

ManageEngine
Endpoint Central

Patch mayhem: When vulnerability discovery massively outpaces remediation

*Defending against **AI-accelerated exploitation***

Table of contents

Executive summary 03

Threat landscape: Vulnerability discovery at machine speed and the collapsing exploit window 04

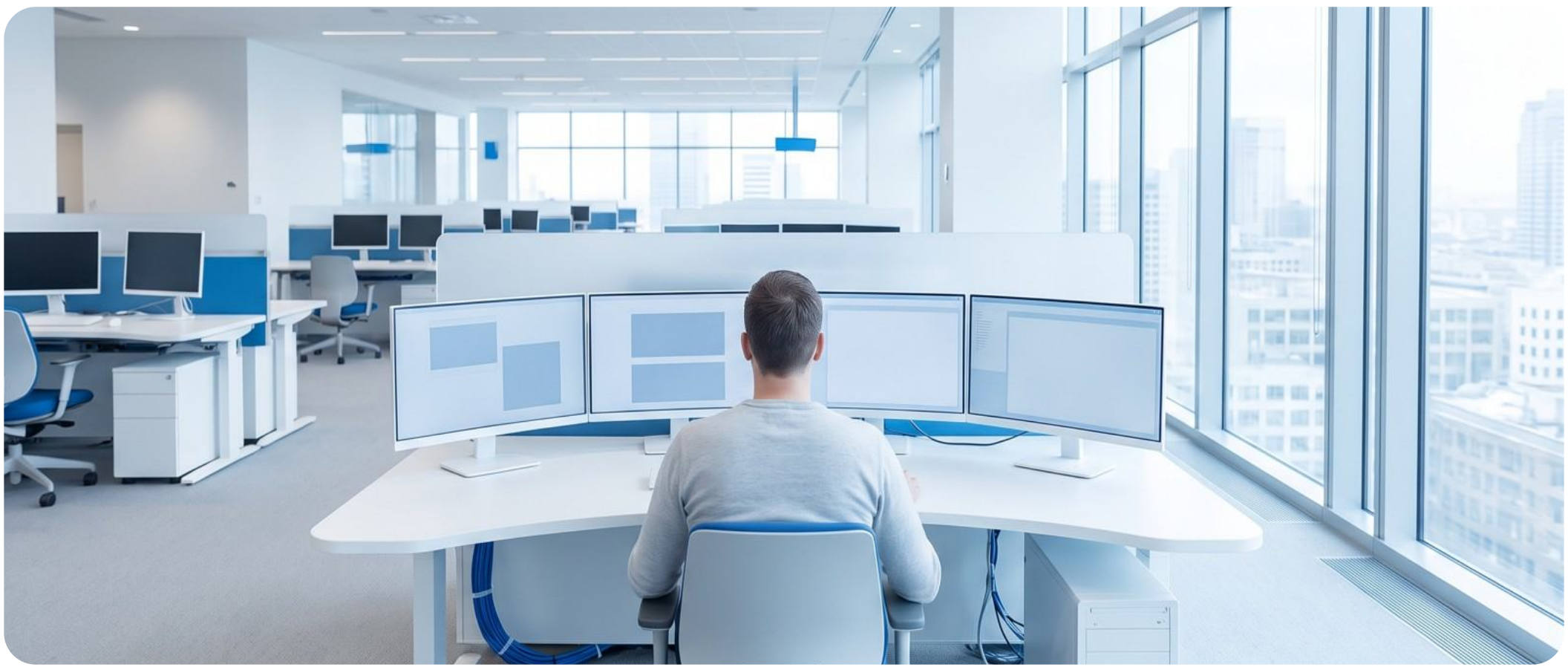
- The surge: vulnerability discovery at machine speed 04
- The exploit window is collapsing 06
- Three answers, one problem 07

Defending against AI-accelerated exploitation 08

- Patching, reimagined 08
  - Prioritization beyond CVSS 08
  - Patch reliability scores 09
  - Autonomous patch testing: dynamic test bed creation 09
  - Deployment rings and idle-time deployment 10
  - One platform, not a handoff between two 11
- Attack surface management 11
  - Policies that self-correct drift 11
- Runtime defense 12
  - Detection has to be behavioral, not just signature-based 12
  - Closing the loop back into remediation 12

The three layers, one platform 13

Conclusion 14



Executive summary

Vulnerabilities being exploited before a fix exists isn't new. It's the definition of a zero-day: A threat actor finding a flaw before a vendor does. Historically, this kind of attack was rare and resource-intensive, being the product of a skilled researcher, threat actor, or nation-state investing days, sometimes weeks, on a single target.

