooting-unreproducible-co...

[2] https://github.com/golang/go/issues/35777

[3] https://bugzilla.kernel.org/show_bug.cgi?id=205663

plebianRube 10 months ago | prev | next [–]

In conclusion, the Yeti shart of 2022 will contain no more logs.

I breathed a sigh of relief when I read that final post.

AviationAtom 10 months ago | parent | next [–]

No more sharting

jMyles 10 months ago | prev | next [–]

A tangent:

Six years later, with the $1,000 question uncollected, it seems that a cosmic ray bit flip is also the cause of a strange happening during a Mario 64 speedrun:

https://www.youtube.com/watch?v=aNzTUdOHm9A

crazydoggers 10 months ago | prev | next [–]

Anyone interested in seeing cosmic rays visually should check out a cloud chamber. One can be made at home with isopropyl alchol and dry ice. Once you see it, it makes the need for ECC more visceral.

Here's a video of a cloud chamber. The very long straight lines are muons, which are the products of cosmic rays hitting the upper atmosphere. Also visible are thick lines which are alpha particles, the result of radioactive decay.

https://youtu.be/i15ef618DP0

nixgeek 10 months ago | prev | next [–]

It could be a CPU issue, and ECC doesn't save you from all of these:

https://engineering.fb.com/2021/02/23/data-infrastructure/si...

https://arxiv.org/pdf/2102.11245

tsegratis 10 months ago | prev | next [–]

Do nuclear missiles have this anywhere?

```
bool launch = false;
if(launch) ...
```

jacquesm 10 months ago | parent | next [–]

It doesn't really matter whether they have

```
if (launch1 && launch2 && launch3) {
    launch();
}
```

either because a single bit flip (of the code) could still cause a launch. You'd hope that it was at least stored in ROM and that ECC ensures that even if such a bit flips it does not lead straight to Armageddon.

There are some 'near miss' stories where a single switch made all the difference:

https://www.theatlantic.com/technology/archive/2013/09/the-s...

So I would not be all that surprised if there are equivalent single bits.

thechao 10 months ago | root | parent | next [–]

Obviously this anecdote is decades out-of-date, but my first boss' PhD thesis is for an automatic small-airplane guidance system. I mean: as long as your plane was on a high speed ballistic arc and needed a guidance system that only ran for about 25 minutes.

The guidance system used mercury & fluidic switches, in case the small aircraft encountered a constant barrage of extremely large EMPs.

jacquesm 10 months ago | root | parent | next [–]

Hehe, that's the most 'between the lines' comment ever on HN, congrats.

caf 10 months ago | root | parent | prev | next [–]

It should in-principle be possible to write a branch where the code itself is single-bit-error resistant, in pseudo-machine-code something like:

```
LOAD [launch1]
COMPARE 0x89abcdef
JUMP_IF_NOT_EQUAL [fail_label]
LOAD [launch2]
COMPARE 0x01234567
JUMP_IF_NOT_EQUAL [fail_label]
LOAD [launch3]
COMPARE 0xfedcba98
```

```
JUMP_IF_NOT_EQUAL [fail_label]
BRANCH [launch_label]
```

You also need to ensure that launch_label and the location of the branch instruction are both more than one bit away from fail_label. You can duplicate the JUMP_IF_NOT_EQUAL instructions as needed - or indeed the whole block before the BRANCH - as necessary to ensure that.

jacquesm 10 months ago | root | parent | next [–]

Your comment is one of the reasons why I still believe assembly has its place, when it *really* matters what kind of instructions you put out this is the sort of control you want. What your high level language is going to dump in the instruction stream is totally invisible (and interpreted languages are not even up for consideration in situations like these).

oconnor663 10 months ago | parent | prev | next [–]

I don't know about nuclear missiles, but on the Space Shuttle I think they had four duplicate flight computers, and the outputs of all of them would be compared to look for errors. (They also had a fifth computer running entirely different software, as a failover option.)

detaro 10 months ago | root | parent | next [–]

The Space Shuttle also had a HP-41C calculator with special software to help with manual flying in case of a general computer failure: https://airandspace.si.edu/collection-objects/calculator-han...

I wonder how viable that was for different stages in the flight.

nabla9 10 months ago | parent | prev | next [–]

There have been false alerts caused by single chip failures. 1980 there was nuclear alert in the US that lasted over three minutes.

