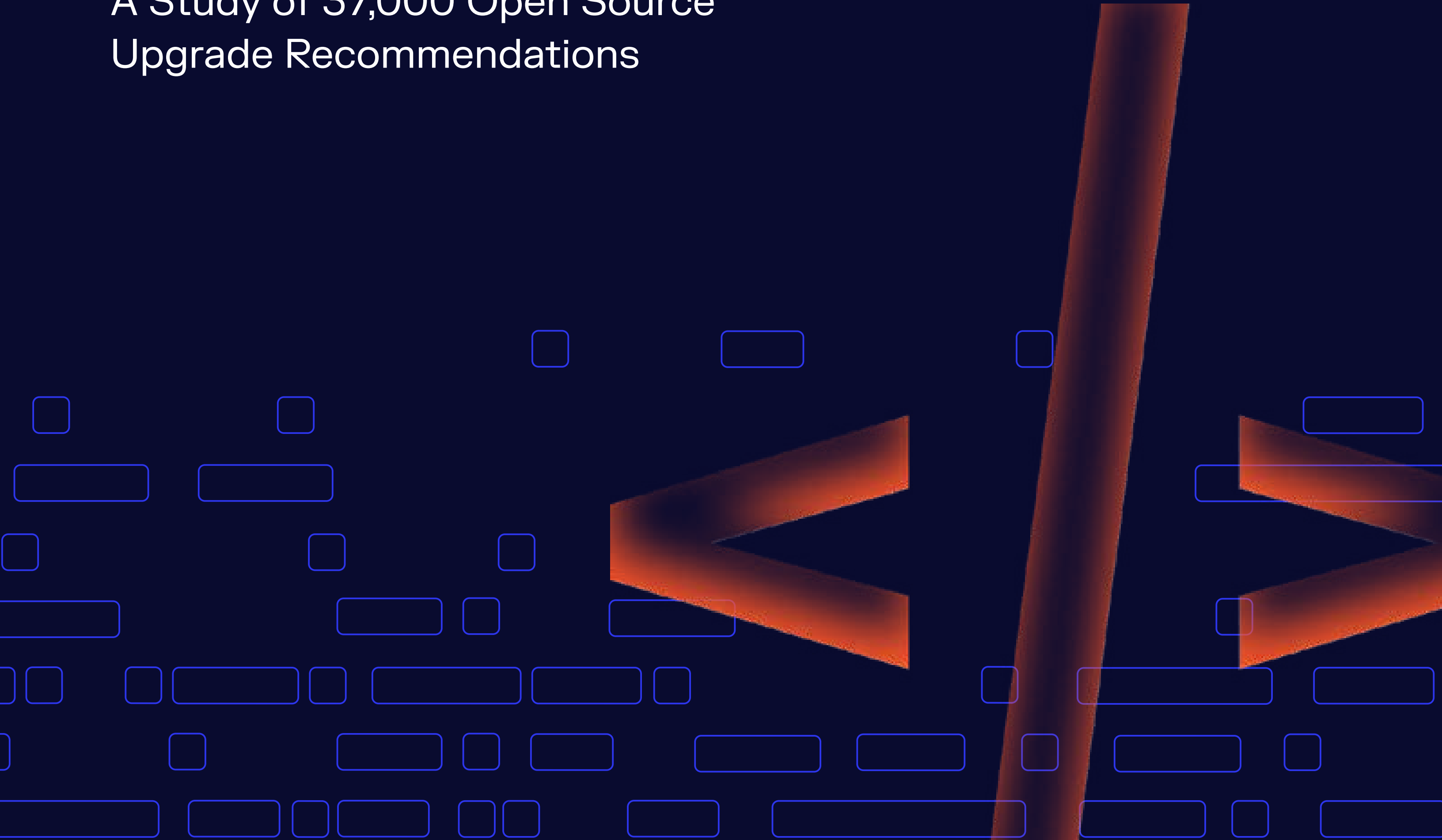
sonatype

Sonatype Research

MAKING AI SOFTWARE DEVELOPMENT SAFE at Machine Scale

A Study of 37,000 Open Source
Upgrade Recommendations



Executive Summary

AI models are becoming highly effective at generating code, but they remain structurally weak at dependency decisions. In Part 1 of this study, published in the 2026 State of the Software Supply Chain Report, Sonatype analyzed 36,870 dependency upgrade recommendations across Maven Central, npm, PyPI, and NuGet against GPT-5 and found that found that it often recommended versions, upgrade paths, or fixes that did not hold up in real software ecosystems. In practice, those failures drive wasted AI spend, wasted developer time, unresolved vulnerability exposure, and technical debt before code reaches production.

This study expands the analysis to newer frontier models, including Claude Sonnet 3.7 and 4.5, Claude Opus 4.6, Gemini 2.5 Pro and 3 Pro, and GPT-5.2, with additional testing on GPT-5 Nano. The goal was to assess whether newer models are improving at dependency remediation, whether larger models produce safer outcomes, and how ungrounded results compare with Sonatype's Hybrid approach that applies real-time software intelligence at inference time.

Frontier models are improving, but not enough to close the security gap. The best ungrounded models still fabricate about 1 in 16 dependency recommendations. At the same time, newer models increasingly recommend "no change" for roughly 1 in 3 components, reducing visible errors but often retaining risk. The most cautious models left 800 to 900 unresolved Critical and High vulnerabilities.

The issue is not model scale but rather ecosystem intelligence. AI models lack the real-time dependency, vulnerability, compatibility, and enterprise policy context required to make safe remediation decisions. Residual risk falls into three groups. About 60–70% can be removed through safe, non-breaking upgrades to Golden Versions. Roughly 20% require an engineering decision. The remaining ~10% reflects cases with no safe upgrade path.

The takeaway is clear:

> **AI models are good at generating code, but they are not equipped to make dependency decisions on their own. Better software outcomes depend on combining model productivity with the real-time dependency intelligence, security policy, and enterprise context those models lack.**

