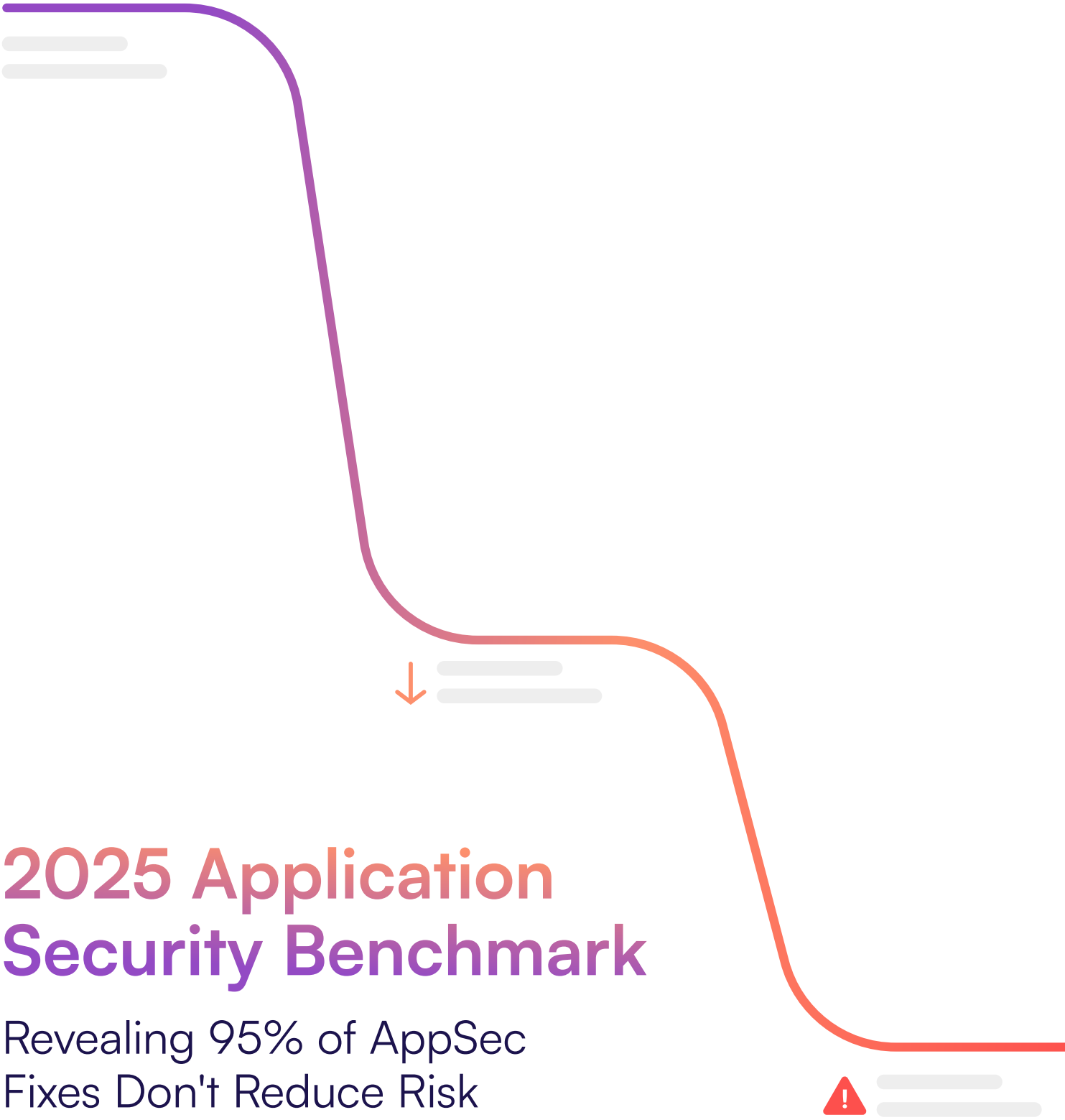


2025 Application Security Benchmark

Revealing 95% of AppSec Fixes Don't Reduce Risk

Presented By ox security

March 2025

Table of Contents

Executive Summary: The Problem of Humans Cleaning Up After Machines

The OX 2025 Application Security Benchmark Report highlights a critical challenge in application security: the increasing attack landscape combined with automated and AI-enhanced security tools generates an overwhelming volume of alerts that the security and development labor force simply cannot manage.

OX's analysis of over 101 million security findings across **178 organizations** reveals the extent of the well-known "alert fatigue" problem. Our data shows only **2-5%** of alerts require immediate action, while the remaining **95+%** are only informational. This huge gap causes the exhaustion of resources and deepens the friction between development teams and security teams sent to remediate non-critical issues. It also illustrates the gap between what security leaders are trying to achieve — an improved application security posture for organizations — and the actual result: an ever-growing burden of manually analyzing a mountain of automatically generated alerts.

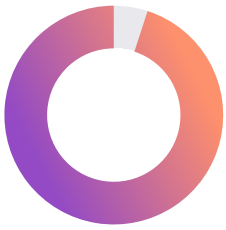
Key Findings

On average, an organization faces **569,354** security alerts, which can be **reduced to 11,836** through context-based prioritization.

Critical issues are a mere 202.

+95% of Application Security Alerts can be safely deprioritized.

- **32%** of all issues have low exploit risk
- **25%** of all alerts lack public exploits
- **25%** of all findings relate to indirect or development dependencies



+95%

of Application **Security Alerts** can be safely deprioritized.

32% of all issues have **low exploit risk**

25% of all alerts **lack public exploits**

25% of all findings relate to **indirect** or **development dependencies**

Only 2-5% of alerts are critical and require action:

- 1.71% are Known & Exploited Vulnerabilities (KEV), representing actively exploited vulnerabilities that can be exploited in the specific customer environment and demand immediate attention.
- 1.62% represent Exposure of secrets, making it one of the most prevalent high-risk security findings.