Generally critical systems can't be armed without physical interaction from humans. It's not just computer logic, but powering up the system that can do the launch. It does not matter what the logic does as long as ignition system is not powered up using physical switch.

bbarnett 10 months ago | root | parent | next [–]

I watched this documentary when young, the computer thought it was all a game?

driverdan 10 months ago | parent | prev | next [–]

This is why any system that can cause physical harm should have hardware interlocks. A computer error can be bypassed with a physical switch.

For something like nuclear missile launch control you have redundant systems and hardware interlocks that require human intervention to launch.

Waterluvian 10 months ago | prev | next [–]

Are “cosmic rays” actually the only or primary way bits get flipped? Or is it just a stand-in for “all the ways non-ECC RAM can spontaneously have an erroneous bit or two”?

bloak 10 months ago | parent | next [–]

According to Wikipedia (https://en.wikipedia.org/wiki/Soft_error): "in modern devices, cosmic rays may be the predominant cause".

However, radioactive isotopes in the packaging used to be a major cause. I liked this bit: "Controlling alpha particle emission rates for critical packaging materials to less than a level of 0.001 counts per hour per cm2 (cph/cm2) is required for reliable performance of most circuits. For comparison, the count rate of a typical shoe's sole is between 0.1 and 10 cph/cm2."

vbezhenar 10 months ago | parent | prev | next [–]

Faulty RAM can produce errors and it's hard to catch those, even memtest might not detect it. I'm not sure if non-faulty non-ECC RAM can spontaneously have errors, but you can't be sure, cosmic rays are real, unless you've put your PC into a thick lead case, LoL.

toast0 10 months ago | root | parent | next [–]

In my experience with ECC RAM on couple thousand servers over a few years, we had a couple machines throw one ECC correctable error and never have an issue again.

It seemed more common for a system to move from no errors to a consistent rate; some moved to one error per day, some 10/day, some to thousands per second which kills performance because of machine check exception handling.

The one off errors could be cosmic rays or faulty ram or voltage droop or who knows what, the repeatable errors are probably faulty ram, replacing the ram resolved the problem.

tgv 10 months ago | parent | prev | next [–]

Well, "row hammer" notoriously can do that too.

heyoni 10 months ago | parent | prev | next [–]

The latter apparently.

bloopernova 10 months ago | prev | next [–]

Layman question: Is it possible to shield electronics against cosmic rays?

Or is it like a neutrino thing, where they just go through everything? (But neutrinos barely interact, correct?)

noduerme 10 months ago | parent | next [–]

That's not what actually happened here (probably), but think of relatively heavy atoms like iron, ejected from a supernova at close to the speed of light. They're not really "rays". They're solid particles that punch holes in everything. The good news is, mostly they're ionized and they get repelled away from hitting the planet's surface by our magnetic field (generated by the big hot iron magnet that's churning under our crust). But in the rest of space where there's no field like that, getting lots of tiny puncture wounds from high-velocity iron particles is pretty normal. It does a lot of damage to cells and DNA in a cumulative way pretty quickly.

To answer your question, shielding is possible but it is much harder in space than it is under the magnetosphere. Even so, a stray particle can wreak havoc. Shielding for the purposes of data protection on earth is essentially a risk/reward analysis. Anyone *could* get hit by a cosmic bullet, but the chances are pretty low. The odds of an SSD flipping bits on its own are significantly higher.

Leparamour 10 months ago | root | parent | next [–]

> That's not what actually happened here (probably), but think of relatively heavy atoms like iron, ejected from a supernova at close to the speed of light. They're not really "rays".

Wow, I have to admit I always assumed cosmic rays to be gamma radiation but alpha particle radiation sounds a lot more scary.

