A FIELD MANUAL FOR THE AGE OF THE VULNPOCALYPSE

The Security Debt Demolition Guide

Security debt is not just 'tech debt' that slows down features. It is 'risk debt' that threatens the bottom line.

VERACODE

VERACODE

BEFORE WE BEGIN:

Why This Guide Exists Right Now

In April 2026, Anthropic made a decision that should stop every security leader cold.

The company built an AI model — codenamed **Mythos Preview** — and then refused to release it. Not because it didn't work. Because it worked too well. Mythos demonstrated unprecedented vulnerability-discovery capabilities: the kind of capability that, in the wrong hands, could surface and weaponize decades of dormant security debt across the world's codebases in a matter of months. Instead of a public launch, Anthropic shared it with a limited circle of technology partners to help them shore up their defenses before the broader threat emerges.

At RSAC 2026, security professionals were already giving this moment a name: the vulnpocalypse.

The vulnpocalypse isn't a single catastrophic breach. It's a structural shift — the moment AI compresses the timeline between vulnerability existence and vulnerability exploitation from years to months, or from months to days. Chris Wysopal, a pioneer of application security, put it plainly:

"We are compressing time. Years of latent technical debt are now being surfaced in months."

If that sentence doesn't land as urgent, consider what it means in practice. Every vulnerability your teams have been deferring, deprioritizing, or simply ignoring because it seemed low-risk in a world of human-speed exploitation — that dormant flaw is about to become an active liability. The backlog isn't a backlog anymore. It's a target list.

This guide is your plan for getting ahead of that reckoning. It draws directly from Veracode's 2026 State of Software Security (SoSS) Report — an analysis of 1.6 million unique applications and 141.3 million raw findings — to give you a data-grounded, actionable framework for eliminating security debt before it eliminates you.

VERACODE

PART ONE: UNDERSTANDING THE CRISIS

The Numbers Don't Lie — They Warn

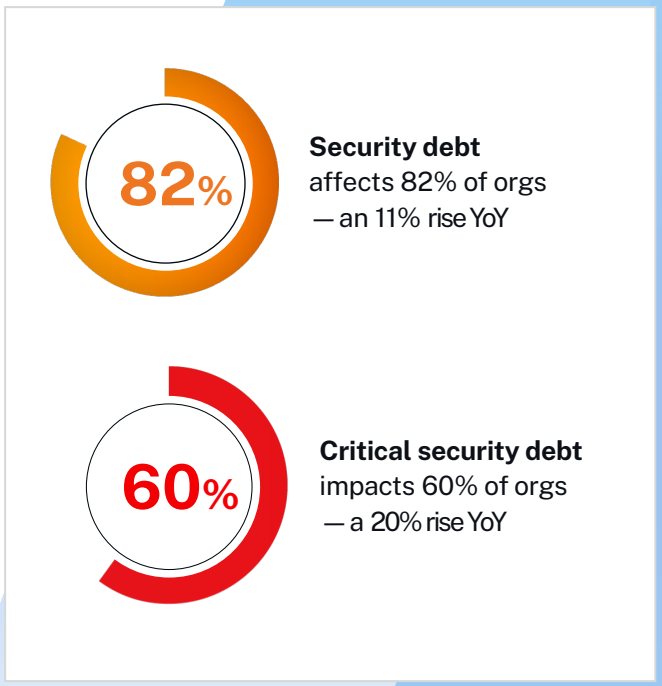
Let's start with where things actually stand, because the trajectory matters as much as the snapshot.

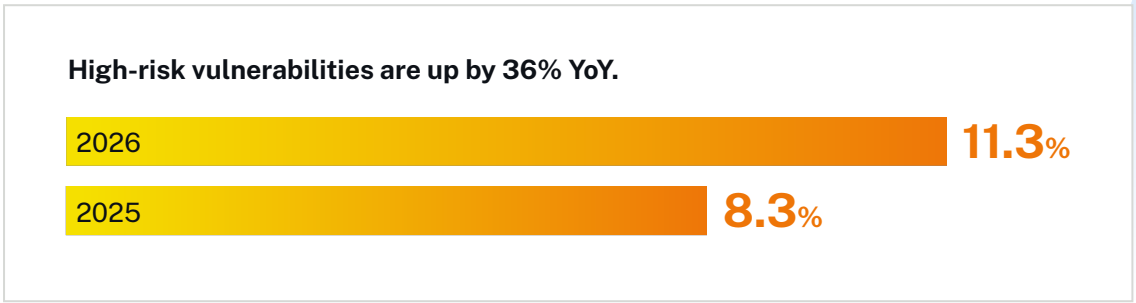
Security debt — vulnerabilities left unresolved for more than a year — is not a niche problem for underfunded startups or legacy-code shops. It is endemic across the industry. In 2024, 71% of organizations carried security debt. In 2025, that figure rose to 74%. In 2026, it has surged to **82%**. That's an 11-point jump in a single year, and the steepest year-over-year increase in the report's history.

