



eBook

# The Security Lakehouse: Outpacing AI Zero-Days



# Contents

Introduction ..... 3

The Capability Gap ..... 7

From Patch Cycle to Exploit Window ..... 9

Widespread Threats.....11

The Security Lakehouse.....13

AI Agents for Defense .....15

Summary .....17



## Introduction

Vulnerability management used to run on a comfortable clock. A vulnerability was disclosed, the team triaged it, performed some testing and scheduled the patch for the next recurring maintenance window. However, vulnerabilities aren't always in a third-party product that can be patched this way. Just as often, the vulnerability is in the organization's own application code, its infrastructure configuration or the software it shipped to customers, and it frequently arrived via a popular open source library woven across all three. There, the fix was not just a patch to install but a hunt to run, finding every place the vulnerable code or dependency was used, upgrading or rewriting it, then rebuilding and redeploying. This all took time, and accepting the risk associated with that gap has become routine.

This approach has (mostly) worked because we could assume attackers moved at roughly human speed. Someone had to read the advisory, understand the bug, write a working exploit and successfully target exposed assets before real harm could be inflicted.

That assumption no longer holds. Attackers now easily point powerful AI models and agent harnesses at fresh advisories and produce working exploits in hours, sometimes faster. Anthropic's **Project Glasswing** made this concrete. Their **Claude Mythos Preview** model finds and exploits software vulnerabilities on its own, and public reporting credits the model with surfacing flaws that had gone unreported for years in widely used open source software, including a **27-year-old bug in OpenBSD and a 16-year-old bug in FFmpeg that automated testing had missed**.

The **UK government's AI Security Institute** confirmed the leap independently, rating Mythos Preview the first model to complete its multistep simulated corporate intrusion fully and to clear 73% of expert-level capture-the-flag challenges. OpenAI followed with **Daybreak**, a comparable initiative built on their Codex Security agent.

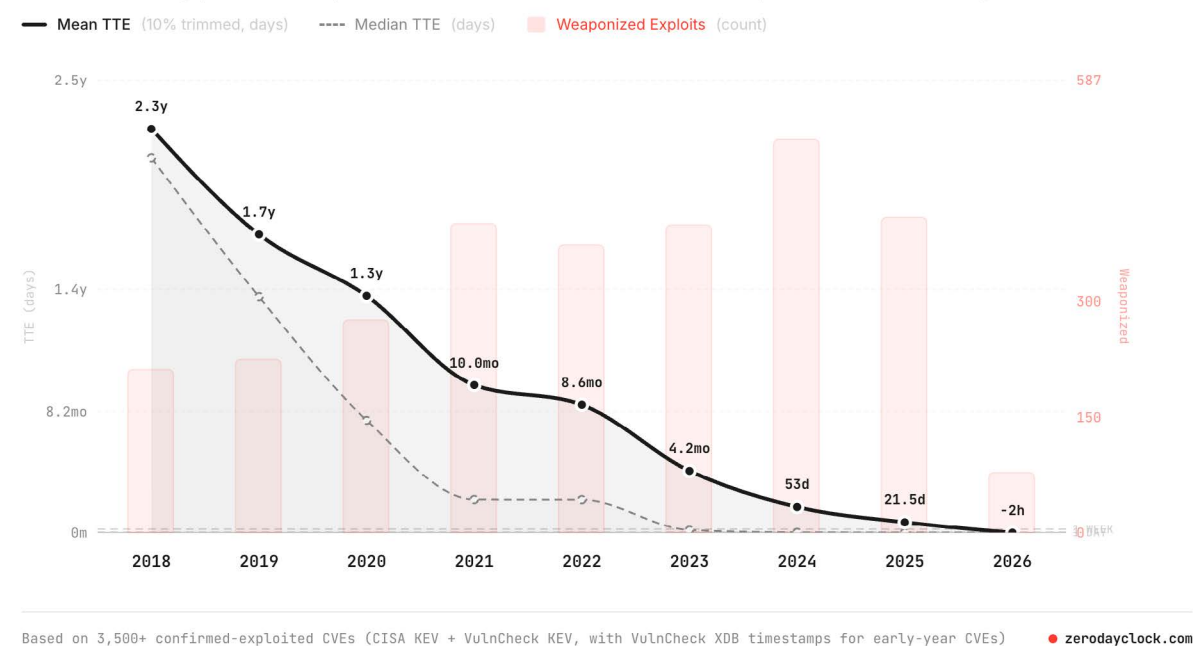
The partners in these programs are already running the models against their own code to identify weaknesses which, in turn, could affect millions of systems and users around the world. **Cloudflare**, for one, pointed Mythos at their repositories behind a harness of roughly 50 parallel hunter agents.

And this is no longer confined to research and defensive programs. **Google's threat intelligence group** has documented the first confirmed AI-generated zero-day exploit, attributed to threat actors using a frontier model to find and weaponize a logic-flaw authentication bypass that fuzzers and static analyzers can't catch.