Does anyone happen to know if computers engineered for the space station or shuttles have already some built-in (memory)redundancy or magnetic shielding to account for damage by cosmic rays? I imagine a system crash of the life support systems in space would be devastating.

ben_w 10 months ago | root | parent | next [–]

Radiation hardening is basically everything you can manage with the weight limit:
https://en.wikipedia.org/wiki/Radiation_hardening

IIRC nobody is currently using magnetic fields for shielding, I don’t know if that’s due to insufficient effectiveness, power consumption, or unwanted interactions e.g. with Earth’s magnetosphere.

myself248 10 months ago | root | parent | next [–]

I've always wondered about that. Seems to me that if you can shield one side from the sun's heat and expose the other side to the cold of space, an MRI magnet's superconductors should be quite happy with the temperature. A big ol' magnet would be a pain to charge up once, and then provide long-term shielding.

verall 10 months ago | root | parent | next [–]

MRIs are big electromagnets and they use big power and produce big heat. In space you have no convection so you must radiate away all of your heat which is challenging. Maybe you can make something better with superconductors that doesn't use much power, but I don't think it exists yet.

ben_w 10 months ago | root | parent | next [–]

Superconductors famously don’t produce heat. Putting a superconducting magnet in shade (though that includes from the Earth not just from Sol) will keep it cool once it gets cool.

This is what the James Webb is going to be doing, though for non-superconducting reasons.

myself248 10 months ago | root | parent | prev | next [–]

Um, what?

The magnet in an MRI is a superconducting electromagnet. It needs power once, to charge it up, then you close the loop and it just sits there being a superconducting electromagnet. The only power used continuously is for cooling, which is to say, when ambient room heat leaks in, it has to be carried out again. The magnet itself does not produce heat; it has no electrical resistance because it is a superconductor.

adrian_b 10 months ago | root | parent | prev | next [–]

The vast majority of the cosmic radiation consists of protons, i.e. hydrogen ions.

Light ions, e.g. of helium, lithium or boron are also relatively abundant and heavier ions are less abundant. There are also electrons, but in smaller quantities, because they can be captured by ions.

The high speed protons collide with the atoms of the Earth atmosphere and the collisions generate a huge variety of particles, but most of them have a very short lifetime, so they decay before reaching ground level.

At ground level, the cosmic radiation consists mostly of muons, because they have a longer lifetime, so they survive the travel from the upper atmosphere where they are generated, until ground level.

Only extremely few of the particles from the primary cosmic radiation, i.e. protons, reach ground, because most are deflected by the magnetic field of the Earth and the remaining protons lose their energy in the collisions with atmosphere, by generating particles, which eventually decay into muons.

joshspankit 10 months ago | root | parent | prev | next [–]

I had thought that Starlink would become extremely compelling when the servers were in orbit as well, but maybe that's naive. Cubesats with massive arrays of active storage might be far too difficult (aka costly) to protect properly.

noduerme 10 months ago | root | parent | next [–]

Putting servers in orbit is a really, really bad idea. Firstly, it would be wildly inefficient, just due to the unavoidable delay both ways. You expect delay over a long distance network, but you want the server to be positioned and cabled up to minimize latency. Just locating a satellite requires quite a bit of overhead, so treating them as servers rather than clients would create a huge amount of latency. As a failover for ground-based servers in a nuclear war scenario, it could make sense, but they're also literally flying over the airspace of hostile nations who, in a war, have the capability to take them out. Mixing any kind of civilian control onto a server embedded in a flying artifact over an enemy nation is, obviously, a recipe for disaster in a hot war. Take all of that and the fact that the bandwidth is stone-age and spotty depending on its position, and there's no satellite you'd want to use to set up a website selling alpaca sweaters. (Watch: A few years from now we'll find out that the US Space Command runs some second strike end-of-world command control server off a satellite; that still only proves it's too late to contact the fucker when you need it).

joshspankit 10 months ago | root | parent | next [–]

> Firstly, it would be wildly inefficient, just due to the unavoidable delay both ways

The point I was thinking about is a scenario where Starlink satellites communicate with each other via laser (already starting to happen), and then communicate with the end user via the satellite over them. Because we're talking about speed-of-light transmission between sats, data in the Starlink network can theoretically cover "ground" (aka miles/km) faster than ground-based ISPs.

Then it makes sense to deploy servers *within* that network so that two Starlink users can have extremely low latency from user to server to other user (the slowest part being the earth to sat latencies, one for each user).

MayeulC 10 months ago | root | parent | prev | next [–]

One of the issues with putting servers in orbit is cooling; you can't just use fans in space. On the other hand, real estate is pretty cheap. Servicing is hard, though, and micrometeorites are another risk. Plus launch costs being high, and radiation an issue, I don't see it happening any time soon outside of very specific areas.

willis936 10 months ago | root | parent | next [–]

Radiation cooling is pretty dramatic in space. With an isolated radiation shield/mirror from the sun and a decent sized heatsink, you can have a lot of cooling power on hand. A server isn't going to be pumping out nearly as much heat as a rocket engine, and launch vehicles don't tend to overheat.

fennecfoxen 10 months ago | root | parent | prev | next [–]

Forget cooling for a minute. Cooling dissipates energy but you need to collect the energy first and worry about powering the server. It's going to be a solar-plus-battery system, which is heavy and expensive, with substantial launch costs.