The critical debt numbers are even more alarming. **Critical security debt** — flaws that are **both** severe in impact *and* highly exploitable — now affects **60% of organizations**, representing a stark 20% relative increase from the previous year. In 2024, that figure was 46%. In 2025, it was 50%. And now in 2026, it's 60%. The acceleration is increasing.

Look at what that means for the composition of your vulnerability population. The share of flaws sitting in the dangerous intersection of high severity *and* high exploitability grew from 8.3% to 11.3% — a **36% relative increase** in the most dangerous category of vulnerability in a single year. These are the flaws attackers specifically seek out. These are the flaws Mythos-class AI tools could find first.

Meanwhile, the percentage of applications carrying any security debt has climbed from 42% to **49%** in a single year. Nearly half of application portfolios are a ticking clock.





The Fix Rate Isn't Keeping Up

Here's the brutal mathematical reality: you cannot fix your way out of a debt problem by fixing things at the same rate they're being created. And right now, the creation rate is winning.

The average fix half-life across all scan types sits at **243 days**. That means it takes the average organization the better part of a year to remediate half of its discovered flaws. For third-party and open-source vulnerabilities found via software composition analysis (SCA), that half-life extends to **358 days** — 115 days longer than the already-too-slow overall average.

Third-party critical debt, while showing some improvement (down to 66% from 70%), still means that organizations carry the majority of critical risk in code they didn't write. Dependencies are debt by inheritance.

The OWASP Top 10 failure rate didn't improve — it increased. **50% of applications now fail OWASP Top 10 tests**, up from 48% the year prior. These aren't exotic zero-days. These are the most well-documented, well-understood vulnerability categories in the industry. Failing them in 2026 is not a skills gap. It's a prioritization and process failure.

VERACODE

Why AI Is Making This Exponentially Worse — Before It Gets Better

The promise of AI-assisted development is speed. The reality, documented by the comprehensive testing of over 150 large language models, is that **only 55% of AI-generated code passes basic security tests**. In 45% of cases, AI introduces known security flaws directly into your codebase. The code looks right. It's syntactically correct more than 95% of the time. It compiles. It ships. And it harbors vulnerabilities.

The Spring 2026 GenAI Code Security Update confirmed this isn't a temporary growing pain: "Despite the marketing hype and genuine functional improvements, nearly half of all AI-generated code contains known security vulnerabilities when no security guidance is explicitly provided." Two years of revolutionary model releases from OpenAI, Google, and Anthropic have moved the AI code security needle from approximately 55%... to approximately 55%. The velocity gains are real. The security improvements are not.

Consider a specific example: cross-site scripting (XSS) attacks. In 85% of tests, AI-generated code **failed** security tests for XSS — one of the most devastating and common vulnerability categories, capable of enabling severe data breaches. The irony is cruel: AI performs best on the well-documented vulnerabilities (82% pass rate for SQL injection, 86% for insecure cryptographic algorithms) — the vulnerabilities that are already well-understood and well-defended. It fails precisely on the nuanced, context-dependent vulnerabilities that cause the most damage.

And now those AI-generated vulnerabilities are about to be discovered — at machine speed — by the same class of AI tools that Anthropic was too responsible to release to the public.

This is the double-edged sword:

AI is simultaneously creating new vulnerability patterns at scale and offering potential solutions for remediation at scale.

The question is which edge you grab first.

The New Mandate: Software Trust

The security conversation has fundamentally changed. It used to be: Did we scan it? The new question — the one that Anthropic's Mythos moment has made unavoidable — is: *Can we trust it?*

Veracode defines this emerging imperative as **Software Trust**: the continuous ability of an organization to answer four critical questions before releasing software:

1. What risk actually matters?

Not every finding is equal. Trust begins with understanding real exposure — what is exploitable, reachable, material, and consequential.

2. Can we reduce that risk fast enough?

In an AI era, remediation cannot depend on human bottlenecks and backlog theater. Trust requires continuous, scalable risk reduction.

3. Can we govern AI safely?

As AI generates code and increasingly acts on behalf of developers, organizations need clear control over how AI enters the software lifecycle.