~1 IN 3

COMPONENTS RECEIVED A "NO-CHANGE" RECOMMENDATION

4X

DECREASE IN HALLUCINATED COMPONENT VERSIONS IN NEWER MODELS

60–70%

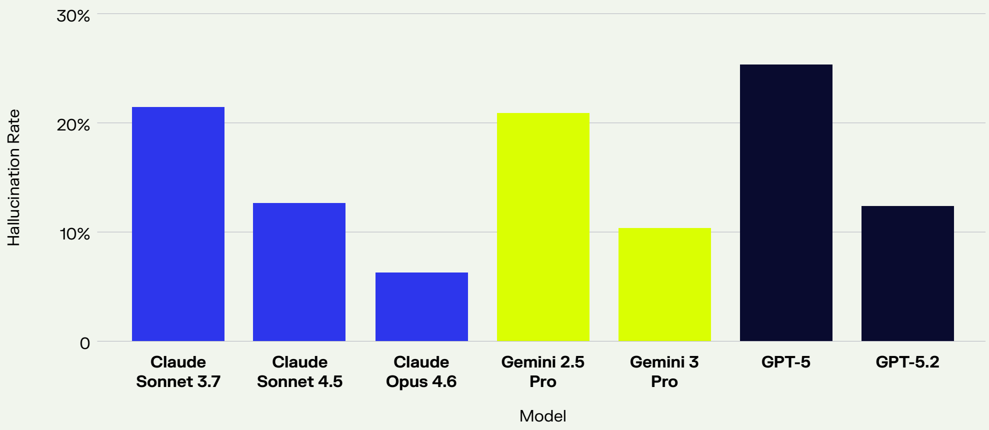
RISK REDUCTION WHEN MODELS ARE GROUNDED IN REAL-TIME INTELLIGENCE

Data Hallucinations in Generative AI Software Development Across Seven Models

Figure 1 shows that the earliest frontier models routinely produced upgrade suggestions that could not exist in the real world. Instead of selecting from published releases, they often generated version numbers that no registry would resolve, a failure mode that wastes tokens and increases build time.

FIGURE 1

Hallucination Rates by Model



4x

DECREASE IN HALLUCINATED COMPONENT VERSIONS IN NEWER MODELS

At the rates observed in previous models (around one in four recommendations for models like Claude Sonnet 3.7, Gemini 2.5 Pro, and GPT-5), hallucinations are not an edge-case anomaly. It meant that version selection itself was unreliable without external validation. Now, the rate of hallucinations has decreased almost 4x.

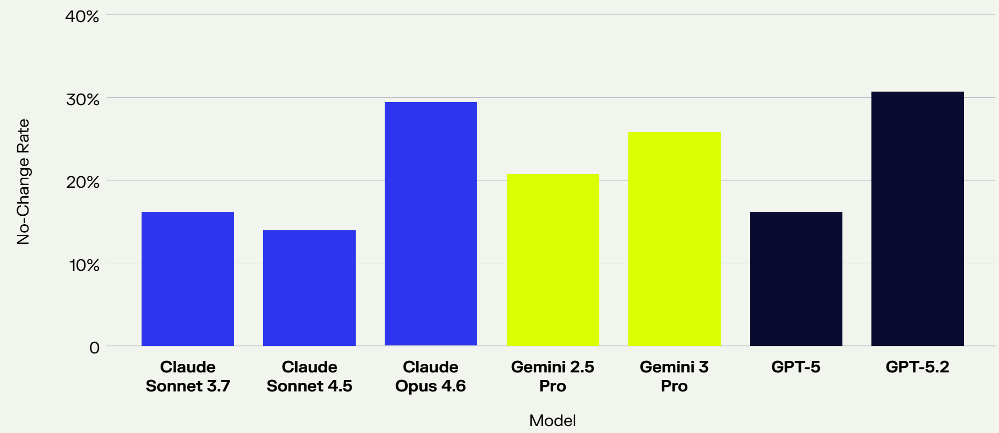
This is real progress. But even the best ungrounded model still fabricates roughly 1 in 16 recommendations, a non-trivial error rate at machine scale. However, the way this improvement was achieved introduces a new problem.

> **EVEN THE BEST UNGROUNDED MODEL STILL FABRICATES ROUGHLY 1 IN 16 RECOMMENDATIONS, A NON-TRIVIAL ERROR RATE AT MACHINE SCALE.**

AI Models Exercise Caution, But Recommend Inaction

FIGURE 2

“No-Change” Rate by Model



The newest models didn’t learn which versions actually exist.

Figure 2 explains how this happened. Models didn’t hallucinate nearly as much; instead, they simply got more cautious when uncertain.

Across providers, “no-change” recommendations, or cases where the model advises “stay on what you have,” nearly doubled over the same period that hallucination rates fell.

Where earlier models recommended “stay put” ~17% of the time, the newest generation does so closer to ~33% of the time. The reduction in fabricated versions was accompanied by a clear rise in abstention, a move toward caution rather than improved version awareness.

The pattern is consistent across Anthropic, Google, and OpenAI. Roughly one in three components now receives a “no-change” recommendation from the latest generation of models. The model avoids fabricating a version by declining to act at all.

~1 IN 3

OPEN SOURCE COMPONENTS RECEIVE A “NO-CHANGE” RECOMMENDATION

The Caution Tradeoff

Without access to live registry validation, vulnerability intelligence, or upgrade impact analysis, a model has only two options when uncertain: make a probabilistic guess or decline to act. "Do nothing" becomes the safest alternative to "make something up" and could mean nothing needs to be done, that they are uncertain whether action is required, or that some other rationale is being considered.

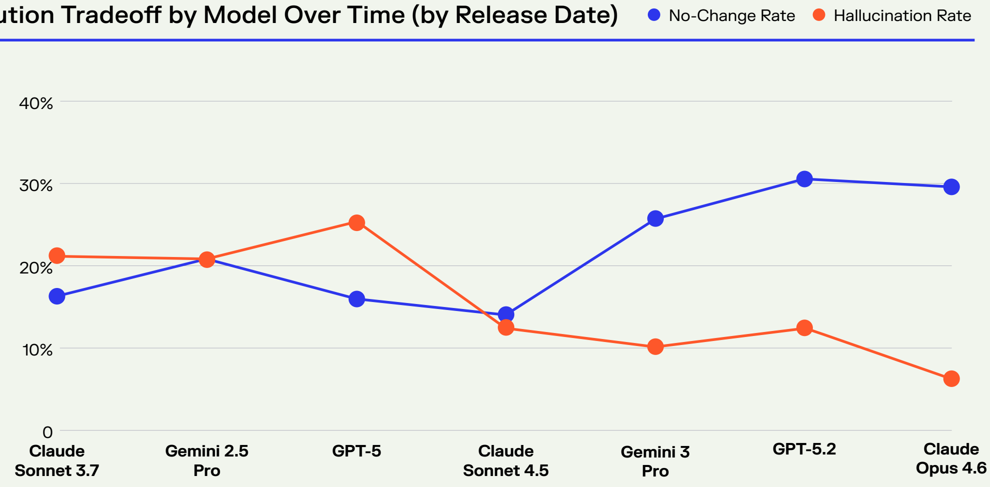
Caution is not the same as correctness. When a model recommends that developers stay on the same component, it can still lead to significant rework and time spent by engineering teams. For example, developers may have to manually validate that the advice really is the safest option, investigate alternative mitigation paths, or later revisit that dependency when a real security fix is available. Organizations are also left with technical debt and persistent risk, which remains unfixed and corrodes the software's security integrity over time.

HALLUCINATIONS CREATE PHANTOM DEPENDENCIES; INACTION PRESERVES AVOIDABLE RISK.