Mythos and Project Glasswing hit the mainstream news cycle, but the underlying shift is older and broader than any single model, lab or program. The time between disclosure and weaponization has been shrinking for years. The **Zero Day Clock**, an industry coalition that tracks the median time from disclosure to first in-the-wild exploitation, marks a fall from 771 days to single digits over the past five years, with the curve still trending toward zero.

### From Vulnerability to Exploitation

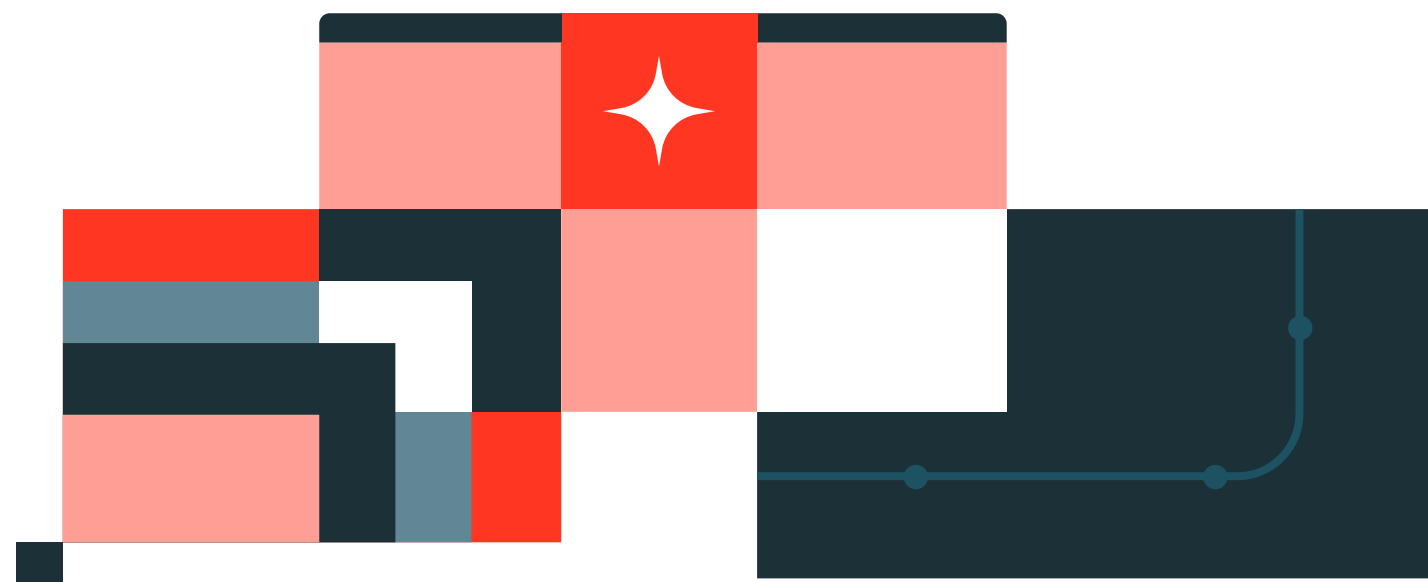
TTE measures the gap between CVE public disclosure and first confirmed in-the-wild exploitation. Zero = same-day.



Here is the uncomfortable part: Most security programs still measure themselves by the patch cycle — mean time to patch, percentage of assets patched within the SLA and backlog burn-down. Those metrics describe how fast a team closes the door, but they say nothing about how long the door actually stood open, which runs until the fix is installed on every exposed asset, not the moment the vendor ships the patch.

Worse, programs are losing ground by their own measure. Verizon's 2026 **Data Breach Investigations Report** found that only 26% of the vulnerabilities on CISA's Known Exploited list were fully remediated in 2025, down from 38% the year before, while the median time to patch slipped from 32 to 43 days, against roughly eight times the vulnerability volume of 2022. The same report puts vulnerability exploitation as the top initial-access vector at 31%, up from 20%.