4. Can we prove software is safe to ship?

Not whether a policy exists. Not whether a scan ran. But whether there is verifiable evidence that the software, its dependencies, and its AI-assisted changes meet a standard of trust.

The companies that win in the AI era won't be the ones that find more issues.
They'll be the ones that can prove their software can be trusted with every release.

That is the goal this guide is designed to help you reach. Here's the framework to get there.

PART TWO:

The Security Debt Demolition Framework

The Veracode 2026 SoSS Report doesn't just diagnose the problem — it prescribes a structured response. The framework comes in three strategic phases — **Prioritize, Protect, Prove** — each with specific tactics grounded in the data. What follows is an expanded operational guide for implementing each phase.

Prioritize

Protect

Prove

VERACODE

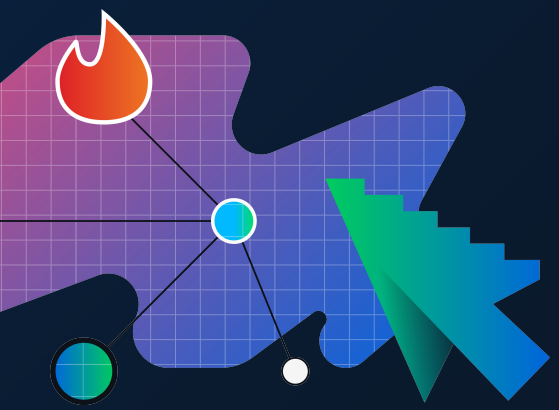
VERACODE

PHASE 1:

Prioritize — Stop Treating All Debt Equally

One of the single most dangerous assumptions in software security today is that all vulnerabilities deserve equal attention. They don't. Trying to fix everything is a recipe for fixing nothing that matters or only what's easiest to fix and not what's riskiest if you don't fix it.

The data makes the prioritization imperative crystal clear: **traditional CVSS scoring is insufficient**. A vulnerability's CVSS score tells you how bad it theoretically could be. What you actually need to know is whether it's exploitable in your environment, whether it's reachable from an attack surface, and whether exploitation would cause material consequences to your business.



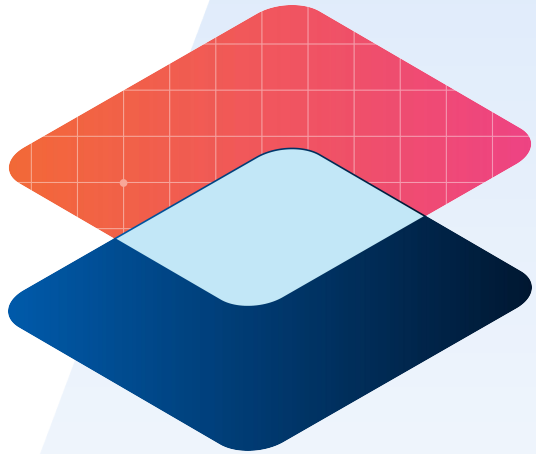
Triage by the danger zone, not the backlog.

The 36% increase in flaws sitting in the "high severity + high exploitability" intersection defines your emergency ward. These are the vulnerabilities that nation-state actors, ransomware gangs, and AI-powered attack tools will target first. Your team should target them first, too. Implement a formal Emergency Triage Protocol that flags any flaw meeting both criteria — high exploitability AND high severity — for immediate escalation and remediation, bypassing normal queue processes entirely.

VERACODE

Use ASPM to achieve unified visibility.

Most organizations are scanning with multiple tools — SAST, DAST, SCA, container scanning — and seeing fragmented, context-free results. Application Security Posture Management (ASPM) solutions can correlate findings across all these tools and layer in runtime context and business priority. A critical vulnerability in a test environment is fundamentally different from the same vulnerability in a public-facing payment system. ASPM makes that distinction visible and actionable.



Map your assets before you map your vulnerabilities.