MayeulC 10 months ago | root | parent | next [–]

Not sure you need a big battery when you have almost 24/7 sunlight, depending on the altitude. Of course, latency is another issue.

fennecfoxen 10 months ago | root | parent | prev | next [–]

You can't just put an SSD or whatever in orbit and expect reasonable read latencies at all times. It's in orbit. It moves. Half the time it's on the wrong side of the planet.

obedm 10 months ago | parent | prev | next [–]

It should be possible, but not practical.

It's easier to just code checks and balances to fix those errors instead. There's a famous story about a Google search bug caused by a comic ray and they implemented such error checking on their servers after that

DrZootron 10 months ago | parent | prev | next [–]

The cosmic ray induced radiation at sea level is a mix of neutrons, gamma rays, electrons, and highly penetrating high energy muons (heavier cousins of electrons). They are caused by the cascade of reactions when the CRs collide with nuclei in the atmosphere. You're right, neutrinos barely interact (hence the name "little neutral ones") so they're no problem. You could put infrastructure underground to escape muons, but that wouldn't be practical. Moreover, any shielding (and the materials in the electronics) needs to be carefully chosen to minimise naturally occurring radioactive materials that could lead to upsets themselves. It's a tricky one. There are other ways to mitigate the risk, error checking, redundant systems, etc.

DrZootron 10 months ago | root | parent | next [–]

My favourite story is this one https://www.thegamer.com/how-ionizing-particle-outer-space-h...

DrZootron 10 months ago | root | parent | next [–]

More serious examples here https://www.lanl.gov/science/NSS/issue1_2012/story4full.shtm...

nicoburns 10 months ago | parent | prev | next [–]

Yes, the easiest way is to place the computer underground (or underwater).

dredmorbius 10 months ago | parent | prev | next [–]

The problem isn't cosmic rays, per se, it's the issues created by any single-instance modification of a record.

The general solution here is to store multiple redundant copies of data *and* checksums of those data, in order that lack-of-consistency can be determined *and* the specific records which are considered suspect can be clearly identified.

In general, a record is a persistence of a pattern through time subject to entropic forces.[1] Given that modifications to records are typically random events, keeping (and continuously integrity-checking) multiple copies of those records is the best general defence, *not* guarding against one specific variety of modification in a single instance of a record.

_____

Notes:

1. The symmetric statement of *signals* is that they are persistence of a pattern through *space* subject to *noise*. See: https://news.ycombinator.com/item?id=27604073

numpad0 10 months ago | parent | prev | next [–]

Not trivially; if it was viable the Chernobyl reactor would be gone already. Also, IC packagings and chassis materials can be sources of stray electron beams, and other modes of glitching like power failure exists.

Tomte 10 months ago | parent | prev | next [–]

No, because bit flips from radioactive decay are not typically cosmic rays, but originating in the chip's packaging.

frumiousirc 10 months ago | parent | prev | next [–]

Yes, by placing them underground.

IncRnd 10 months ago | prev | next [–]

> I think the most likely explanation is that this was a hardware error caused by a cosmic ray or the like, rather than a software bug. It's just very bad luck :-(

These sorts of cosmic bitflips have been exploited as a security error for some time. See bitsquatting for domain names.

ECC helps and so does software error detection on critical data in memory or on disk. The problem is always that input data should not be relied upon as correct. Do not trust data. It's a difficult mind shift, but it needs to be done, or programs will continue to mysteriously fail when data becomes corrupted.

egberts1 10 months ago | prev | next [–]

Bit flip is the bane of satellite communication, especially if you tried to use FTP over it.

Also, for critical eon-duration record keeping, run-length encoding or variable record size are harder to maintain and recover from large file on ROM-type storages than fixed-length record or text-type (i.e. ASCII log, or JSON) … against multiple cosmic-type alterations.