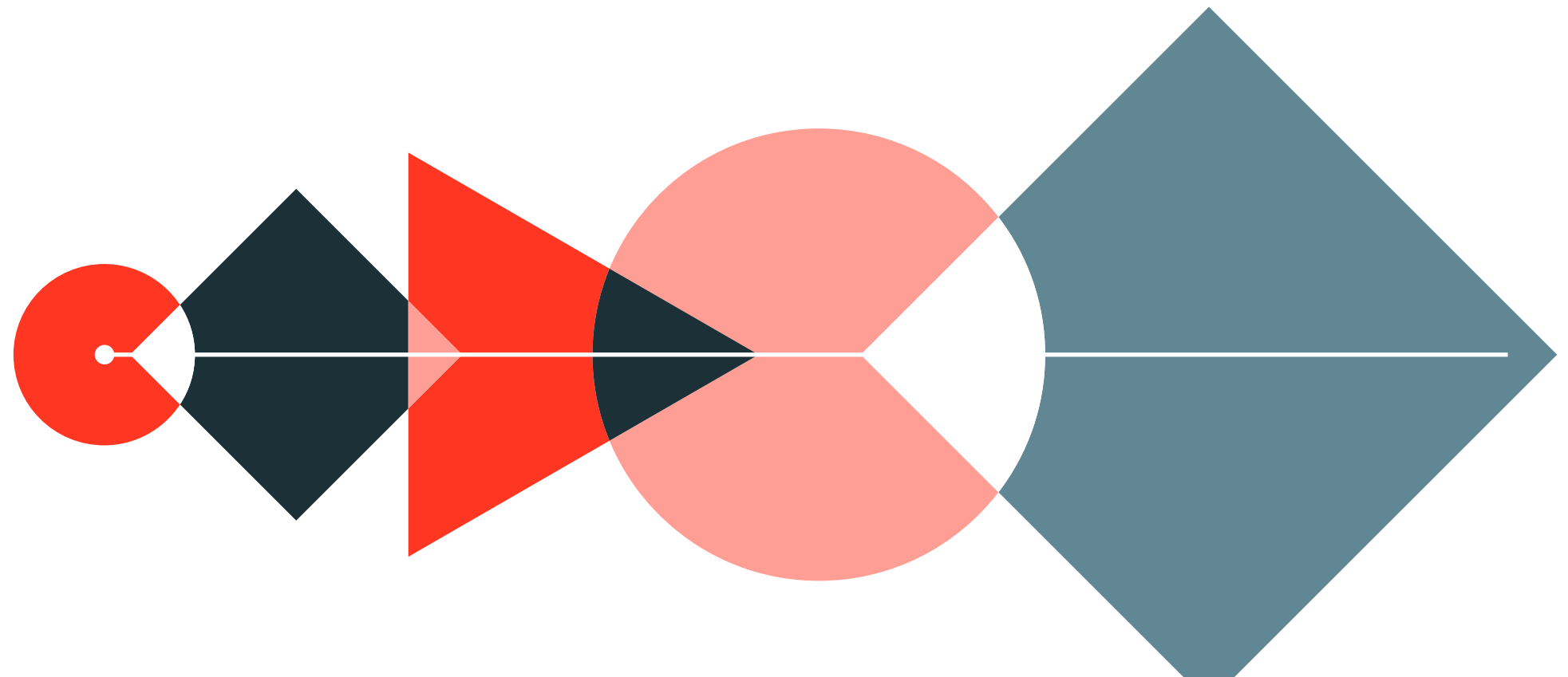
This guide argues for a different unit of measure, the exploit window, and an operating model to compress it using new data and AI approaches. That work tackles the vulnerability crisis facing security teams today, but the bigger picture matters too. The vulnerability crisis will almost certainly be just the first AI-driven security crisis of this scale, not the last. The AI-driven challenges that follow may be harder, and they may not arrive one at a time. The durable question for a CISO is how to meet this crisis while building a data and AI foundation flexible enough to pivot to whatever comes next.



The foundation is a platform, not a point tool, and it should serve the whole security organization. Security organizations comprise many different teams: security operations/SOC, incident response and forensics, vulnerability management, threat intelligence, offensive security, application and product security, identity and access management, insider threat, customer trust, and governance, risk and compliance (GRC). These functions all run on data and AI. Successful security teams are choosing data and AI platforms that are flexible and powerful enough to meet the needs of all of them.

This guide is about building that platform and putting it to work, with the vulnerability crisis as the proving ground. It shows security teams how to:

- Reframe vulnerability management in terms of the shrinking exploit window
- Consolidate security telemetry onto one open, governed platform
- Put defensive AI agents to work continuously
- Govern agents and the data they can access



## The Capability Gap

The challenge is clear. Attackers have AI that finds and weaponizes vulnerabilities at machine speed and scale, and most defenders are still working at human speed on tools built for a slower adversary. That is a capability gap, and it's widening.