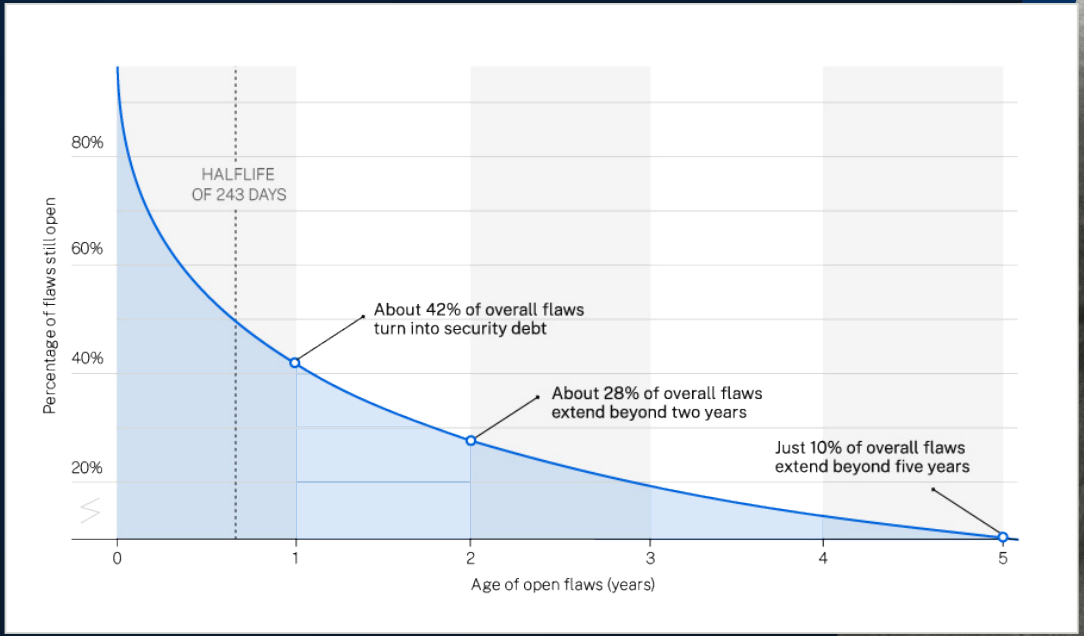
You can't prioritize what you can't see. The first step is a complete, current inventory of your application portfolio — understanding which applications are public-facing, which handle sensitive data, which are customer-critical, and which are internal-only. This asset map becomes the foundation for every prioritization decision. High-risk vulnerabilities in high-value, high-exposure applications are your non-negotiable starting point.

Classify debt, don't just count it.

Not all security debt is equal either. There's a meaningful difference between a medium-severity information disclosure flaw that's been open for 14 months in a low-traffic admin tool, and a high-exploitability injection vulnerability in your customer-facing API that's been open for 13 months. Both qualify as "security debt" by the one-year definition. Only one belongs on your emergency list. Build a debt classification system that segments your portfolio into: Critical Debt (high severity + high exploitability), Elevated Debt (high severity or high exploitability, not both), and Managed Debt (lower risk, longer-term remediation track).

Set concrete fix-half-life targets.

The current industry average of 243 days to fix half of known vulnerabilities is far too slow for the world Mythos (and other tools like it) is about to create. The SoSS Report recommends a target of **fewer than 90 days for the fix half-life of critical vulnerabilities**. That's a more than 60% improvement from the current baseline for the most dangerous class of flaws. Set this as an organizational KPI and make it visible at the leadership level. Keep in mind that some regulations mandate an even shorter timeline, so check if they apply to you.

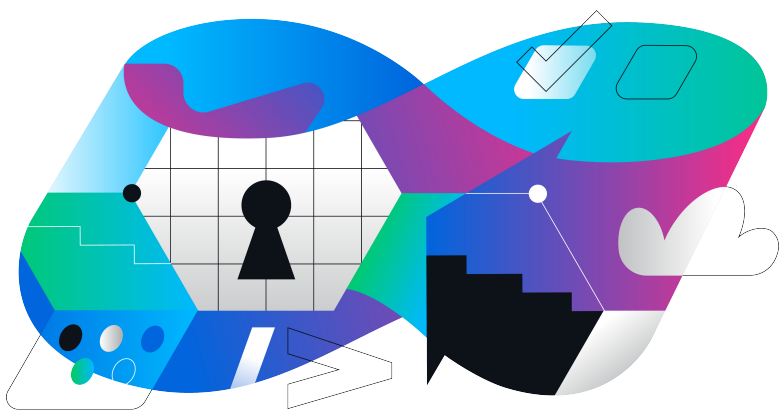


VERACODE

PHASE 2:

Protect — Build Continuous Defense Into the Pipeline

Prioritization tells you what to fix. Protection determines how you systematically prevent the problem from compounding faster than you can address it. The era of periodic security scans and annual pen tests is definitively over. Protection in 2026 and beyond requires continuous, automated defense integrated directly into the development lifecycle.



Fix before you close.