Frontier AI models have changed that equation. Claude Mythos, Anthropic's unreleased frontier model, has automated vulnerability discovery at a scale and speed with no historical precedent. Anthropic has deployed it defensively first, through Project Glasswing, before making any version of the capability public. Around **50 organizations**, including major technology vendors and critical infrastructure operators across power, water, healthcare, and telecommunications, now use Mythos Preview to find flaws in their own software before attackers do. The result has already surfaced **over 10,000 high- and critical-severity vulnerabilities**, some undiscovered for decades.

This containment is deliberate. **Anthropic** has cited the model's offensive capability as exceeding its defensive utility if released without safeguards, stating that full proliferation of Mythos-like capabilities will be inevitable at the rate AI is progressing. When it arrives, the advantage currently held by defenders will be available to attackers as well.

But the more immediate problem is not the future risk. It is the present one. Defensive use of Mythos-class discovery has already produced a surge in vulnerability disclosures and patches, arriving at a pace most enterprises were never built to absorb.

Patch management, on the other hand, **remains largely manual**—or **a little automated at best**—across industries. Even before frontier AI models, organizations have been behind on patching: **nearly 60% of breach victims report** being compromised through a vulnerability for which a patch was already available. That gap has since widened. Mandiant's 2026 M-Trends report, drawn from over 450,000 hours of incident response, found the mean time to exploit across observed campaigns has gone negative, exploitation now beginning an average of seven days before the vulnerability is even publicly disclosed, let alone patched.

Increasing patching velocity alone will not close this gap. What is required instead is a **structural shift across three complementary layers.** The first is reducing the attack surface itself, so fewer vulnerabilities are relevant to begin with. The second is detecting and containing compromise at runtime, so that a gap in coverage does not automatically become an incident. The third, and the primary focus of this white paper, is remediating what remains through risk-based prioritization: ranking assets by actual risk, prioritizing which vulnerabilities matter before allocating remediation effort, and moving testing, approval, and deployment toward autonomous execution wherever the risk of doing so is acceptable.

AI and the threat landscape: Vulnerability discovery at machine speed and the collapsing exploit window

Two shifts define the vulnerability landscape from April 2026 onward:

Surge in vulnerability discovery

Disclosure is now happening at a volume and speed that has no historical precedent, driven by discovery moving from manual research to machine-speed automation.

Collapse of the exploit window

The time between a vulnerability becoming known and being used against an organization has collapsed significantly. For a growing share of vulnerabilities, it has shrunk to days, or vanished entirely.

The surge: Vulnerability discovery at machine speed

For nearly a decade, vulnerability disclosure grew at a pace security teams could plan around. The year 2025 set a **record** with 48,185 published CVEs, a 21% increase over the prior year, bringing the cumulative catalog past 308,000. That growth was already straining triage capacity, but it was gradual enough that annual forecasts, staffing plans, and patch cycles could adjust to it year over year.



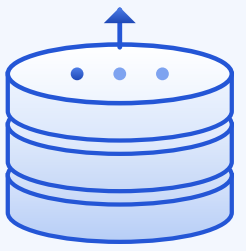
48,185

published CVEs in 2025



21%

increase over the prior year



308,000

cumulative catalog passed

Image source: CVEdetails.com

Raw CVE counts have never been a reliable measure of actual risk. CVEs are issued by CVE Numbering Authorities (CNAs), a federated network of vendors and projects, each deciding independently which flaws warrant an identifier. That threshold varies widely: WordPress plugin security firms and the Linux kernel file CVEs thoroughly even for minor issues, while other ecosystems report far more selectively.

Reporting-practice differences can sometimes inflate raw vulnerability counts without any corresponding rise in actual risk. However, the surge following the release of Claude Mythos Preview cannot be explained away on those grounds. This was not simply a rise in volume. It was a rise specifically in high- and critical-severity findings as shown in the figure below, breaking every prior pattern across every available measure.