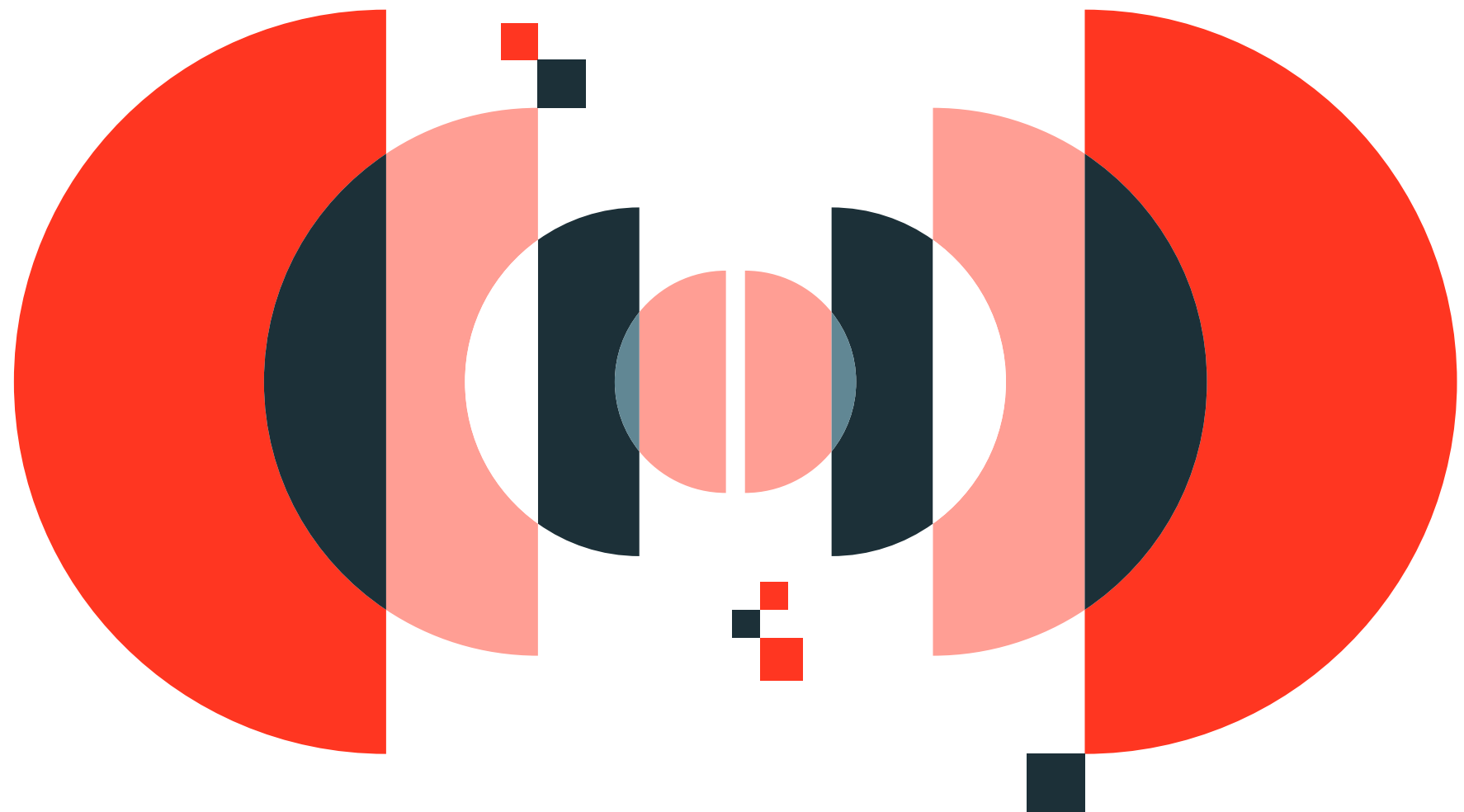
To understand the real-world impact of this gap, consider the following typical timeline. A serious flaw is disclosed late on a Friday afternoon. In the hours that follow, attackers point powerful models at the advisory, generating a working exploit, perhaps before the on-call engineer has even seen the news. By Monday, the team is still scoping the organization's exposure, but the exploit is already circulating and causing damage. This adversary capability should be expected and accounted for in threat models. The old patch process can't keep up.

The old model assumed a buffer between disclosure and exploitation, which offered time to triage, test and roll out a fix before the flaw would likely be used. AI is compressing that buffer dramatically. When exploitation can begin the day a vulnerability is published, a patch process measured in days or weeks leaves the door open. Teams should model exposure as starting from the moment a flaw goes public.

Acting inside that window also takes visibility, and most programs don't have a consolidated view of their attack surface. To prioritize, a team has to see what is exposed, what an attacker can reach and what is already being hit, yet those answers sit in different tools. The **2025 Pulse of the AI SOC report** found that 88% of security teams run 10 or more separate products for detection, investigation and response. Each holds a slice of the picture, so assembling the whole picture is slow, and another product adds another slice, not a comprehensive view. Security teams are left owning that exposure and answering to a board for it, while inheriting a stack that can't close the gap on its own.

The **Mythos-Ready briefing**, a joint emergency response from the Cloud Security Alliance, SANS, the OWASP GenAI Security Project and [un]prompted, reaches the same conclusion. It classifies human-speed detection and response against machine-speed attack as a capability gap, and its first critical action — pointing AI agents at code and pipelines — is the operating model this guide builds out.

Closing the capability gap comes down to two changes — what the program measures and where its data lives. The chapters that follow take each in turn.