One of the highest-leverage policy changes any organization can make costs nothing but discipline: implement a **"fix before close" policy** for high-risk vulnerabilities in new code. Any ticket, story, or PR that introduces or reveals a high-risk vulnerability doesn't get closed until the vulnerability is resolved. This single practice prevents the most pernicious form of debt creation — the deliberate deferral of known-at-creation vulnerabilities into the backlog.

Integrate security directly into developer workflows.

Security tools that exist only in security team dashboards are security tools that developers work around. Integrate automated scanning and fix suggestions directly into IDEs, so developers see vulnerability flags and remediation recommendations in the same environment where they write code. The goal is to make secure coding the path of least resistance, not a separate workflow requiring context-switching.

VERACODE

Automate remediation of the "long tail."

Not every vulnerability requires a human expert to fix. A significant portion of your vulnerability backlog consists of well-understood, repeatable patterns — the same classes of injection issues, the same insecure configurations, the same outdated cryptographic implementations — that AI-powered remediation tools can address with high accuracy. Deploy these tools for the long tail of simple, repetitive vulnerabilities, and **allocate 10–15% of sprint capacity specifically to security debt reduction** using AI-generated fixes reviewed by humans. This creates a systematic, sustainable cadence for debt reduction rather than leaving it to heroic individual efforts.

Use a package manager firewall.

For the 66% of organizations carrying critical third-party debt, dependency management is not a supplementary concern — it is a core security function. Implement package manager firewalls that prevent vulnerable dependencies from entering the codebase at all. Pair this with a formal dependency review process that evaluates security risk alongside functional fit and license compatibility. The 358-day fix half-life for third-party vulnerabilities is a direct consequence of organizations discovering third-party risk late and remediating it even later. Stop the inbound flow of vulnerable dependencies before they become part of your debt.

Govern AI code generation explicitly.

Given that 45% of AI-generated code introduces known security vulnerabilities, and given the specific, documented failure of AI tools on XSS vulnerabilities (86% failure rate), organizations cannot treat AI-generated code as equivalent to human-reviewed code for security purposes. Build explicit governance around AI in the SDLC: require security scanning of all AI-generated code before merge, provide AI coding tools with security-focused system prompts and guardrails, and implement security training for developers specifically focused on c**ommon vulnerability patterns in AI-generated code**. The sophistication of your AI governance will directly determine your vulnerability exposure trajectory.

Instrument your CI/CD pipeline end-to-end.

Continuous integration shouldn't just mean continuous build and test — it should mean continuous security validation. Organizations that integrate security scanning across the CI/CD pipeline with automated gates that prevent high-risk vulnerabilities from advancing to production are systematically protected from the most dangerous forms of debt accumulation. Leverage tools that integrate seamlessly into CI/CD pipelines to automate vulnerability scanning, dependency checks, and fix validation. Make security a quality gate, not a post-gate audit.

VERACODE

PHASE 3:

Prove — Make Security Outcomes Visible and Measurable

The third phase of the framework is where many organizations fail to complete the loop. They prioritize, they protect — and then they fail to measure what changed, fail to demonstrate improvement to leadership, and fail to sustain the organizational commitment that security debt reduction requires. Proving isn't just about compliance or reporting. It's about accountability, visibility, and the cultural change that makes security improvements durable.

Make security debt a board-level KPI.

The 2026 SoSS Report is explicit on this point: security debt must become a **board-level key performance indicator** alongside technical debt and SRE metrics. This isn't merely cosmetic escalation. When security debt lives only in security team dashboards, it competes for developer time against features, and features usually win. When security debt is visible to the board with the same weight as revenue metrics and operational uptime, the organizational prioritization changes. Leadership must own this metric, not just sponsor it.

Tie security outcomes to developer OKRs.