Neither outcome is acceptable in a production dependency pipeline. Data hallucinations introduce phantom versions that fail resolution or create unpredictable behavior. Inaction, meanwhile, quietly preserves whatever risk already exists in the system. If the current version carries critical or high vulnerabilities, a same-version recommendation locks that exposure in place.

FIGURE 3

Caution Tradeoff by Model Over Time (by Release Date)



WHY DOES AI RECOMMEND "NO CHANGE?"

The LLMs will provide a basis for their decision-making when asked. Below are a few scenarios and their associated reasoning for when the models recommended staying on the current version in this sample:

- **Deprecated with a known successor:** The component is end-of-life and has a named replacement (e.g., cx-Oracle → python-oracledb, Hystrix → Resilience4j). The model infers that upgrading within a deprecated package adds little value and defaults to the current version, though identifying deprecation from stale training data remains difficult.
- **Mature and stable:** The component has not had a newer version published in months or years. The model determines there is simply nothing to upgrade to (object-keys, Brotli, sniffio).
- **Near the LLM's training data boundary:** The component version is recent enough that the LLM is unable to confirm whether a newer patch exists. Therefore, it defaults to recommending the current version rather than guessing (jackson-annotations 2.19.0, Flask 3.1.1).
- **Niche or obscure:** The LLM doesn't have enough data about the component to make a confident recommendation, so it falls back to "stay on current" as the safest advice (IBM WebSphere SPI packages, image-ssim).
- **Rapidly evolving:** A component's version releases outpace training data. Some components ship so frequently that the LLM knows its recommendation will be stale almost immediately, so it punts (e.g., mypy-boto3-*, openai, langsmith).

In other words, AI model hallucination and inaction are not opposites. They are two sides of the same coin. Both are symptoms of reasoning without the data required to determine the correct upgrade path. The real issue is the price we pay for their restraint. What happens to vulnerability exposure when a third of components receives a "stay put" recommendation?

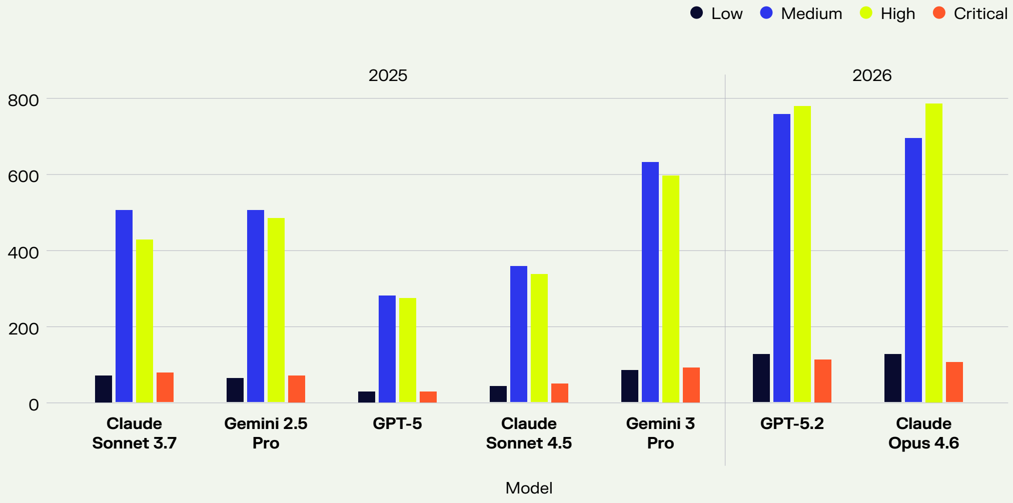
The Cost of Inaction in AI Software Development

So, what does doing nothing actually preserve? Across models, there are some vulnerabilities that simply do not have a remediation path, and most same-version recommendations (approximately 93-95% across models) do land on clean components — that matters.

But breaking the retained vulnerabilities into individual severity tiers reveals the true shape of the risk. Critical and High CVEs are not statistical outliers buried beneath a large base of Low-severity findings. They are the primary drivers of retained exposure in same-version recommendations.

FIGURE 4

Severity Breakdown of Vulnerable No-Change Recommendations



The newer, more conservative models retain more vulnerabilities in aggregate — and more of the most dangerous ones. Compared to earlier generations like GPT-5 and Claude Sonnet 3.7, the gap widens at every severity tier. The pattern is clear: cautious models retain more vulnerability exposure, not less. The three most cautious models each retain nearly double the exposure seen in earlier generations.

> **NEW MODELS RETAIN RISK IN AGGREGATE, INCLUDING THE MOST SEVERE.**

This is the hidden cost of reduced AI model hallucination. When a model defaults to “stay on what you have,” it is not making a neutral choice. It is affirmatively deciding to preserve whatever risk currently exists, including remotely exploitable flaws, privilege escalation paths, and high-impact weaknesses that regulators and attackers both prioritize. If that component carries a remotely exploitable vulnerability, the exposure remains.

Caution without intelligence doesn’t make AI software development less risky; it reshapes it. Inaction is not safety. It is deferred decision-making, and at machine scale, AI code security risks compound quickly.

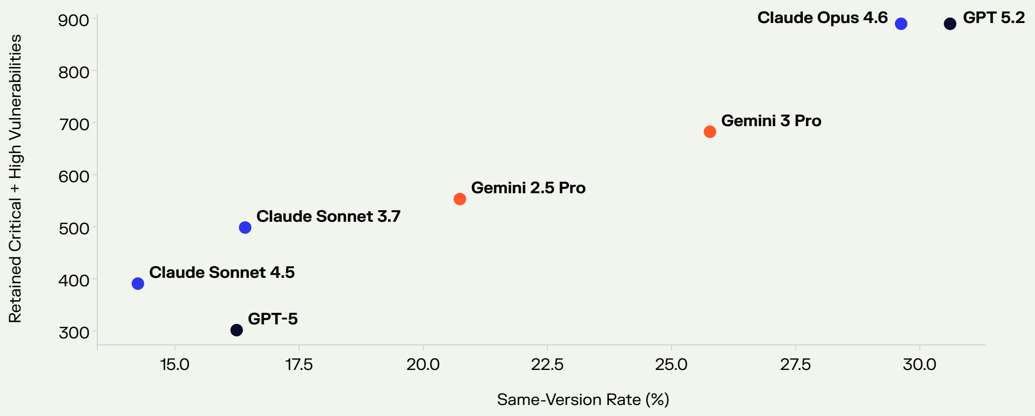
A Structural Pattern, Not a Vendor Artifact

Plotting each model's "no-change" rate against its retained Critical and High vulnerability count makes the relationship unmistakable. As "no-change" rates increase, retained High-severity risk increases with them. And this is not isolated — the correlation is provider-agnostic.

OpenAI's GPT-5.2, Anthropic's Claude Opus 4.6, and Google's Gemini 3 Pro all sit in the top-right quadrant of the chart, the region defined by the highest levels of caution and the highest levels of retained Critical and High exposure. Earlier models cluster toward the lower-left: lower "no-change" rates and lower retained High-severity counts, despite higher rates of hallucinated versions. The trajectory is consistent across vendors and model families.