## From Patch Cycle to Exploit Window

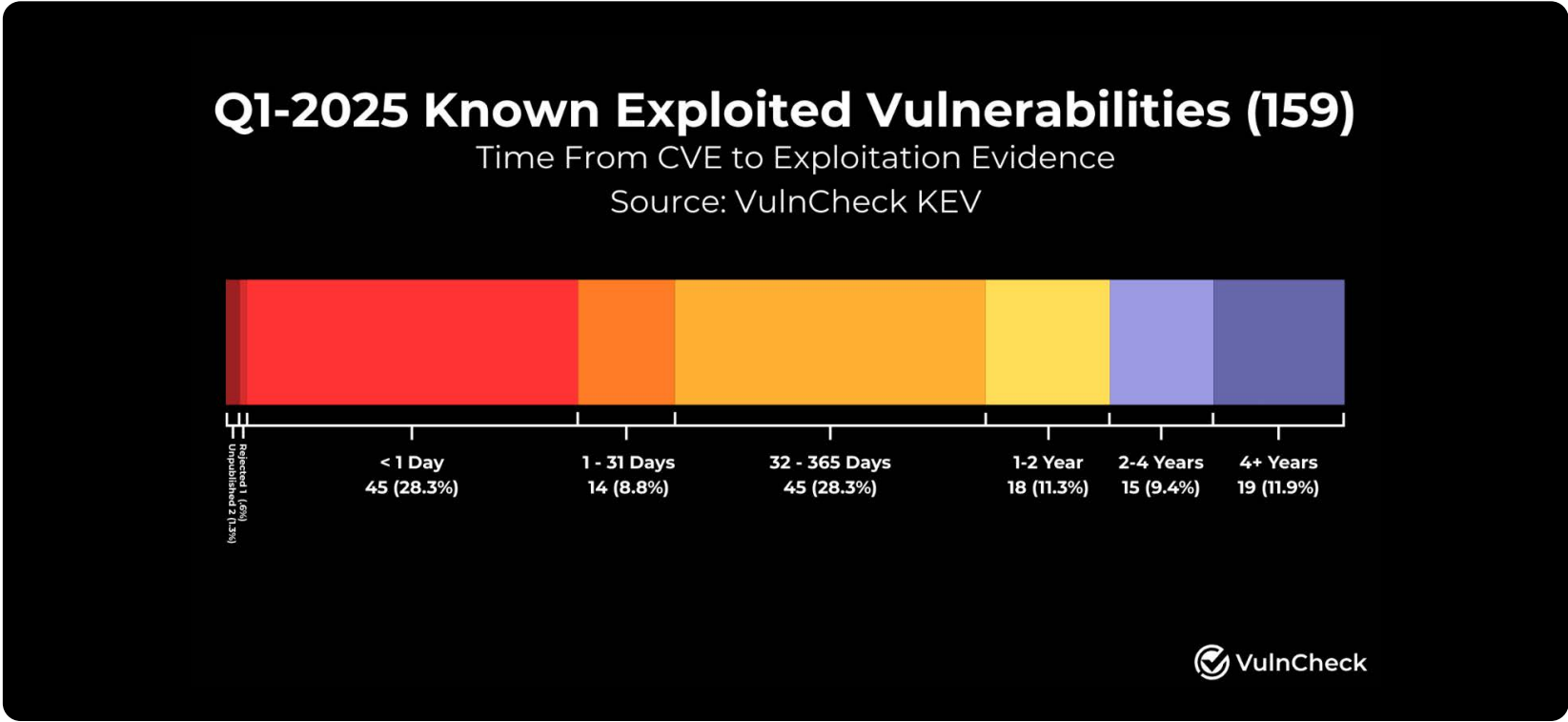
The patch cycle and the exploit window measure different things, and a program built around the wrong one optimizes the wrong work.

- **Patch cycle** — Tracks an internal process, useful for hygiene but blind to real exposure
- **Exploit window** — Tracks what actually causes harm, the time a known vulnerability stays reachable and unpatched while a working exploit exists

Two slow steps dictate the duration of the exploit window. The first is finding where the exposure actually exists. The second is getting the fix onto every vulnerable asset. The window doesn't close when a patch ships — it closes when the patch is live everywhere, or the risk is otherwise mitigated. On a large estate, that last mile can take days or weeks. Patch availability and patch deployment aren't the same milestone.

Compressing the exploit window is a fundamentally different job than patching faster. The work is to find those few flaws that are both under active exploitation and reachable in the environment, and fix them first. Many never qualify. **VulnCheck** counted 159 exploited vulnerabilities in the wild in the first quarter of 2025, a small fraction of the thousands disclosed.

CVSS measures severity, not the probability of a successful attack. The flaws ranked as highest severity are often never exploited, while those under active attack often land in the medium range. This is one reason why CISA maintains the **Known Exploited Vulnerabilities catalog** and FIRST created **EPSS** to estimate the exploitation odds CVSS leaves out. A backlog built around CVSS prioritizes severe-on-paper flaws while not addressing the few under attack.



The **Mythos-Ready briefing** recommends turning this ongoing work into a permanent capability, which it calls VulnOps. It describes a vulnerability operations function, staffed and automated in a way modeled after DevOps. The briefing rates it critical on a six-to-twelve-month horizon, which makes it a proposal rather than an established practice.

As the authors frame it, VulnOps turns on one continuous question: Given everything disclosed and everything the organization runs, where do real exploitability and real exposure actually coincide, and what closes that gap fastest? Answering that at machine speed requires two things many programs still lack: the relevant signal in one place and automation that can act on it without waiting to be asked.

Two findings land in the same scan, both critical. One is a CVSS 9.8 in a library buried in an internal tool that nothing outside can reach. The other is a medium-rated flaw in an internet-facing service already being exploited in the wild. Sorted by score, the 9.8 takes priority. Sorted by exposure, the second one does, and that's the call VulnOps exists to make obvious.