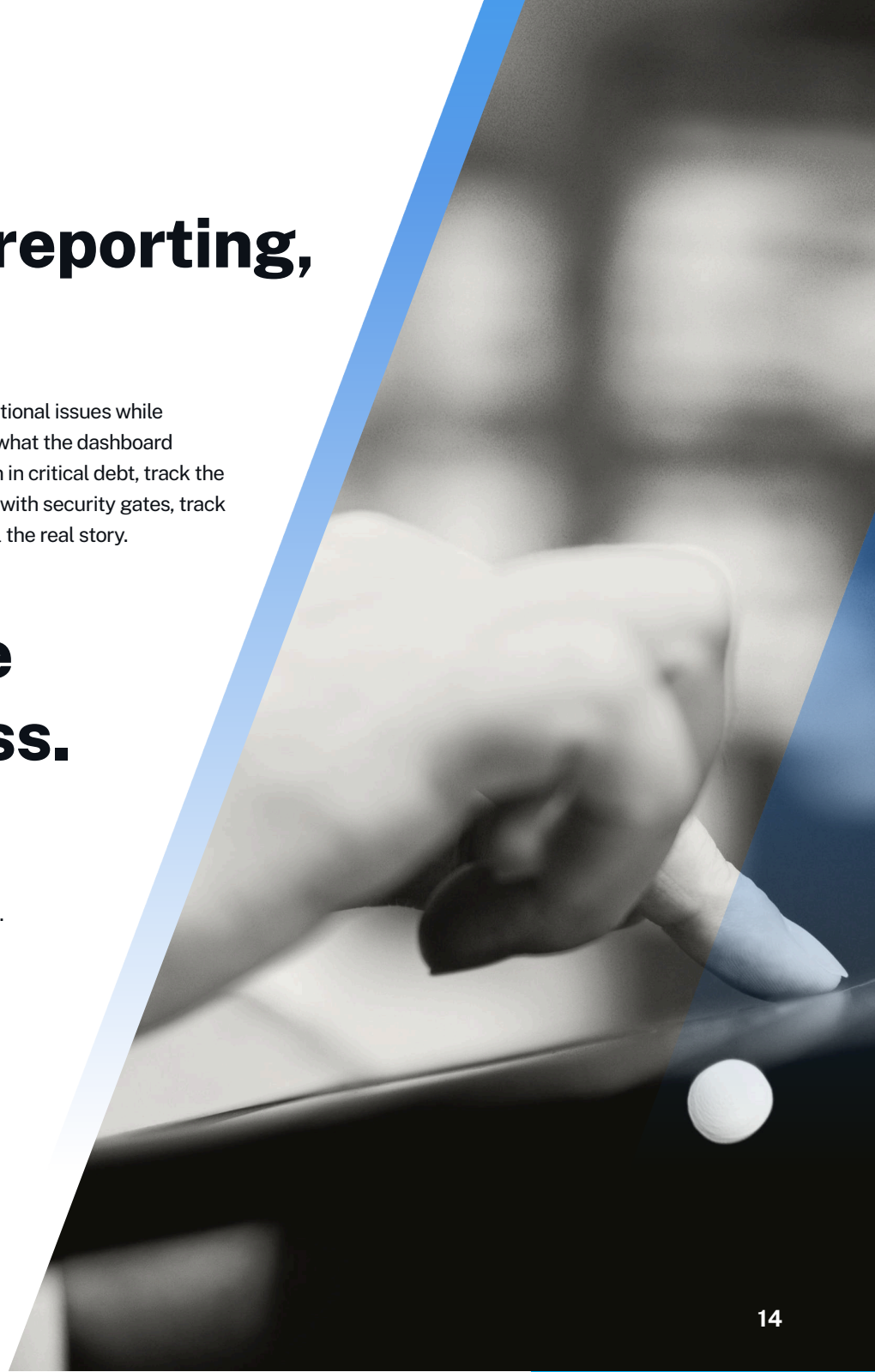
The most durable way to change security behavior is to make security outcomes part of how engineering teams are measured and rewarded. Tie security debt reduction targets directly to development team Objectives and Key Results (OKRs) and performance reviews. **Dedicate 15–20% of sprint points to security debt reduction** for teams carrying elevated debt loads. When developers and engineering leaders are accountable for security outcomes, security debt reduction becomes a team priority rather than a security team request.

Implement risk-reduction reporting, not flaw-count reporting.

Flaw counts are a misleading measure of progress. A team that closes 500 low-severity informational issues while neglecting 10 critical exploitable vulnerabilities has moved in the wrong direction regardless of what the dashboard shows. Replace flaw-count metrics with **risk-weighted remediation metrics**: track the reduction in critical debt, track the improvement in fix half-life for high-risk vulnerabilities, track the percentage of CI/CD pipelines with security gates, track the ratio of security debt in AI-generated code versus developer-written code. These metrics tell the real story.

Build audit-ready evidence of software trustworthiness.

The fourth question of Software Trust — *Can we prove software is safe to ship?* — has regulatory, commercial, and operational dimensions. Customers and partners increasingly require demonstrable evidence of security posture as a condition of doing business. Regulators are moving toward requiring verifiable software bills of materials (SBOMs) and security attestations. Build the processes now to generate, maintain, and present evidence that your software, its dependencies, and any AI-assisted changes have been validated before release. This isn't future-state compliance theater — it's competitive differentiation in a market where software trust is becoming a purchase criterion.