FIGURE 5

Higher "No-Change" Rates Retain More High-Severity Risk



This matters because it rules out a simple explanation like "one provider tuned too conservatively." Instead, the pattern suggests something systemic about ungrounded LLMs operating in dependency management tasks. The result of this caution is a predictable tradeoff.

Reducing AI model hallucination improves headline accuracy metrics, but retained vulnerability exposure rises in parallel. Without grounding in real-time supply chain intelligence, ungrounded LLMs appear to face a ceiling: they can guess and risk fabricating versions, or they can default and preserve existing exposure. Across providers, the models are choosing the latter.

The Grounding Gap: Real-Time Intelligence Eliminates Avoidable Risk

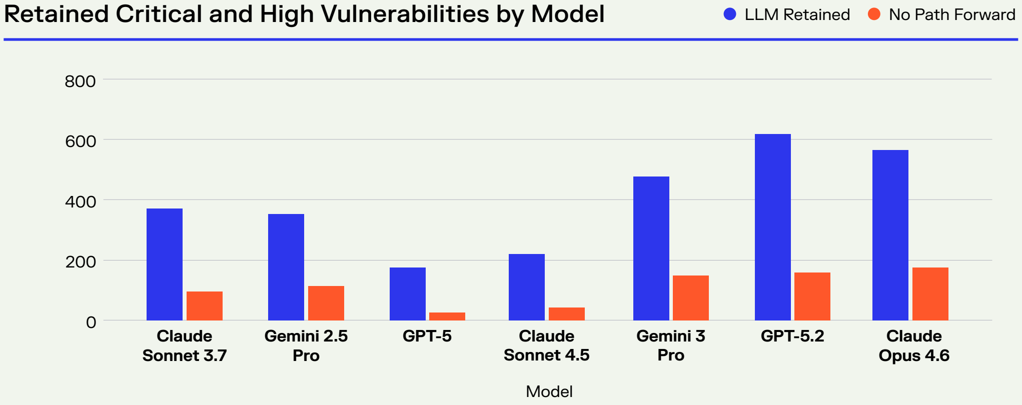
Figure 6 isolates what may be the most important distinction in this analysis: how much of the retained high-severity risk is actually unavoidable, and how much is simply the result of incomplete information.

The blue bars show the Critical and High vulnerabilities each model preserves by recommending "stay put." The orange bars show what would remain if those same component recommendations were made with real-time intelligence applied and represent No Path Forward.

The difference between the two is avoidable risk.

FIGURE 6

Retained Critical and High Vulnerabilities by Model



For the most cautious models, the numbers are stark. Across every provider and model generation, the Hybrid approach reduces retained high-severity exposure by more than half — often substantially more at 70-90%.

Importantly, the orange bars are not zero. Some components genuinely lack a clean upgrade path. In those cases, the LLMs are correct that no immediate remediation exists. But for the majority of cases, a better version does exist, the model simply didn't know.

> **THE CAUTION WE OBSERVED PRESERVES CRITICALAND HIGH-SEVERITY VULNERABILITIES.**

The caution we observed earlier is not protecting systems from unavoidable risk. It is preserving vulnerabilities that could be reduced with access to live registry data, vulnerability intelligence, and structured upgrade analysis.

When models operate without real-time supply chain intelligence, they default to inaction and retain exposure. When recommendations are anchored in data-driven version strategies, the majority of the retained Critical and High risk disappears.

That delta is the cost of ungrounded AI software development.

“Stay Put” in the Real World

The aggregate numbers tell one story. The individual CVEs tell another. When models recommend “stay on your current version,” here is what they are silently endorsing.

Many of the retained vulnerabilities are household names: Next.js, Rollup/Vite, Pillow, Kafka, foundational libraries and frameworks that power millions of production systems. These are core infrastructure dependencies in modern web stacks, data pipelines, and AI applications.

TABLE 1

Retained Vulnerabilities in Same-Version Recommendations

CVE	Package	CVSS	Impact	Models “Stay Put”	Sonatype Fix (at time of writing)
CVE-2025-55182	next (npm) 15.3–15.5	9.3	**Pre-auth RCE** in React Server Components — deserializes untrusted payloads from HTTP requests. Public PoC + active research.	4 (Sonnet 4.5, Opus 4.6, Gemini 3, GPT-5.2)	16.1.5
CVE-2025-66516	tika-core (Maven) 1.13–3.2.1	10.0	**Critical XXE** in Apache Tika — XML External Entity injection via crafted PDF. Widely used in enterprise content extraction.	4 (Son 3.7, Son 4.5, Opus 4.6, Gem 2.5)	3.2.2
CVE-2025-58367	deepdiff (PyPI) 5.0–8.6.0	10.0	**RCE via pickle deserialization —** class pollution in Delta constructor allows arbitrary code execution.	3 (Son 3.7, Opus 4.6, Gem 3)	8.6.1

The pattern is equally important. These are not single-model quirks. In many cases, four or five frontier models, spanning OpenAI, Anthropic, and Google, independently converge on the same conclusion: do nothing. The cross-provider consensus makes the failure more compelling, not less. It demonstrates that the issue is not a tuning artifact from one vendor, but a shared limitation in ungrounded generative AI software development.

And yet, for most of these components, an actionable upgrade path exists. Sonatype’s Hybrid strategy, which combines model reasoning with real-time intelligence on package health, compatibility, and risk, identifies a concrete version that reduces or eliminates the exposure. The models were not blocked by an unsolvable tradeoff. They simply did not have the data required to see the safer alternative.

The severity of the retained vulnerabilities removes any ambiguity about impact. Multiple entries carry CVSS 10.0 scores, including pre-authentication remote code execution flaws. These are not theoretical AI code security risks buried in documentation. They are publicly disclosed, actively researched, and in some cases, already exploited vulnerabilities.

When a model says “stay put” in these scenarios, it is endorsing continued exposure to known, high-impact risk, despite the existence of a viable remediation path.

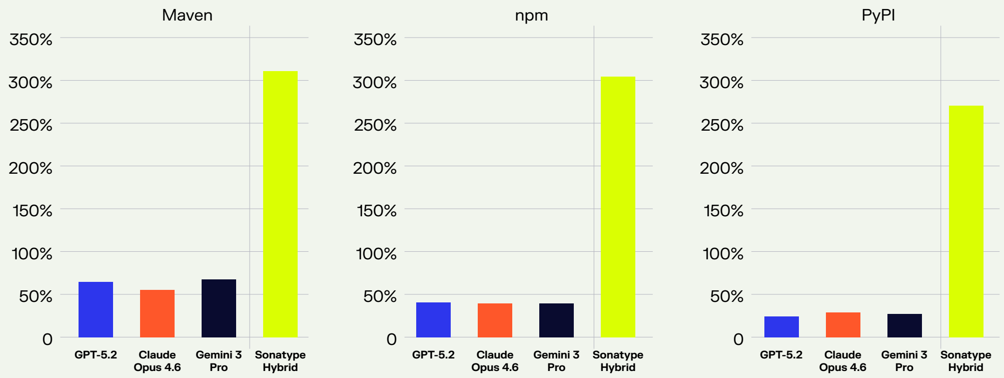
Where the Risk Concentrates: Ecosystem Differences

The top-line results in this study are directionally consistent across the full dataset, but software ecosystems are not interchangeable. Maven Central, npm, and PyPI differ in scale, release behavior, and versioning patterns, which makes it important to understand whether the same performance gaps hold when dependency recommendations are broken out by ecosystem.