High- and critical-severity CVE disclosures

Monthly disclosures across 21 notable organizations

Image source: Epoch AI



Pre-Mythos monthly record

By June 2026
more than three times the prior record

By July 2026
nearly five times the pre-Mythos baseline

Epoch AI's tracking of monthly high- and critical-severity CVE disclosures across 21 notable organizations shows the pre-Mythos monthly record at roughly 490. By June 2026, that figure had reached approximately 1,550—more than three times the prior record. By July, it reached approximately 2,500—nearly five times the pre-Mythos baseline and 60% above June's own record-breaking total. This was not a gradual climb. It was a steep change immediately following the release of Claude Mythos Preview.

The industry's own forecasting could not keep pace with the shift it was trying to predict. Even Patch Tuesday, historically one of the more predictable rhythms in enterprise security, broke its own pattern: after holding between 56 and 172 CVEs a month for 18 consecutive months, **July 2026 delivered 569, the largest Patch Tuesday on record.**

Forum of Incident Response and Security Teams (FIRST) February 2026 forecast projected roughly 59,427 CVEs for the year, the first projection ever to exceed 50,000. By its June revision, actual disclosures were already running 46.3% ahead of that projection, forcing FIRST to raise its full-year estimate to approximately 66,000. As FIRST's own liaison put it,

> **"We're witnessing a major shift in the vulnerability landscape, not because software is suddenly less secure, but because our collective ability to find flaws has been structurally transformed."**
>
> **FIRST** liaison — Forum of Incident Response and Security Teams

Discovery volume did not simply increase. It decoupled from the system's capacity to name, score, and route what it found, at every layer of the pipeline.

At the intake layer, GitHub's CNA requests reached almost 4,000 in May alone, nearly ten times the prior year, and the organization reported it had not consistently met its own publication goals since mid-April despite record output.

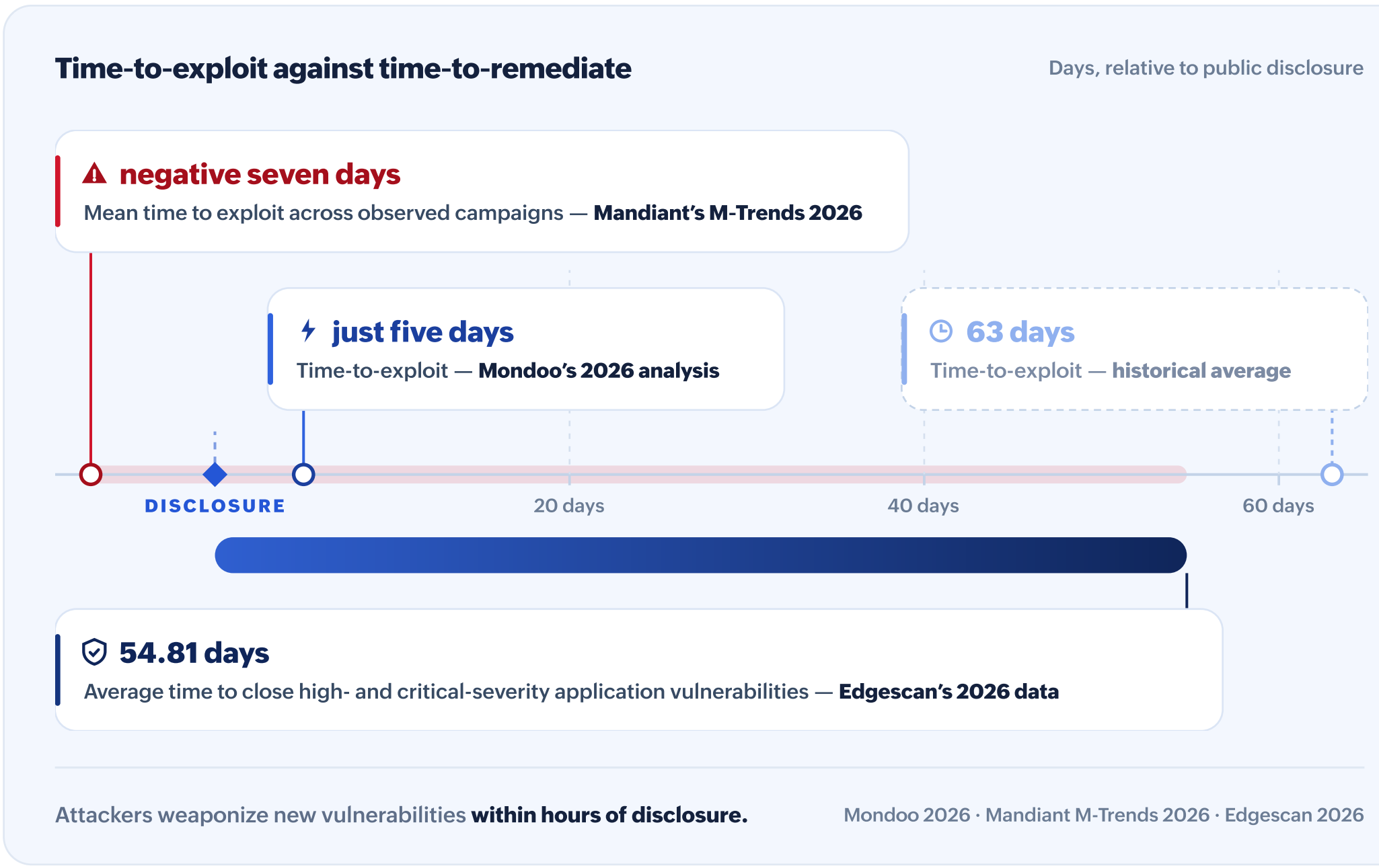
At the enrichment layer, NIST formally changed how the National Vulnerability Database (NVD) operates on April 15, 2026, limiting full enrichment to CVEs meeting specific criteria and citing the availability of frontier models, specifically Claude Mythos and GPT-5.4-Cyber, as a driver of a submission flood expected to only grow stronger. Post-policy **NVD enrichment coverage** is now estimated at just 15 to 20% of incoming CVEs, meaning a growing share of new disclosures arrive without the severity scoring most automated triage pipelines depend on.