## Widespread Threats

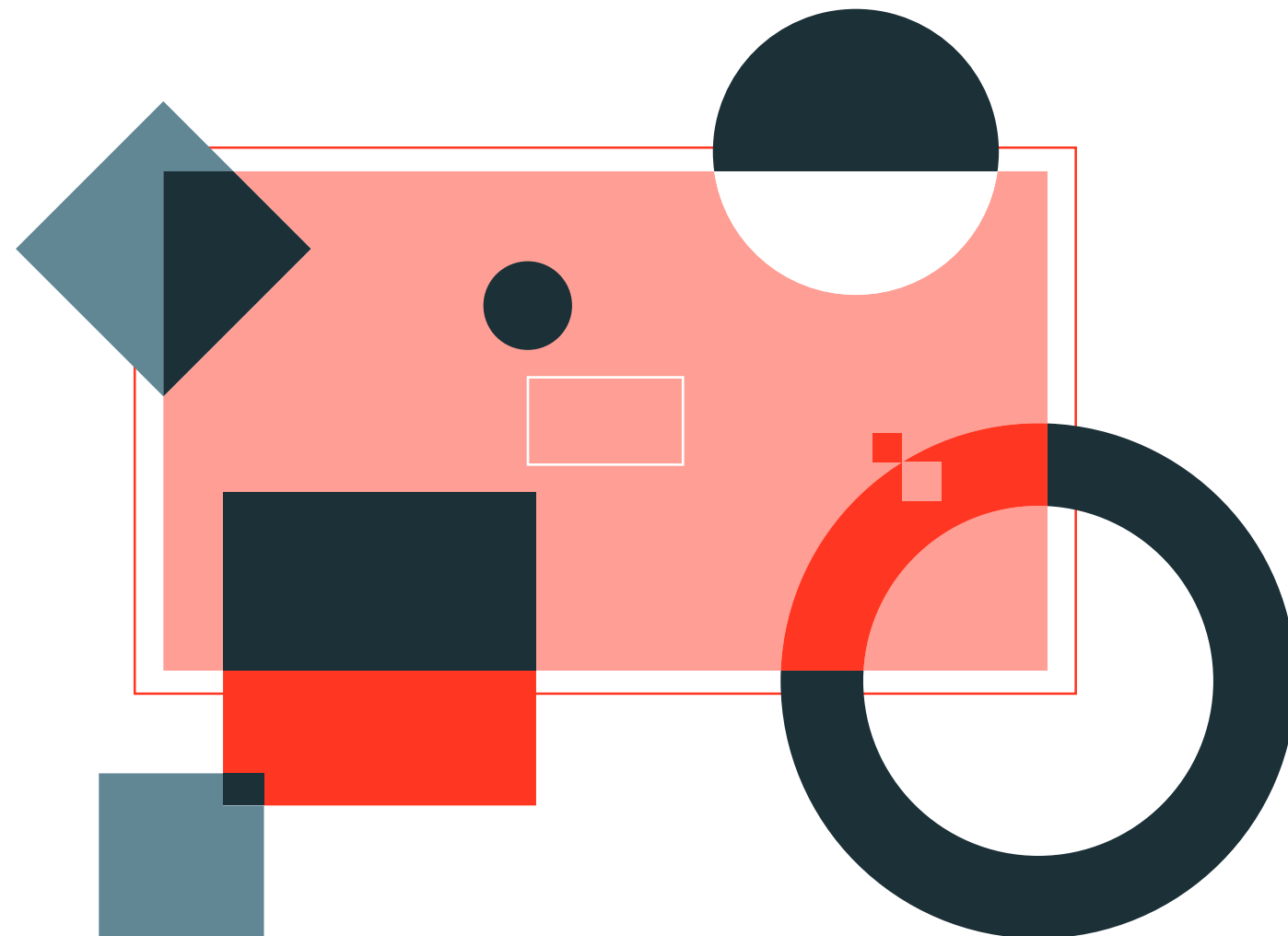
Everything so far describes ordinary vulnerability management, the steady queue a team works through daily. A flaw on the scale of **Heartbleed**, **Log4Shell** or the 2025 SharePoint **ToolShell** breaks the routine. When one hits, it's all hands on deck. That means several workstreams running in parallel.

A typical response in a large organization runs roughly like this:

- **Deploy patches across the estate:** The obvious step, and still a slow one, because patch availability and full deployment are different milestones
- **Assess exposure:** Where does the organization run the vulnerable component, and where does its own code call it? Those answers live in two different places: asset and configuration data on one side, and source and dependency data on the other. The nature of the vulnerability decides which matters more.
- **Assess compromise:** Query the ingested telemetry, spanning the retained history, for indicators that someone had already exploited this before the patch landed. Exposure says what could happen, while telemetry says what already happened.
- **Stand up quick-and-dirty detections:** Get rules in place for exploitation attempts within hours, accepting that they'll be noisy
- **Replace them with sustainable detections:** Once the panic passes, swap the throwaway rules for durable, maintained detection-as-code
- **Adjust the practices, policies and controls:** Implement the longer-term work that reduces the likelihood of exposure to similar flaws in the future

Each of these steps needs the same thing: the relevant data in one place and the ability to act on it fast. Exposure assessment needs asset, SBOM and code access. Compromise assessment needs deep, searchable telemetry with adequate retention. Detection engineers need a platform to experiment, author and deploy rules against that telemetry. Running this across dozens of disparate consoles loses the coordination race, not the analysis.