The answer is yes: ecosystem differences change the shape of model failure, but not the conclusion. Figure 7 shows that Sonatype's Hybrid approach delivers materially stronger security outcomes in every ecosystem with a meaningful sample size than any LLM alone.

FIGURE 7

Main Security Score Improvement by Ecosystem & Strategy



Across Central, npm, and PyPI, Sonatype Hybrid achieves 269% to 309% mean security score improvement, compared with just 24% to 68% for the best LLM in each ecosystem. This is not a narrow lead or an ecosystem-specific advantage. It is a consistent, cross-ecosystem gap that shows ungrounded models remain structurally limited in dependency remediation.

npm remains the hardest ecosystem for ungrounded models to navigate reliably, while PyPI shows the strongest caution tradeoff and Central generally sits between those extremes. That matters because lower visible error does not always reflect better decision making.

> REGARDLESS OF ECOSYSTEM, LLM-GENERATED RECOMMENDATIONS SEE 5-10X SECURITY SCORE IMPROVEMENT WITH GROUNDING IN REAL-TIME INTELLIGENCE.

- In Central, LLMs deliver 57% to 68% mean security score improvement, while Sonatype Hybrid reaches 309%.
- In npm, where model performance is weakest, LLMs cluster around 40%, while Hybrid delivers 305%.
- And in PyPI, where models are most cautious, LLMs manage only 24% to 28%, while Hybrid still delivers 269%.

The relationship is clear: the ecosystems where ungrounded models struggle most to act safely are also the ecosystems where their security gains are weakest. Sonatype's data-driven approach, by contrast, maintains a high floor regardless of format.

The issue is that dependency decisions rely on live information about published versions, known vulnerabilities, compatibility constraints, and safe upgrade paths. Those inputs are not embedded in model weights, and they vary continuously across ecosystems. Without that real-time intelligence, even improving frontier models remain constrained. With it, security outcomes improve consistently across Maven Central, npm, and PyPI.

Cross-Model Overlap: Independent Agreement on Inaction

To test whether retained risk was a provider-specific artifact, we compared two of the most recent models from different vendors: Claude Opus 4.6 (Anthropic) and Gemini 3 Pro (Google). The question was simple: when both models recommend "stay on your current version" for a component that carries a Critical or High vulnerability, are they identifying the same components or different ones?

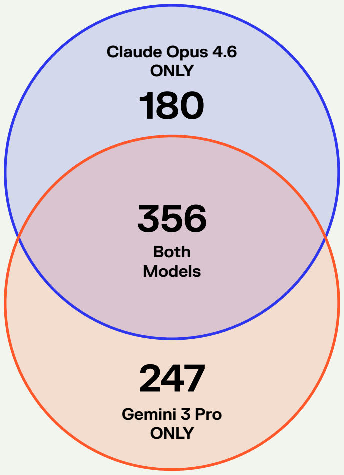
As seen in Figure 8, the majority of vulnerable same-version recommendations overlap. In 356 cases, both models independently chose inaction on the same known-vulnerable components. This shared cluster represents the largest segment in the chart.

Each model also has its own distinct blind spots. Opus 4.6 uniquely retains 247 vulnerable components that Gemini does not. Gemini 3 Pro uniquely retains 180 that Opus does not.

These differences suggest model-specific gaps in upgrade reasoning. But the dominant signal is the overlap.

FIGURE 8

Vulnerable No-Change Recommendation Overlap by Model



GROUNDING IN REAL SUPPLY CHAIN DATA IS THE MISSING VARIABLE.

The fact that two independently developed frontier models, trained by different organizations, with different architectures and tuning strategies, converge on inaction for the same vulnerable components reinforces the broader conclusion: this is not a quirk of one model or one provider.

When LLMs lack access to live version registries, vulnerability intelligence, and upgrade impact analysis, they converge on the same defensive equilibrium. They reduce hallucinations by increasing abstention, and they often abstain from the same high-risk components.

The overlap makes the implication clear: scaling model size nor switching vendors alone will not resolve the retained-risk problem.

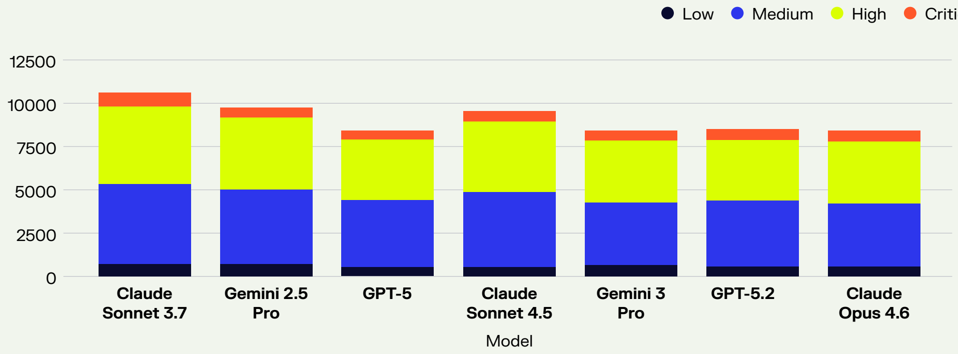
The Full Recommendation Landscape

Up to this point, we have focused on “no-change” decisions and the cost of inaction. But what happens when we apply all of a model’s recommendations, including both upgrades and “stay put” choices?

Figure 9 shows the residual vulnerability counts by severity after implementing each model’s full set of recommendations across the ~37,000-component evaluation set.

FIGURE 9

Severity Breakdown After Applying All LLM Recommendations



The picture is sobering. Even the strongest-performing models leave behind 540+ Critical and 3,500+ High vulnerabilities. Across the newest generation — Gemini 3 Pro, GPT-5.2, and Claude Opus 4.6 — results cluster tightly.

The reduction over earlier models is real, but modest. No matter which frontier model is selected, roughly 4,100-4,200 Critical and High CVEs persist. That floor is stubbornly high.

The earliest model in the set, Claude Sonnet 3.7, leaves behind roughly 5,300 Critical and High vulnerabilities. The latest, Claude Opus 4.6, reduces that to roughly 4,100, an improvement of approximately 23% across nearly a year of rapid model releases. Meaningful progress, but far from transformative.