The exploit window is collapsing

Volume explains why the patching queue has grown. It doesn't explain why it can no longer be worked through in order. That requires the second shift: **How much time now exists between a vulnerability becoming known and being used in an attack.**

That window used to absorb delays. A vendor disclosed a flaw, a patch typically followed, and defenders had days to weeks to test and deploy before attackers caught up. Mondoo's 2026 analysis puts a number on how far that buffer has collapsed: time-to-exploit has fallen from a historical average of **63 days to just five.** For a meaningful share of vulnerabilities, it has gone further than that. Mandiant's M-Trends 2026, drawn from over 450,000 hours of incident response, found the **mean time to exploit across observed campaigns is now negative seven days,** exploitation beginning before the CVE is even disclosed.

The consequence is a direct mismatch. Edgescan's 2026 data shows **attackers weaponizing new vulnerabilities within hours of disclosure, while high- and critical-severity application vulnerabilities still take an average of 54.81 days to close.** Hours to exploit against nearly two months to remediate is not a gap that faster patching alone closes. For a large and growing share of vulnerabilities, the deadline has already passed before remediation work begins.



Three answers, one problem

Faced with a queue that has outgrown its infrastructure and a window that has collapsed toward zero or past it, the industry has not converged on a single response to this shift. **Three distinct positions exist:**

Risk-aware, autonomous patch management

Out of tens of thousands of new disclosures a year and a window that has gone to zero or negative, what gets fixed first, by whom, and how much of that process can safely run without a human in the loop at every step.

Reducing the attack surface

A vulnerability that doesn't exist in an operating environment can't be exploited. **Sidero Labs' Talos Linux**, a minimal Kubernetes OS with under 50 binaries and no shell, SSH, or package manager, proves the principle by **eliminating entire CVE categories outright.** Full rebuilds aren't realistic for most enterprise estates, but inventorying and trimming unnecessary components still meaningfully cuts exposure.

Runtime detection and defense

A third position holds that prevention alone cannot close a gap this wide, and that security investment should re-balance meaningfully toward runtime detection and response, catching what remediation could not prevent in time.

These positions are not mutually exclusive. Reducing unnecessary attack surface, detecting compromise at runtime, and remediating through risk-based, appropriately autonomous patching are three layers of the same defense, not three alternatives to choose between.

