

Forum:[Hack Forums](#)

Thread: [New Phoenix Variant Defeats DDR5 TRR Root Access Possible on Real Hardware](#)

JoseMar - September 16, 2025, 5:03 am

New Phoenix Variant Defeats DDR5 TRR
Root Access Possible on Real Hardware

Saw the Phoenix paper and I am equal parts impressed and annoyed. Felt like dumping my half-baked thoughts here before I forget them or someone else steals them. This will be messy. Good.

OP: Phoenix Rowhammer??

Okay, quick recap for anyone who skim-reads and then asks the same basic question ten minutes later. Phoenix is a Rowhammer variant that apparently works on DDR5. Yes, DDR5, the “new and safe” stuff. Researchers from ETH Zurich and Google found a way to bypass the in-DRAM TRR mitigations, using something they call self-correcting synchronization. The attack can keep in sync with refresh cycles long enough to cause bit flips and, in a test, they used those flips to get root in around 109 seconds. CVE-2025-6202 is the tag. Cool and terrifying depending on your day job.

Why I care: Rowhammer used to feel like old news, DDR3, DDR4, weird lab setups. DDR5 was supposed to be harder. Turns out “supposed to” and “actually is” do not always match.

So why does this suck

Short version: those TRR things in DDR5 are more complicated, but they are not bulletproof. Phoenix basically admits when its timing drifts and corrects itself. It is not some lightning-fast dodge trick where you never miss a beat. It is more honest: “I missed a refresh, ok realign, continue.” That sounds... grossly effective.

Also: they tested 15 SK Hynix modules from 2021 to 2024 and got flips on all of them. That is not a one-off fluke. That is a pattern. If you rely on vendor secrecy or undocumented behavior for security, this is a reminder that secrecy is brittle.

And yes, pumping refresh rate up by 3x works, but that costs 8.4 percent performance in their tests. For data centers or latency-sensitive workloads that is not nothing. So we are left at a trade-off table: security vs throughput. Different shops will choose differently and that uncertainty is a vector too.

Proof / demo bits (what actually happened)

They reverse-engineered TRR timing enough to find “blind spots.” That phrase alone makes me imagine memory doing its housekeeping while someone pokes holes in the curtains. The trick is not raw speed. It is persistence plus the ability to detect when the cadence slips. So Phoenix doesn't necessarily need exotic hardware or modified firmwares. It runs on commodity machines with default settings. That is the part that should keep you awake if you run any shared infrastructure.

The researchers reportedly achieved root in as little as 109 seconds on a test system. Which, again, is a lab demo. It is not the same as seeing this as a worm in the wild, but the step from local exploit to remote or cloud escalation is not as big as some would like to believe. Especially if an attacker can land code on a VM in a multi-tenant environment.

They disclosed to SK Hynix, CPU vendors, and major cloud providers in June 2025. Responsible disclosure happened, so credit where due. But what now? Vendors will patch firmware, maybe change controllers, maybe push microcode. Or they will say “increase refresh rates and be done.” Which is a blunt instrument.

Personal brain-dump / implications

My gut says a few things. One, relying on opaque, proprietary mitigations is a risky long-term bet. Security-by-obscurity dies slowly and then the obituary is written by a grad student with enough patience. Two, vendors need auditable, verifiable defenses. If a black box does something complicated, someone has to be able to say definitively “this class of access patterns cannot flip bits.” Otherwise the story repeats.

Three, cloud operators and folks who do multi-tenant hosting should treat this as a reminder to harden attack surfaces that seem “too low level to matter.” Hardware bugs are slow poison: they look like corner cases until someone turns them into privilege escalation.

Also, forensics will be messy. Bit flips are ephemeral and weird. If you are the incident responder and the flip changes a page table bit? That leaves a trail, yes, but not the usual one. This complicates attribution.

Would I panic? No. But I would plan. Audit: what DDR5 modules do you run? Who is the vendor? Can you apply microcode/firmware updates if offered? Can you evaluate the performance hit of increased refreshes and whether a temporary increase is worth it while patches roll out?

Questions I want answered (and you might too)

Can other DDR5 vendors be hit the same way or is there something unique to the tested SK Hynix modules? The paper suggests the TRR interplay is complex and vendor-specific, but my suspicion is that other implementations could have similar weaknesses.

Is there a cheap in-software mitigation short of tripling refresh? Maybe OS-level monitoring that looks for suspicious, repetitive memory access patterns? Maybe, but attackers can be subtle. Also, the performance cost might be worse if implemented in software.

Could this be weaponized for remote attacks? Currently more likely as a local escalation, but chaining is always a thing. If you can get code execution somewhere, this lowers the bar to full system compromise.

Phoenix is clever and annoying

It reminds us that hardware is still part of the attack surface and that “newer” does not mean “safe.” The defense story will be interesting to watch: short-term mitigations will appear, vendors will issue firmware updates, cloud providers will tweak configs, and somewhere in that churn the attackers will look for the next invisible seam.

If you run DDR5 in anything critical, poke your vendors and your ops team. Ask them if they were notified, and what the mitigation plan is. If you are curious or paranoid like me, read the research when you can. It is not bedtime reading, but it will change how you think about memory for a while.

COMSEC Channel dropped a YouTube vid with proof

Phoenix going all the way to root

[Video: https://www.youtube.com/watch?v=1emxVQ6__qg]