Medium-severity vulnerabilities follow the same pattern. They decline from roughly 4,700 in the earliest generation to about 3,700 in the newest, but plateau among the most recent models.

This clustering suggests we are approaching the ceiling of what ungrounded LLMs can achieve in dependency upgrade reasoning.

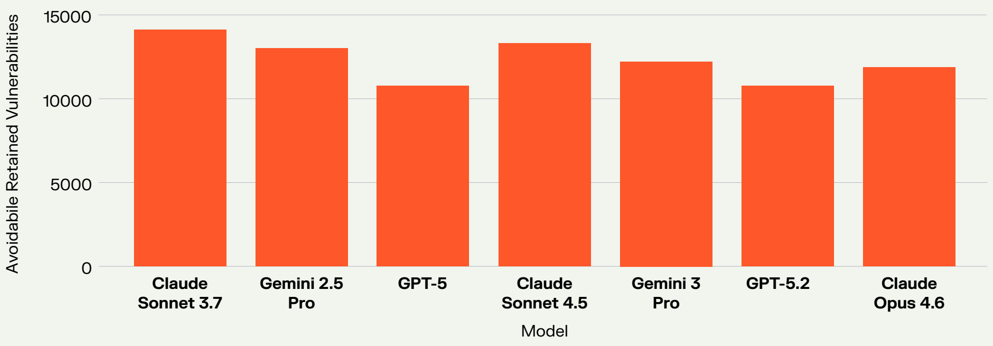
Even when models successfully avoid hallucinating and do recommend upgrades, they frequently select versions that still carry known vulnerabilities. The pattern is consistent with reasoning driven by recency heuristics, semantic versioning patterns, and naming signals, not by live security intelligence. Without access to current vulnerability data and structured upgrade strategies, the models cannot reliably identify the version that actually minimizes exposure.

All Models Benefit From Good Data

If the earlier charts isolated the cost of inaction, Figure 10 expands the lens to the entire recommendation landscape. The orange bars show the total Critical and High vulnerabilities remaining after applying each model's full set of recommendations, upgrades and stay-put decisions combined, across the ~37,000-component evaluation set.

FIGURE 10

Avoidable Vulnerabilities Across All Model Recommendations



This is purely avoidable risk. When models are grounded in real-time open source intelligence, these vulnerabilities are not recommended, suggesting that the future of AI-assisted software development will be defined less by model scale and more by access to real-time intelligence.

Across models, when real-time intelligence is applied, the LLMs recommend between 6,400 and 9,900 fewer vulnerabilities rated Critical and High per run, with the remaining vulnerabilities representing No Path Forward. That translates to a 60-70% reduction in residual High-severity risk,regardless of which LLM you start from.

- Even the strongest-performing model in the set (GPT-5) sees 59% of its remaining risk eliminated.
- The weakest (Claude Sonnet 3.7) sees 69% removed.
- The newer generation — Gemini 3 Pro, GPT-5.2, Claude Opus 4.6 — all fall within the same reduction band.

60–70%

RISK REDUCTION WHEN MODELS ARE GROUNDED IN REAL-TIME INTELLIGENCE

What makes this chart interesting is the uniformity of avoidable risk. The retained vulnerabilities vary widely, from roughly 10,830 to 14,325 remaining Critical and High vulnerabilities, reflecting genuine differences in model recency, tuning, and upgrade behavior. But the benefit of grounding with the right intelligence is stable across every model.

The ceiling we observed earlier is not a hard limit on risk reduction. It is a limit imposed by operating without real-time vulnerability intelligence, registry validation, and upgrade strategies.

Once that intelligence layer is introduced, the residual risk floor collapses regardless of the underlying model.

When Upgrade Goes Wrong

The previous sections showed the cost of inaction, cases where models recommended "stay put" and silently preserved known risk. These examples illustrate a failure in AI-assisted software development when it lacks grounding in real-time intelligence. Here, the models did not hesitate. They acted and chose wrong.

TABLE 2

Retained Vulnerabilities in Upgrade Recommendations

Package	LLM Rec	CVSS	Impact	Models	Sonatype Fix (at time of writing)
vLLM (PyPI) **0.4.0**	0.6.3 CVE-2024-9053, CVE-2024-11041, CVE-2025-30165 + 13 more	9.3	**16 Crit+High on target** (up from 5). Deserialization RCE in the AI inference engine itself — AI agents building their own vulnerable toolchain.	3 (Son 4.5, Opus 4.6, Gem 3)	0.17.0
Apache Parquet (Maven) **1.12.3**	1.13.1 CVE-2025-30065	10.0	**CVSS 10.0 deserialization RCE —** schema parsing executes arbitrary code. Parquet is the standard columnar format for every major data platform (Databricks, Snowflake, AWS Athena).	2 (GPT-5.2, Son 4.5)	1.15.2
Ray (PyPI) **2.9.2**	2.10.0 CVE-2023-48022, CVE-2025-62593 + 8 more	9.8	**10 Crit+High on target.** Remote code execution in Anyscale Ray — the distributed compute framework used by OpenAI, Uber, and Amazon for training and serving LLMs.	2 (Son 4.5, Opus 4.6)	2.53.0
Spring Framework (Maven) **5.3.31**	5.3.39 CVE-2016-1000027 + 3 more	9.8	**RCE via deserialization** in the dominant Java web framework. Models upgrade within Spring 5.3.x (EOL branch) instead of recommending the secure Spring 6.x path.	2 (Gem 3, GPT-5.2)	6.2.12

WHAT MAKES THESE DIFFERENT FROM INACTION

These are not passive “do nothing” outcomes. In each case, a frontier LLM actively selected a specific upgrade target. The version exists. The recommendation resolves. The model made a deliberate choice, and that choice embeds Critical or High-severity vulnerabilities

In multiple examples, including PySpark, Transformers, and vLLM, the recommended version contains more high-severity vulnerabilities than the user’s current version. The upgrade is not neutral. It actively makes the security posture worse. This is a reasoning-without-data pro

The AI stack itself is vulnerable. Many of the vulnerable components recommended are foundational to modern AI systems: vLLM, Transformers, Ray, PySpark. These are the libraries used to train, fine-tune, orchestrate, and serve LLMs. The irony is difficult to ignore: AI agents recommending upgrades inside the AI stack are themselves failing to avoid Critical vulnerabilities in the very tools that power them.

In each case, a safer alternative exists. Unlike inaction, where risk is preserved through caution, these examples demonstrate something more subtle and more dangerous: confident misdirection. The model provides a precise, plausible upgrade recommendation that appears to improve the system, while in reality embedding known, exploitable vulnerabilities.

Together, these cases illustrate the second ceiling of ungrounded AI: even when it acts decisively, it cannot reliably distinguish between “newer” and “safer” without access to real-time security data.