Defending against AI-accelerated exploitation

Patching, reimagined

The threat landscape section established what patch management is now up against: a queue that has outgrown its infrastructure, and a window that has (for a meaningful share of vulnerabilities) already closed before remediation work begins. Faster patching alone cannot solve either problem. Instead, organizations need a clearer answer to a simpler question: **how much of remediation should run on its own, and what needs to be verified before it's trusted to?**

Prioritization beyond CVSS

Ranking vulnerabilities by CVSS score alone is no longer defensible. CVSS measures theoretical severity, not actual exposure. A critical-severity flaw in a system with no internet-facing footprint and no path to sensitive data is a lower real-world risk than a medium-severity flaw sitting on an internet-facing asset with a public exploit already circulating. Effective prioritization has to fuse three signals that most environments currently keep in separate tools:

Exploit intelligence

KEV listing, EPSS score, or evidence of active exploitation.

Asset context

What the system does, what it connects to, and whether it is internet-facing.

Business criticality

What breaks, and for whom, if this asset goes down or is compromised.

Risk-based prioritization by itself is table stakes now. Every serious vendor in this category has shipped some version of it. What separates a genuinely useful prioritization engine from a checkbox feature is what happens after the ranking exists, and whether the system can act on that ranking with appropriate speed and appropriate caution, which is where the rest of this section is aimed.

Patch reliability scores

A ranked list says what to fix first. It does not say whether fixing it will work cleanly. This is the gap that patch reliability scoring closes: a predictive signal, built from deployment telemetry across similar configurations, that estimates how likely a given patch is to install cleanly, cause a conflict, or require a rollback, before it ever reaches a production endpoint.

This matters because it directly answers a fear that has historically slowed remediation more than any CVSS score ever did: the fear that patching will break something. A reliability score turns that fear into a number an operator, or an automated policy, can act on. A patch with a high reliability score on a given configuration is a strong candidate for expedited, low-friction deployment. A patch with a low or uncertain score is a candidate for additional validation before it goes anywhere near production, not a reason to delay it indefinitely.

Autonomous patch testing: Dynamic test bed creation

Static test groups, a handful of representative machines checked before every release, do not reflect the actual diversity of a modern enterprise estate. The gap this leaves is that a patch can validate cleanly against a test machine while still failing against a real production configuration the test group never modeled.

The more defensible architecture is dynamic test bed creation: creating an environment that mirrors the production environment closely, at the moment a patch needs validating, rather than relying on a fixed set of test machines. Because the environment is generated on demand, testing can scale to the actual diversity of the fleet, not just the diversity someone anticipated when the test group was built.

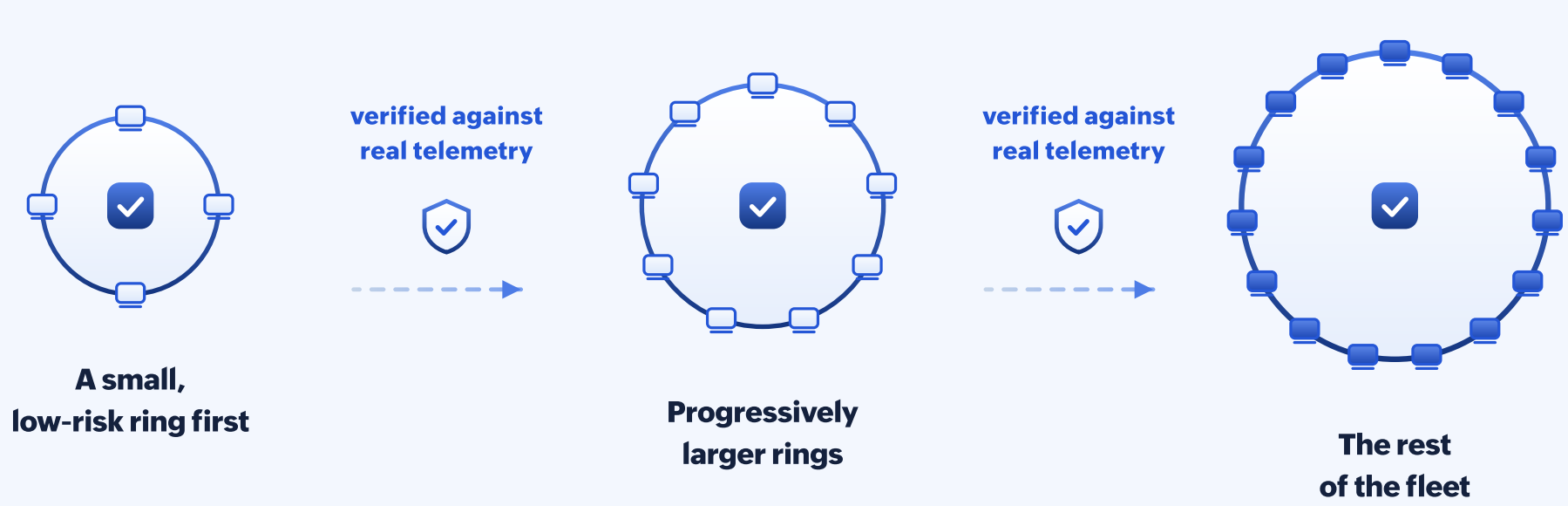
This is where reliability scoring and dynamic testing reinforce each other directly.