Benchmark against the SoSS data — and set aggressive targets.

The 2026 SoSS Report gives you the industry baselines you need to calibrate your performance honestly.

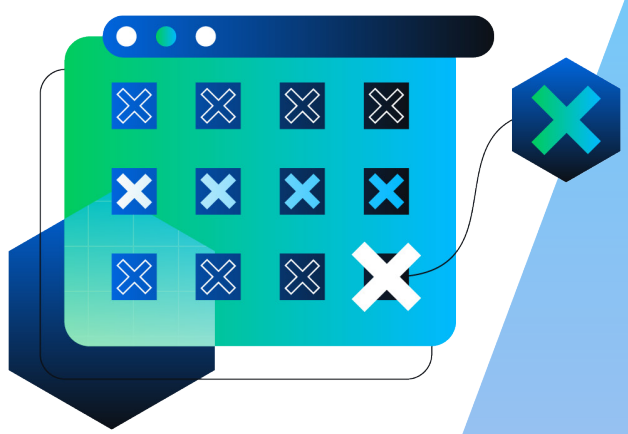
Your targets for the next 12 months should be:

- ☑ **Fix half-life for critical vulnerabilities:**
  Target under 90 days (30 days for some regulated industries), versus the current industry average trajectory.
- ☑ **Critical security debt prevalence:**
  Drive below the 60% industry average.
- ☑ **OWASP Top 10 failure rate:**
  Achieve below 40%, versus the current industry-wide 50% failure rate.
- ☑ **Third-party critical debt:**
  Target below 50%, versus the current industry average of 66%.
- ☑ **Sprint capacity for security debt:**
  Establish and maintain 15–20% allocation.

These aren't aspirational moonshots. They are achievable targets that, if met, would put your organization in the leading quartile of security posture — the position you need to occupy as Mythos-class capabilities become broadly available.

PART THREE:

The 90-Day Demolition Plan



Strategy without execution is just documentation. Here is a concrete 90-day implementation plan drawn from the SoSS Report's recommended action sequence, organized for security leaders who need to move from this guide to measurable progress.

Days 1–30: Emergency Triage and Inventory

The first 30 days are about getting an honest, complete picture of where you stand and immediately addressing the highest-risk exposure in your portfolio.

Begin with a full asset inventory. Map every application in your portfolio — internal, external, customer-facing, partner-integrated — and establish its risk tier based on exposure and data sensitivity. This inventory is the foundation of every subsequent prioritization decision. Then run a full-portfolio security scan across all scan types (SAST, DAST, SCA) and classify every open vulnerability against the two-dimensional risk matrix: severity on one axis, exploitability on the other. Every flaw in the high-severity + high-exploitability intersection enters your Emergency Triage Protocol immediately. No queue. No deferral. These are the vulnerabilities that AI-powered discovery tools could find and weaponize first.

In parallel, audit your third-party dependencies. Given the 358-day average fix time for third-party vulnerabilities, your open-source debt is almost certainly older and more exposure-dense than your first-party debt. Identify every open critical and high vulnerability in your dependency tree and begin prioritizing them alongside your first-party findings.

Begin the governance process for board-level KPIs. Security debt belongs on the board agenda, and 30 days is enough time to prepare an executive briefing with the SoSS data as context and your own organization's current state as the specific action call.

Days 31–60: Pipeline Integration and Policy Implementation

The second 30 days are about changing the process so that debt stops accumulating at the current rate.

Deploy IDE-integrated security tooling across your development teams. Make vulnerability detection and fix suggestions available at the point of code creation. Implement "fix before close" policies for all high-risk vulnerabilities and configure CI/CD pipeline security gates that prevent high-risk findings from advancing to production. These two changes — a policy and an automation — will have an immediate and measurable effect on new debt creation.

Stand up your package manager firewall and implement a formal dependency review process. From day 60 forward, no vulnerable dependency should enter your codebase without explicit security review and tracking.

Launch your AI code governance program. Define and publish internal standards for AI-generated code: what tools are approved, what security scanning is mandatory before merge, and what training developers need on AI-specific vulnerability patterns. This is not an optional side project given the 45% vulnerability introduction rate of current AI coding tools.

Begin tracking sprint-level security debt reduction metrics. The first metric to baseline: what percentage of each team's sprint capacity is currently going to security work? Use this as your starting point for the 15–20% allocation targets.

Days 61–90: Measurement, Accountability, and Momentum

The final 30 days of the initial cycle are about institutionalizing the change — making sure the improvements in the first 60 days don't decay when the urgency of launch fades.