> **THE MODEL’S FAILURE IS NOT DUE TO AN UNSOLVABLE TRADEOFF, BUT DUE TO MISSING INTELLIGENCE.**

How Often Do LLMs Align With Real-Time Intelligence?

When an LLM recommends a specific version, how often does it choose the same version that Sonatype's data-driven strategies (Auto, Best, or Hybrid) would select? The answer is almost never. In AI software development environments, this divergence means probabilistic reasoning consistently underperforms data-driven upgrade strategies.

A year of rapid model advancement, including roughly 4x hallucination reduction and substantial gains in reasoning capability, moved the needle from 6% to 10%. The newest models, Claude Opus 4.6, Gemini 3 Pro, GPT-5.2, cluster tightly between 9-10%.

DATA-DRIVEN UPGRADE STRATEGIES

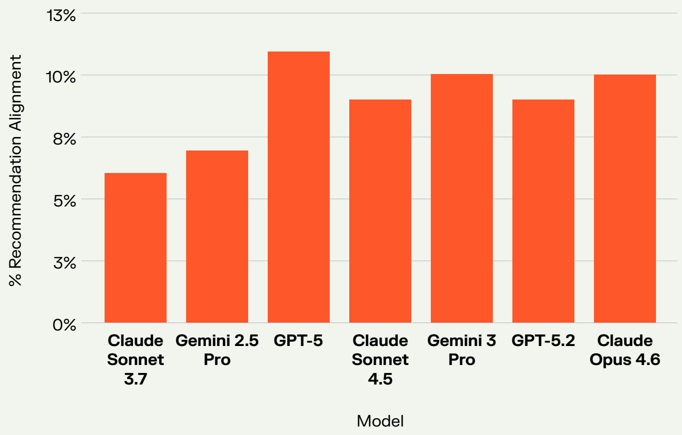
☑ **Sonatype Auto:** Chooses highest safe version without breakage; high security gains with minimal refactoring

☑ **Sonatype Best:** Chooses highest version score regardless of breaking changes; highest security improvement overall

☑ **Sonatype Hybrid:** When a version has a perfect security score, it recommends Auto; otherwise, it defaults to the Best recommendation.

FIGURE 11

How Frequently Do LLM Recommendations Align With Real-Time Intelligence?



That is the full extent of convergence. What makes this result especially striking is its consistency across providers. OpenAI, Anthropic, and Google all land within the same narrow band. No model, regardless of size, generation, or release date, exceeds 11% agreement.

LLMs are not failing to reason syntactically about version numbers. They are reasoning without the inputs required to make the reasoning meaningful. And when these models have built-in reasoning, it is with commodity data — inherently lower quality. The version that minimizes security exposure, respects license constraints, and avoids breaking changes cannot be reliably inferred from static training data. It depends on live vulnerability disclosures, real registry state, and upgrade intelligence.

That information is not embedded in model weights.

Scaling parameters, refreshing training data, or tuning prompts does not materially close the gap. The divergence between probabilistic version selection and data-driven upgrade strategy is not shrinking with model improvement.

Without grounding in real-time supply chain data, LLMs will continue to recommend versions that differ from, and often perform worse than, strategies informed by live security analysis.

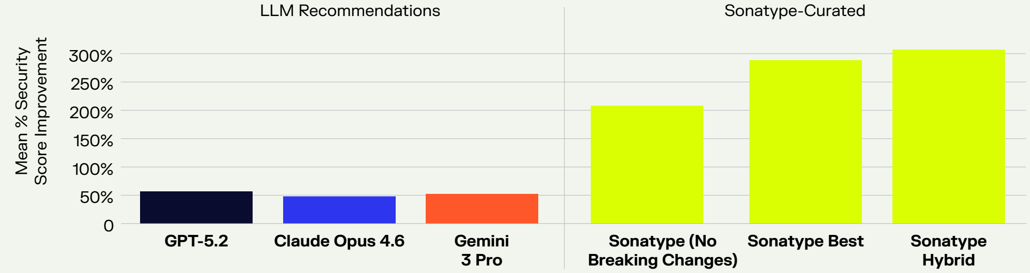
Mean Security Score Improvement Per Application

Revisiting a core metric from our prior study, Sonatype measured mean per-application security score improvement, and extends it with a larger dataset and a new generation of models. This analysis recreates the earlier methodology (originally presented in Figure 4), but with two important changes:

1. An expanded sample size: an additional ~10,000 components were incorporated into the evaluation set.
2. More conservative model behavior: as shown in earlier sections, the newest foundation models default to "stay put" more frequently when uncertain.

FIGURE 12

Mean Security Score Improvement Per Application by Strategy



Across the updated sample, frontier LLM recommendations yield only modest average security improvements per application. By contrast, Sonatype's curated strategies produce dramatically larger improvements. The magnitude gap is not incremental.

Even with a year of model advancements and substantial reductions in hallucination rates, the updated evaluation shows foundation models performing worse than what was previously reported for GPT-5 in the earlier study. The increased conservatism observed in newer models — more same-version recommendations, more abstention — directly reduces achievable security improvement.

> MODEL SOPHISTICATION ALONE DOES NOT DETERMINE SECURITY OUTCOMES IN AGENTIC SOFTWARE DEVELOPMENT.

In other words, models improved on headline hallucination metrics but regressed in net security impact. The ceiling we observed earlier appears again here. Ungrounded LLMs can produce incremental gains through heuristic version selection, but they plateau quickly. Without access to live vulnerability intelligence and upgrade analysis, they cannot consistently identify the versions that meaningfully reduce application-level risk.

Grounded strategies, by contrast, compound improvements across components, resulting in 4-6× greater average security score gains per application.

This reinforces the central conclusion of the study: model sophistication alone does not determine security outcomes in generative AI software development. Access to current, structured supply chain intelligence does.

Small Model + Sonatype Guide: The Grounding Experiment

The previous sections established that ungrounded LLMs face a structural ceiling: they hallucinate versions, default to inaction, and almost never choose the version a data-driven system would select. The gap is not closing with scale. But what if you flip the equation — give a small, cheap model access to the right data instead of throwing a larger model at the problem blind?