This is also where the case for automation gets concrete. A human team works these streams mostly in sequence and is stretched thin trying to cover all of them at once. Defensive AI agents, still an emerging capability, run several of these in parallel: exposure assessment, telemetry hunting and a first pass at detection rules, which can be the difference between a week of exposure and a day.



## The Security Lakehouse

The exploit window can't be compressed with data scattered across a dozen consoles. The first move is architectural. A fast response needs that signal on a single open foundation.

A security lakehouse consolidates the inputs that VulnOps actually reasons over. Vulnerability scan results. Software bills of materials. Asset inventory and patch state. Cloud configuration and workload telemetry. Threat intelligence on what's being exploited in the wild. Endpoint, network and detection signal. When those sit in one place in open formats, the intersection question becomes a query rather than a multi-team fire drill.



Three properties make the lakehouse the right foundation for this work.

- 1. Open, normalized data.** Land high-volume telemetry on an open standard rather than a proprietary store. The **Open Cybersecurity Schema Framework** (OCSF) provides a common structure, so a scan finding, an asset record and an exploit signal can be combined without a custom integration for each pair. Open formats also prevent data from being held hostage to a vendor's ingest pricing or query language.
- 2. Data retention you can actually afford.** Separating storage from compute changes the economics of keeping data hot. Investigations that reach back months or years stop being a restore-from-cold project. Multiyear searchable retention at a fraction of legacy SIEM cost is the practical difference between answering "Has this exposure ever been touched?" and guessing.
- 3. Governance that travels with the data.** Consolidation only helps if it doesn't create a new risk. **Unity Catalog** governs the security data alongside the rest of the enterprise data, with lineage, access control and audit on every table. That governance becomes load-bearing once AI agents start reading and acting on this data. The **Databricks AI Security Framework** (DASF 3.0) maps the specific controls this guide names to their application areas.

The same foundation doesn't stop at vulnerability operations. The detection engineers tuning their rules, the analysts running investigations and threat hunts, the incident responders reconstructing what happened and the GRC team proving controls to an auditor all draw on the same governed telemetry. Consolidating it once, in the open, lets every one of those functions work from a single source instead of its own silo.

## AI Agents for Defense

Attackers use AI to find and weaponize flaws faster than a security team can review code by hand, so defenders need agents working the same surface, continuously, on the same timescale.

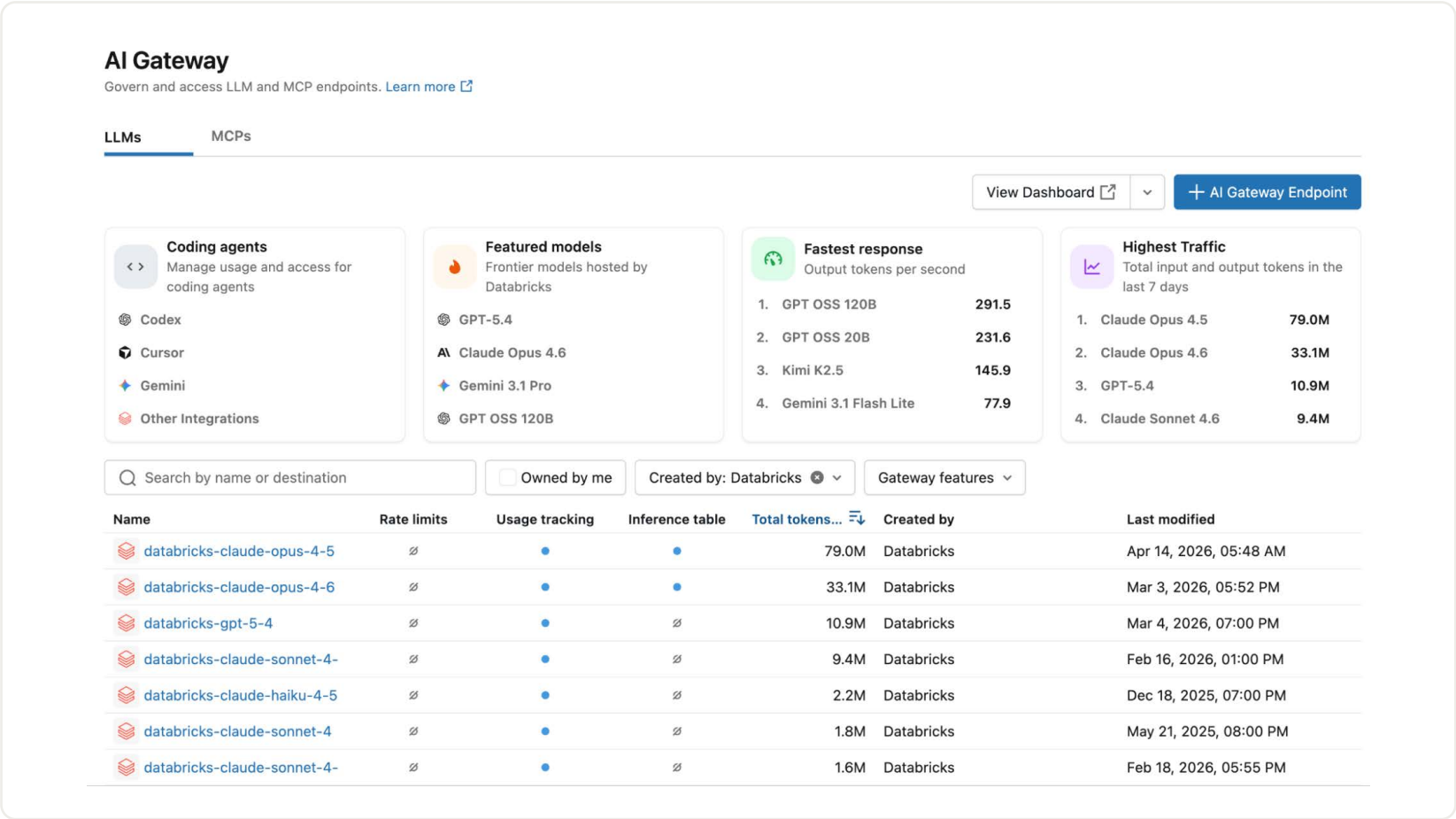
A defensive agent does work that doesn't scale with headcount.