Formalize security OKRs for engineering teams. Define the specific security outcomes each team is responsible for over the next two quarters, tie them to performance conversations, and make them visible in the same systems where other OKRs live. Present the first SoSS-benchmarked security posture report to the board or senior leadership. Show the baseline you started from, the actions taken in 90 days, and the trajectory you're committed to — calibrated against the 2026 SoSS industry data.

Build the process for continuous software trust evidence. Define what a "safe to ship" evidence package looks like for your organization — what scans must have run, what gates must have passed, what critical flaws must have been resolved — and begin generating that package for each major release.

At day 90, you should have: a classified, prioritized vulnerability portfolio; pipeline security gates actively preventing new high-risk debt from forming; a formal AI code governance policy; sprint-level security debt reduction in progress; board-level visibility; and a measurable baseline to compare against six months from now.

VERACODE

PART FOUR:

The Threat is Real — But So Is the Opportunity

It would be easy to read all of this — the vulnpocalypse framing, the Mythos story, the accumulating debt statistics — and conclude that the situation is hopeless. It isn't.

The same AI capabilities that Anthropic was cautious enough to withhold from the public are available to defenders too. The same tools that can surface decades of dormant vulnerabilities at machine speed can be directed at your own codebase first — on your schedule, with your security team in control. Organizations that use this window to systematically reduce their security debt will find themselves in a fundamentally different risk position than the organizations that wait.

The 2026 SoSS data shows two populations: organizations where security debt is actively compounding and critical risk is escalating, and organizations where consistent investment in prioritization, pipeline integration, and measurable accountability is beginning to bend the curve. The difference between those two populations is not resources. It's decisions — about prioritization, about process, and about how seriously leadership treats security debt as a business risk rather than a technical inconvenience.

You can't fix what you ignore. But you can fix what you prioritize. The framework is here. The data is clear. The clock is running.

Start demolishing.

APPENDIX:

Quick Reference — Key Metrics from the 2026 SoSS Report

Metric	2024	2025	2026	Trend
Organizations with security debt	71%	74%	82%	↑ Worsening
Organizations with critical security debt	46%	50%	60%	↑ Worsening
Applications carrying security debt	—	42%	49%	↑ Worsening
High-risk vulnerability share (sev. + exploit.)	—	8.3%	11.3%	↑ +36% YoY
Fix half-life (all scan types)	—	252 days	243 days	→ Marginal improvement
Fix half-life (SCA / third-party)	—	—	358 days	↑ Lagging
Third-party critical debt prevalence	—	70%	66%	↓ Small improvement
OWASP Top 10 failure rate	—	48%	50%	↑ Worsening
AI-generated code security pass rate	—	~55%	~55%	→ No improvement
Overall flaw prevalence in apps	—	80%	78%	↓ Slight improvement

APPENDIX:

Your Security Debt Demolition Checklist

Phase 1: Prioritize

- Complete a full application portfolio asset inventory with risk tiering
- Run comprehensive scans across SAST, DAST, and SCA
- Classify all findings against the severity × exploitability matrix
- Implement Emergency Triage Protocol for all critical-debt flaws
- Deploy ASPM to correlate findings and add runtime/business context
- Audit all third-party dependencies for open critical and high vulnerabilities
- Set organizational fix-half-life target: under 90 days for critical vulnerabilities (or 30 days for some regulated industries)

Phase 2: Protect

- Implement "fix before close" policy for high-risk vulnerabilities
- Integrate IDE security tooling with automated fix suggestions
- Configure CI/CD pipeline security gates for high-risk flaws
- Install package manager firewall and dependency review process
- Define and publish AI code governance standards
- Establish 15–20% sprint capacity allocation for security debt reduction
- Train developers on AI-specific vulnerability patterns (XSS, injection, etc.)

Phase 3: Prove

- Establish security debt as a board-level KPI
- Tie security outcomes to engineering team OKRs and performance reviews
- Shift reporting to risk-weighted metrics, not flaw counts
- Build "safe to ship" evidence generation process for releases
- Baseline current posture against 2026 SoSS benchmarks
- Present first security posture report to senior leadership within 90 days
- Schedule quarterly SoSS-benchmarked progress reviews

VERACODE

Questions? Contact Us:

Contact our team at www.veracode.com
to schedule a demo or request additional information.

This guide is based on data from Veracode's 2026 State of Software Security Report, which analyzed 1.6 million unique applications and 141.3 million raw security findings.