THE STRESS TEST

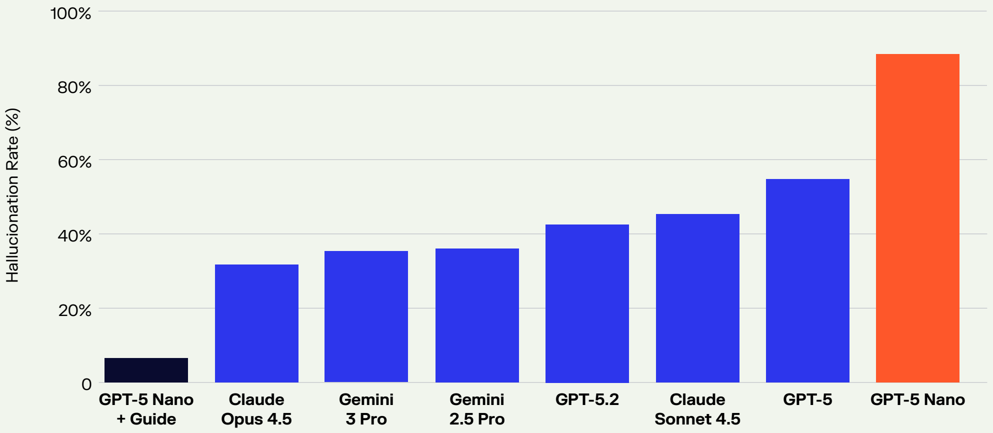
To answer this, we built a subsample of ~400 components drawn from the full evaluation set, selecting the cases where models struggle most. Including known hallucinations, components with critical or high CVEs, and packages with less standardized versioning approaches.

THE SETUP

We equipped GPT-5 Nano, OpenAI's smallest and cheapest model, with a single tool: Sonatype Guide's version recommendation API. The model, 71x cheaper than Opus, can call the tool to look up ranked upgrade candidates with vulnerability data, license analysis, and breaking-change signals. It then reasons over the tool's response to make its final recommendation. No fine-tuning, no retrieval pipeline, just a cheap model with access to one good data source.

FIGURE 13

Hallucination Rate: Small Model + Guide vs Large Ungrounded Models



On the adversarial sample, GPT-5 Nano + Guide achieves a 6.8% hallucination rate, outperforming every ungrounded frontier model, including Claude Opus 4.6 (32.0%) and Gemini 3 Pro (35.8%).

The ungrounded Nano baseline sits at 88.9%, confirming that the dataset is genuinely difficult and that the improvement comes entirely from grounding. The cost differential sharpens the contrast: at $0.14 per million blended tokens, Nano's inference costs are 32× lower than Gemini 3 Pro ($4.50/M) and 71× lower than Claude Opus 4.6 ($10.00/M), conservative comparisons given that frontier "thinking" models consume additional hidden reasoning tokens not reflected in per-token pricing. The implication is straightforward: on the components that matter most — vulnerable, ambiguous, and prone to hallucination, data access matters more than model scale.

The bottleneck was never capability, it was information. The security outcomes tell the same story. Many of these components have no clean upgrade path, every model, grounded or not, leaves behind residual vulnerabilities. But even here, the grounded model comes out ahead. Nano + Guide had 19 fewer Critical and 38 fewer High vulnerabilities than Opus 4.6, a model whose per-token inference cost is 71x higher. Grounding doesn't just prevent hallucinations; it steers the model toward versions with fewer known vulnerabilities when a perfect option doesn't exist.

The experiment confirms what the full-dataset analysis suggested: in generative AI software development, the gap between LLM recommendations and data-driven recommendations is not a model problem. It is a data problem.

Grounding AI Software Development in Live Intelligence

This expanded research reinforces our position that the limiting factor in AI-driven development isn't the reasoning capability of the model, but access to data. Models have improved with time, hallucination rates have fallen, and caution is more prevalent. But, without grounding in real-time intelligence, outcomes are still constrained, with thousands of Critical and High exposures left unresolved.

This changes when the intelligence layer is introduced. Grounded recommendations reduce retained vulnerabilities and outperform frontier models even when paired with smaller LLMs. AI doesn't make software safer on its own, but it becomes safe when grounded in the data.

Methodology

DATASET

We analyzed direct dependencies from enterprise applications scanned between June and August 2025, using the same application sample as the original study. Applications were filtered to valid scans (>10 components), deduplicated to the most operationally mature stage per app, and restricted to four ecosystems: Maven, npm, PyPI, and NuGet. The resulting component set, approximately 37,000 unique (package, version) pairs, was held constant across all seven models, yielding ~258,000 total recommendations.

Sonatype's curated strategy data (No Breaking Changes, Best, and the Security Hybrid) was refreshed from the February 2026 recommendation pipeline, ensuring vulnerability intelligence and version scores reflect the most current data. The original GPT-5 study used the November 2025 pipeline; all cross-model comparisons in this extension use the February 2026 refresh unless noted otherwise.

MODEL EVALUATION

Seven frontier LLMs spanning three providers were evaluated: GPT-5 and GPT-5.2 (OpenAI), Claude Sonnet 3.7, Claude Sonnet 4.5, and Claude Opus 4.6 (Anthropic), and Gemini 2.5 Pro and Gemini 3 Pro (Google), with release dates ranging from February 2025 to February 2026. Each model received an identical prompt: given a package identifier and current version, return a structured JSON recommendation (target version, confidence, rationale). All calls used default (often medium) reasoning effort where configurable and were processed asynchronously.

HALLUCINATION VALIDATION

Recommended versions were validated against Sonatype's package registry (the authoritative source for published versions across all four ecosystems). Any recommended version not found in the registry is classified as a hallucination. A same-version recommendation, where the model returns the current version, is classified separately as inaction rather than hallucination.

SECURITY MEASUREMENT

Vulnerability exposure uses Sonatype's enriched severity scoring (sonatype_severity), which provides broader coverage than raw NVD scores, bucketed into Critical (≥9.0), High (7.0–8.9), Medium (4.0–6.9), and Low (<4.0). Vulnerability counts are deduplicated per component-version at the advisory level before aggregation.

When measuring security outcomes across all recommendations, hallucinated versions are treated as no-ops: the component stays on its current version and retains its existing risk. This reflects the real-world outcome when a fabricated version is rejected by a package manager. Security score improvement uses a 0–100 composite score combining worst-severity with distinct vulnerability type count, computed per application and averaged across apps. Only components below a perfect baseline score are included, and a complete-case filter ensures every compared component has all strategies present.

Sonatype's Hybrid strategy, which selects the zero-vulnerability version when one exists and falls back to the highest-scored alternative otherwise, serves as the data-driven benchmark throughout.

GROUNDING EXPERIMENT

To test whether data access matters more than model scale, we built an adversarial sample of 397 components drawn from the full dataset, deliberately targeting failure modes: prior hallucinations by GPT-5 Nano (~220), components on Critical or High CVEs (~160), and components where multiple frontier models hallucinated (~160). After deduplication and ecosystem-floor balancing, 397 components remain. Absolute rates on this sample are higher than full-dataset baselines, but every model faces the identical set.

GPT-5 Nano, OpenAI's smallest model, was equipped with a single function-calling tool backed by Sonatype Guide's version recommendation API, which returns ranked upgrade candidates with vulnerability counts, breaking-change signals, and Developer Trust Scores. The model autonomously decides whether to invoke the tool, with up to 3 turns per component. Guide responses were pre-fetched for reproducibility. The same registry validation and security outcome methodology applied to the full dataset was used here. Cost comparisons use published list pricing at the observed token ratio; thinking-token overhead in reasoning models is excluded, making reported cost ratios conservative lower bounds.