> **A reliability score built purely from an organization's telemetry is strong. A reliability score confirmed against a test bed that mirrors the actual configuration is a verified result. The first is enough to justify accelerated handling. The second is what justifies unattended execution.**

Deployment rings and idle-time deployment

This is where the execution autonomy question raised earlier stops being theoretical and becomes an operating model. **Factors such as reversibility, blast radius, and verifiability** are not abstract design principles. Deployment rings are the direct mechanism for managing blast radius: a patch reaches a small, low-risk ring first, its outcome is verified against real telemetry, and only then does it expand to progressively larger rings, with each stage functioning as a checkpoint. A failure at ring one affects a handful of machines and is caught before it ever reaches the rest of the fleet. This is the practical answer to the finding raised earlier: that the permission granted to a system, not the sophistication of the model behind it, is what determines whether autonomy produces harm.

Each stage functions as a checkpoint



A failure at ring one affects a handful of machines and is caught **before it ever reaches the rest of the fleet.**

Idle-time deployment addresses reversibility and disruption from a different angle. Rather than pushing patches during fixed maintenance windows that frequently collide with active work, deployment can be scheduled against actual endpoint signals—power state, network conditions, current usage—so that installation happens when a device is genuinely idle. This is not primarily a convenience feature. A patch that installs during genuine idle time is less likely to be interrupted mid-installation, which is itself a meaningful reliability variable that most patch management conversations ignore in favor of talking only about scheduling windows.

Together, rings and idle-time deployment are what let an organization grant real execution autonomy without granting it recklessly: small blast radius, verified before expansion, executed when the risk of interruption is lowest. That is the concrete, operational version of the framework this whitepaper argued for earlier, autonomy earned per action, not claimed wholesale by a product tier.

One platform, not a handoff between two

Here's a structural point worth stating plainly, because it explains a bottleneck that ranking and automation alone do not fix: vulnerability detection and patch remediation living in separate tools, often from separate vendors, is itself a source of delay independent of how fast either tool individually operates. Industry research on this exact bottleneck found that **a majority of** organizations cite coordination between detection and remediation, not the absence of a priority list, as their primary obstacle. A scanner finding a flaw and a patch engine fixing it are, in most environments, connected by an export, a ticket, and a wait, not a continuous signal.

When detection and remediation share a platform, that handoff disappears. A newly discovered vulnerability does not need to be exported into a ticketing system to reach a remediation queue, it is already in one, already scored, already ranked against the same asset context the detection engine used to find it in the first place.

Attack surface management

The threat landscape section made a point about attack surface reduction: what isn't there can't be exploited. But rebuilding the entire environment around minimalistic OS versions isn't realistic for most enterprises running everyday business software. So here's the practical version of that same idea: cutting exposure continuously, without having to rebuild the fleet to do it.

Policies that self-correct drift

Most hardening programs are checklists, applied once, verified at audit time, and quietly decayed by every subsequent change made outside that process: a new application installed, a port opened for a one-off project, a setting reverted during troubleshooting and never restored. The checklist says the environment is hardened. The environment, six weeks later, often is not.

The more defensible model treats a security policy as a continuously enforced state rather than a one-time action: the policy is checked against the live configuration continuously. Drift gets caught as it happens and for low-risk settings, fixing that drift is automated. These fixes can be reversed if needed. This is the practical translation of the "have less to patch" principle for an estate that cannot be rebuilt as a minimal image: **a component cannot always be removed, but the configuration around it can be secured.**

Runtime defense

Attack surface reduction and patch remediation both operate before exploitation happens. They are necessary, but, as the threat landscape section demonstrated, they are not enough. A meaningful and growing share of exploitation now happens before disclosure, before a patch exists, before an attack surface reduction policy could have anticipated the specific flaw. Runtime defense is the layer that has to catch what the first two layers structurally cannot prevent: behavior that occurs when there was no patch to apply and no known exposure to have closed.