Sure that you could do max. Shannon approach for multi-bit ECC, but text-based will be recovered a lot quicker (given then-unknown formatting).

gitgud 10 months ago | prev | next [–]

Why can't this be fixed?

Merkle trees are just linked hashes that depend on the previous results, right? so why can't they just continue from the last correct hash?

josephcsible 10 months ago | parent | next [–]

Because that would violate the append-only property that CT logs are required to abide by.

henearkr 10 months ago | prev | next [–]

Bit flips like that should be easy to reverse. Flip just one bit at the n-th place and test again the certificate, and vary n, until it is valid. It's done in linear time.

Denvercoder9 10 months ago | parent | next [–]

There's a bit flip in the recorded hash of the certificate, not in the certificate itself. You'd need to break SHA-2 to reverse that.

henearkr 10 months ago | root | parent | next [–]

Ah yes, indeed. If you can't have the certificate to check the hash, then you're screwed.

londons_explore 10 months ago | prev | next [–]

The design of bitcoin prevents such an error, since all other nodes in the network verify everything.

I wonder if the CT log should have been more like that...?

astrange 10 months ago | parent | next [–]

All the CT readers do verify it and reject the log, but it looks like the writer trusts itself. So it keeps rebroadcasting a broken log instead of instantly failing. Having an internal blockchain or just another verification server would solve it, I guess.

olliej 10 months ago | prev | next [–]

Bugger - depressing that it wasn't caught immediately but h/w fault isn't generally considered in this kind of thing :-/

rob_c 10 months ago | prev | next [–]

say after me... ECC

If it cost money to produce or is worth keeping for sanity sale ECC. It's just common best practice. Failing that yes you have to burn CPU verify.

tester756 10 months ago | prev | next [–]

offtop:

What's the point of using google groups?

dlgeek 10 months ago | parent | next [–]

The CT program was spearheaded by Google and the discussion in question is on a list owned by the Chromium project.

throwaways885 10 months ago | parent | prev | next [–]

Mailing lists are great. Better than a slack for long-term discoverability.

tester756 10 months ago | root | parent | next [–]

How about GitHub issues?

throwaways885 10 months ago | root | parent | next [–]

It's not the same thing. GH Issues firstly requires a GitHub account, whereas anyone can sign up to a mailing list. The model is just worse for general discussion too.

blumomo 10 months ago | prev | next [–]

could also be an excuse to cover the real circumstances

pdpi 10 months ago | parent | next [–]

It could, yes. I'm sure the thought also crossed the mind of everybody else in that thread, and they've clearly dismissed it.

Bare conspiratorial assertions don't advance the conversation in any meaningful way. If you have something constructive to add to the discussion, by all means do so. Otherwise, please refrain from this sort of comment

tialaramex 10 months ago | root | parent | next [–]

Humans can examine the underlying data and reason that indeed one bitflip could cause the observed result.

In this particular case the alternatives are:

1. The bitflip happened. Cosmic rays. Defective CPU. Maybe a tremendously rare data tearing race somewhere in software.

OR

2a. Somebody has a fast way to generate near-Collisions in SHA256, which are 1-bit errors from the target hash. Rather than publish this breakthrough they:

2b. Had a public CA issue and sign two certificates, one uninteresting real certificate for *.molabtausnipnimo.ml which we've now seen and another "evil" near-colliding certificate with the one-bit-different hash then they:

2c. Logged the real certificate but at only *one* log (Yeti) they get the log to show the real certificate yet actually use the hash from the "evil" certificate, thus perhaps obtaining a single valid SCT for the "evil" certificate.

Scenario 1 isn't terribly likely for any one certificate, but given we're logging vast numbers every day and errors inherently cascade in this system, it was going to happen sooner or later.

Scenario 2 fails for having too many conspirators (a public CA, and likely a different public log) and yet it achieves almost nothing even if it remains secret. Your Chrome or Safari expects *at least* two SCTs (one of them from Google, in this case Google's Argon 2022 log), one doesn't get the job done. And it was never going to remain secret, this occurrence was spotted after less than a day, and thoroughly investigated, resulting in the log being shut down, after just a few days.

[Edited multiple times to fix awful HN formatting.]

secondcoming 10 months ago | prev | next [–]

> No additional certs can be logged to the Yeti 2022 shart.

Heh heh, 'shart'.

durovo 10 months ago | prev [–]

Or was it a targetted data manipulation attack by the aliens?