— FLAGSHIP RELEASE

Claude Mythos found hundreds of Firefox bugs. The real story is the triage.

By **Flowi Editorial** · *May 9, 2026 · 6 min read*

Mozilla used Claude Mythos to find vulnerabilities in Firefox. The interesting shift is what happened to the security team's workflow once the bug volume changed.

A security team that used to find one critical vulnerability a week now finds twelve in a morning. That's the shift Mozilla just published — quietly, in a hacks.mozilla.org post — about what happened when they pointed Anthropic's Claude Mythos preview at the Firefox source tree.

The headline number ("hundreds of vulnerabilities found and fixed") is the wrong thing to focus on. The interesting story is what changed in **how Mozilla's security workflow runs** once the cost of finding bugs dropped to roughly zero. Because that's the part that translates to your codebase, whether or not you ever get a Mythos preview key.

What launched

Anthropic gave Mozilla preview access to Claude Mythos — the version of Claude that's been spending an unusually long time on a single research task before reporting back. Mozilla pointed it at the Firefox C++ source, the Rust modules, the JavaScript engine, and the WebAssembly runtime. They asked it to find memory-safety bugs, race conditions, and integer overflow paths.

What came back was, in Simon Willison's framing, "suddenly the bugs are very good." Mythos didn't just produce a wall of false-positive grep hits. It produced reports that included a hypothesized exploit path, a minimal reproducer, and a suggested patch. Three of those things are usually the bottleneck on a security ticket.

Mozilla published a blog post. Buried in it: the team had to invent a new triage tier because their existing severity ladder was designed for a world where vulnerabilities arrived one at a time, on a Friday afternoon, from a lone external researcher. Mythos delivered them in batches of fifteen, on a Tuesday.

Why it matters

The bottleneck in security work was never finding bugs. Static analysis tools have been finding bugs for thirty years. The bottleneck was deciding **which of the 8,000 reported issues are real and worth fixing** before the engineers stop returning your DMs.

Three things have to be true for a bug report to actually get fixed:

1. The bug is real (most aren't — false positives have always been the dominant failure mode of automated security tooling).
2. Someone can reproduce it cheaply (a one-line "potential nullptr deref in handler.cpp:412" is not actionable).
3. Someone can patch it without breaking three other things.

Mythos is closing all three at once. The reports include the exploit path, so the bug is provably real. They include a minimal reproducer, so engineering can re-run it locally and watch it fail. They include a candidate patch, so the engineering decision becomes "merge this" rather than "investigate."

When you collapse those three steps from "two days of senior security engineer time" into "fifteen minutes of code review," the math of how a security team is staffed changes. Mozilla's blog post mentions this in passing — "we've moved security review earlier in the cycle" — but the operational implications are bigger than that line suggests.

The triage shift nobody is talking about

Here's what you don't see in the Mozilla post but probably should: the security team's job has structurally changed. They're no longer the people who **find** vulnerabilities. They're the people who **rank** vulnerabilities — what fixes ship this release, what fixes can wait, what's a real attack path versus a theoretical one.

That's a different skill profile. The senior security engineer of 2024 was, broadly, an exploit researcher who could think like an attacker. The senior security engineer of 2026 needs to be a **threat-model strategist** who can read fifty Mythos reports a day and decide which three matter most for the threat model of the next release.

This isn't a hypothetical. It's what Mozilla had to actually build a process around in real time, on a preview product, in the last three months. The blog post describes a new internal tier they call "Mythos-class" — a queue of bugs that are real, are exploitable, but for which the priority is set by **product impact** rather than discovery date.

That's the shift you should be paying attention to: the security team has been promoted to threat-model owners. The bug-finding work is now a commodity input.

What this means for your codebase

You probably don't have Mythos preview access. That's fine. The lessons from Mozilla's experience apply anyway, because the trajectory is one-way: AI vulnerability scanners are about to become a baseline expectation in any serious security program. Anthropic, OpenAI, and the open-weight models are all converging on the same capability. The question is when, not if, you can run something Mythos-class against your own code.

When that day arrives, three things will hurt your team if you haven't prepared:

The triage tier doesn't exist. Most teams' bug trackers have "P0 / P1 / P2" tiers that were calibrated for a world where one P0 came in per week. When fifty come in per morning, your tier definitions need to be product-impact-aware, not severity-aware.

Reproducer environments aren't reproducible. A security report with a reproducer is only useful if the reproducer runs. Most internal tooling assumes a developer's local machine, with a particular Docker compose file, with particular env vars. The AI scanner doesn't have that context. Your reproducer infrastructure needs to be self-contained and one-command-to-run, or the

velocity advantage evaporates the moment a developer has to spend forty minutes setting up their environment.

The "merge this patch" muscle is weak. Most security fixes today are written by the engineer who owns the file, often two weeks after the original report. When the AI generates a candidate patch, your code review process needs to be set up to **evaluate the patch at the source-language level**, not "tell the engineer to investigate." That requires reviewers who can read security-class code changes critically — which most engineering teams have outsourced, increasingly, to "the security team."

Who should care

- **Security engineers shipping in production:** start writing your team's playbook for when bug volume goes 10x. The shift from finder to ranker is going to land within twelve months.
- **CTOs and engineering managers:** if your security team is staffed for "find bugs," restaff. Find-bugs is being commoditized. Rank-bugs is the work.
- **Solo and small-team builders:** AI security review is going to become table-stakes for any code you ship to production. Start running Claude Code on your repo with security-focused prompts now, so the muscle is built before the volume arrives.

The Mozilla post will be remembered as the first public deployment write-up of an AI security tool that genuinely changed how a major engineering org operates. The hundreds-of-bugs-fixed number is impressive. The reorganization of the team that fixed them is more so.

If you're building AI-powered systems in production — agents, automations, security tooling — the patterns Mozilla just landed on are the same patterns you'll need. The decision tree for how an agent surfaces work, prioritizes it, and hands it off cleanly to a human reviewer is the bottleneck across every production AI system right now. That's why we wrote Agent Memory: The 5 Patterns That Ship in Production — the failure modes aren't unique to security.

Originally published on useflowi.app/blog/claude-mythos-found-firefox-bugs-the-real-story-is-the-triage.
Flowi — the editorial intelligence layer for AI builders.
Daily brief at useflowi.app/blog · Monthly Dispatch at useflowi.app/dispatch.