- It reads code and runs static analysis (SAST) and software composition analysis (SCA) against it
- It takes the raw findings and validates exploitability, using an LLM to reason about whether a flagged issue is actually reachable and weaponizable in context rather than a theoretical match. This is the step that cuts the flood of false positives that buries most scanning programs.
- It writes the validated findings into the lakehouse as a governed data source, where they join the asset, exposure and threat-intel signals already there

Once findings land in the lakehouse, the intersection of exploit and exposure becomes clear. A validated, reachable flaw in an internet-facing service running a component with a known in-the-wild exploit rises to the top on its own, because all the signal needed to rank it is in one query.

The code-scanning agent is just one example. The same pattern applies in security operations. A governed agent reads from the lakehouse and writes back to it, whether it's triaging and enriching alerts, running a first pass at a threat hunt, drafting and tuning detections or summarizing an incident for the responder who picks it up. Each takes on work no team can staff for, and each lands its output where the next function can use it.

Now the hard part, and the one most rollouts skip: The agents are themselves an attack surface, and pointing an LLM at source code and pipelines is exactly the kind of high-trust deployment that goes quietly wrong. Governing the agents is not optional.



Best practices for securing and governing agents include:

- **Use an AI gateway:** Every model call the agent makes goes through a control point that logs it, rate-limits it and applies guardrails. DASF 3.0 calls this out directly (the AI Gateway control).
- **Capture agent telemetry:** Inference logs from the agents return to the lakehouse, making the agents' behavior observable, auditable and detectable. DASF audit-log and lineage controls apply here, and this telemetry is what lets a team catch an agent that has been manipulated.
- **Enforce least privilege:** Limit the agent's access to the minimum it needs (the DASF control for limiting AI and agent access). An agent that can read a repository shouldn't also be able to write to production.
- **Model failure modes:** The **OWASP Top 10 for LLM Applications** covers direct risks, with prompt injection (LLMO1) the most obvious, where malicious instructions hidden in the code the agent reads hijack what it does. For multistep agentic behavior, **OWASP MAESTRO** provides a layer-by-layer threat model, and the **CSA agentic AI red teaming guidance** gives a way to test it adversarially before it touches anything that matters.

## Summary

Security programs should stop measuring how fast they patch and start measuring how long they're exposed. Put the signal that answers that question in one open, governed place. Then put defensive AI agents on top of it to do the continuous work no human team can keep pace with, and hold those agents to the same controls as any other privileged access.

The board conversation changes once a program makes this shift. Previously, "Are we patched?" could only be answered with "Mostly, eventually." Now, "How exposed are we in the window that matters, and is that window shrinking?" can be answered with a number the team watches trend down. That's the difference between hoping the organization is covered and showing it's resilient.

Three metrics make that visible:

1. The median exploit window across the estate, measured from disclosure to mitigation, rather than to patch availability
2. The share of validated, reachable findings closed inside a target window
3. How long it takes, after a major disclosure, to answer whether the organization was already impacted

A program that can move those numbers, and show them moving, has closed the capability gap.

## Call to Action

See the agentic SIEM built on the security lakehouse

Read the Lakewatch press release

Read the Lakewatch launch blog

Watch Ali Ghodsi's RSA keynote

## About Databricks

Databricks is the data and AI company. More than 20,000 organizations worldwide — including adidas, AT&T, Bayer, Block, Mastercard, Rivian, Unilever and 70% of the Fortune 500 — rely on Databricks to build and scale data and AI apps, analytics and agents. Headquartered in San Francisco with 30+ offices around the globe, Databricks offers a unified platform that includes Lakebase, Genie, Agent Bricks, Lakeflow, Lakehouse and Unity Catalog. To learn more, follow Databricks on [LinkedIn](#), [X](#), [YouTube](#) and [Instagram](#).