- **Detection has to be behavioral, not just signature-based**

A vulnerability exploited before disclosure has, by definition, no existing signature to detect it by. Endpoint detection and response (EDR) earns its place in this architecture specifically because it is built to catch behavior, not identity: a process spawning unexpected children, unusual lateral movement, credential access patterns that do not match the account's normal use, regardless of whether the specific exploit has ever been seen or catalogued before. This is the direct, practical answer to a negative exploitation window. If the vulnerability could not be caught before exploitation, the exploitation itself still has to be caught before it becomes impact.

- **Closing the loop back into remediation**

The one thing that most runtime defense tools miss is that EDR should not function as an isolated safety net sitting apart from the rest of the security stack. A detection on a given endpoint is a strong signal that the asset's actual risk has changed, and that signal should feed directly back into the same prioritization engine driving patch remediation, not sit in a separate console requiring a human to manually correlate the two. An asset with an active detection should see its remediation priority elevated automatically, its patch backlog reordered around it, its access posture reassessed. Treating detection, prioritization, and remediation as three disconnected systems, each doing its own job well in isolation, reproduces the **core bottleneck in patch management, just one layer further out.**

The three layers, one platform

Reducing attack surface, detecting compromise at runtime, and remediating what remains are not three separate purchases solving three separate problems. Run as three disconnected tools, they replicate this exact coordination cost: once between vulnerability detection and patch remediation, once between EDR detection and the rest of the security stack. Run as three views into the same asset context, the same risk scoring, and the same reversibility-governed execution model, they compound instead of coexisting. Less standing exposure means less for runtime defense to have to catch. Faster, better-informed detection means remediation can be prioritized before a detection becomes an incident rather than after. And remediation that already understands an asset's exposure, its configuration drift, and its detection history is remediation that can be trusted with real execution autonomy.

This is the architecture **ManageEngine Endpoint Central** was built around, not as three bolted-together modules, but as a single platform where vulnerability detection, patching, attack surface management, and endpoint security share one asset inventory, one risk model, and one execution engine.

In practice, that means prioritization is not a separate step requiring an export into another tool. A vulnerability surfaced by Endpoint Central's detection engine is already scored against the same asset context, exposure data, and criticality used everywhere else in the platform, and already routed into the same remediation workflow, not a ticket waiting to be picked up by a different system.

Patch confidence and reliability signals, built from deployment telemetry across the platform's install base, inform how aggressively a given patch can move through deployment rings—from a small validation ring through to full fleet rollout. The same asset record carries configuration posture, so drift away from a hardened baseline is visible and correctable in the same place a vulnerability would be, rather than in a separate compliance tool checked on its own schedule. Access decisions, including just-in-time elevation, draw on that same device risk posture rather than granting privilege blind to whether the requesting endpoint is currently compliant, exposed, or already under active investigation.

Where Endpoint Central's endpoint security capabilities detect anomalous behavior at runtime, catching what no patch or hardening policy could have prevented because the exploitation happened before disclosure, that detection feeds directly back into the same risk model, elevating remediation priority on the affected asset automatically rather than waiting for a human to correlate two separate consoles.

None of this eliminates judgment. The framework of **execution autonomy earned per action by reversibility and blast radius, not claimed wholesale by a product tier**, still governs what runs unattended and what waits for a human. What a unified platform changes is the cost of getting that judgment right: the information needed to make it is already in one place, already current, and already connected to everything else that judgment depends on.

Conclusion

The vulnerability landscape changed faster in the months following Claude Mythos Preview's release than in the prior decade combined. The exploit window that once gave defenders room to plan has, for a meaningful share of vulnerabilities, closed entirely. Neither trend is likely to reverse.

The organizations that hold up through the next phase of this shift will not be the ones that simply patch faster. They will be the ones that reduced what they had to defend, could see and stop threats that got through anyway, and had a system they could trust to make the remaining calls fast, with accuracy, and only as autonomously as each decision's risk actually allows.

That is the problem Endpoint Central was built to solve. Now all that's left is to ask a question about your own environment:

When the next Patch Tuesday arrives at five times last year's volume, does your organization know, with confidence, what gets fixed first, and why?

If that question isn't easy to answer, our experts are here to help you.

Schedule a demo →