Only **2-5%** of alerts are **critical** and require action:

Our industry benchmark reveals **consistent noise thresholds of approximately 98%** across different sectors and company sizes.

Financial institutions experience distinctively higher alert volumes.

Key Takeaways

1

Shift from Vulnerability Management to Risk Management:
Focus on high-impact vulnerabilities, implement automated risk prioritization, and integrate business context.

2

Build an End-to-End Application Security Pipeline:
Create an integrated security approach, connecting security tools across the entire code-to-cloud lifecycle instead of relying on isolated scans. Adopt a context-aware model that considers your organization's specific threat landscape and correlates security data across SDLC.

3

Assist Developers Instead of Overwhelming Them:
Shift security left to reduce hourly investment, provide actionable intelligence

4

Foster Cross-Team Collaboration:
Align metrics and objectives

Security Profile of the Average Organization

Active Issues

569,354
Before Context-Based Prioritization

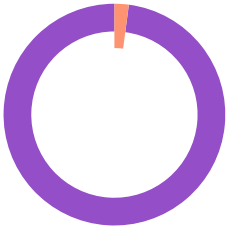
↓ **11,863**
After Context-Based Prioritization

Critical Issues

6,023
Before Context-Based Prioritization

⚠ **202**
After Context-Based Prioritization

2.08% Real Risk



97.92% Noise Rate

469 Active Apps

Introduction: When Security Tools Undermine Their Own Purpose

The digital landscape is evolving at an unprecedented pace. In parallel, the complexity and volume of application security threats continue to grow exponentially. This presents a pressing management problem for businesses that build and depend on applications for effective and efficient operations. While advancements in security tooling promise more sophisticated detection capabilities, they also generate an overwhelming flood of alerts, leaving security teams struggling to distinguish genuine threats from noise. The sheer volume of identified alerts has created an industry-wide crisis, as highlighted in January 2025 Gartner report, **"Innovation Insight: Application Security Posture Management"*:**

"AppSec tools consistently generate extensive data on potential vulnerabilities. Traditional, frequently manual approaches for assessing and prioritizing these findings have failed. They cannot scale to handle the vast amount of data, which has grown exponentially with the introduction of new tests that generate even more findings, nor can they keep pace with modern development processes".

This "paradox of progress" is a central challenge addressed in the Benchmark 2025 report. Our research reveals a stark reality: the vast majority of security alerts (95-98%) do not require action by AppSec or software development teams. This finding defies the prevailing "detect everything" approach common in cybersecurity circles and calls for organizations to fundamentally shift their thinking, processes, and tool deployments toward alert management processes that highlight quality over quantity.

> Our industry benchmark reveals **consistent noise thresholds of approximately 98%** across different sectors and company sizes.

Our analysis reveals that among the critical **2-5%**, Known & Exploited Vulnerabilities (KEV) make up the largest portion - **1.71%** of all issues detected. These vulnerabilities are actively being exploited in the wild and demand immediate attention. Additionally, secrets exposure, including credentials embedded in code, represents a significant 1.62% of all issues and poses a severe security risk.

The industry benchmark included in this report demonstrates that the challenge is consistent across different sectors and organization sizes,

*Gartner, Innovation Insight: Application Security Posture Management, Giles Williams, Aaron Lord, Dionisio Zumerle, 9 January 2025 GARTNER is a registered trademark and service mark of Gartner, Inc. and/or its affiliates in the U.S. and internationally and is used herein with permission. All rights reserved. The Gartner content described herein (the "Gartner Content") represents research opinion or viewpoints published, as part of a syndicated subscription service, by Gartner, Inc. ("Gartner"), and is not a representation of fact. Gartner Content speaks as of its original publication date (and not as of the date of this Annual Report), and the opinions expressed in the Gartner Content are subject to change without notice.

with non-critical alerts reaching approximately **98%** regardless of industry or company scale. However, enterprise security environments and financial institutions face greater complexity due to their broader tool ecosystems, larger application footprints, and higher value to attackers.

Currently, organizations lack comprehensive, real-world data to assess the security state of their own software development, particularly in understanding how critical vulnerabilities manifest in practice.

This inaugural benchmark report, developed by the OX Security Research team, addresses this gap by analyzing real-world data to identify application security trends.

This report will serve as a foundational resource and be updated periodically to monitor industry progress in SDLC security. Future editions will build upon this foundation, identifying emerging trends, tracking year-over-year changes, and providing increasingly sophisticated analysis of the application security landscape.

Methodology: A Data-Driven Approach

The Benchmark Report is based on an analysis of 101 million+ application security findings collected from 178 organizations over 90 days (Q4 2024). To achieve comprehensive and accurate results, the OX research team employed a three-step methodology:

1

Data Collection and Consolidation:
We aggregated and consolidated security findings from various sources, including organizations' existing security tools (SAST, secrets detection, SCA, etc.), open-source intelligence feeds, and vulnerability databases.

2

Data Enrichment:
We enriched the collected data by generating new data points and contextual information for each event. This included mapping cloud assets back to their code origin using OX's proprietary technology to analyze dependencies between components and assess the business criticality of affected systems.

3

Context Analysis and Prioritization:
We applied evidence-based prioritization methodology, combining multiple risk labels per issue to assess real-world impact and actual risk level. This involved incorporating factors such as exploitability, reachability, business impact, and environmental context to categorize and prioritize each finding accurately.

Study Parameters

Scope:
101,344,969
101 M+
AppSec findings

Coverage:
178
Organizations

Duration:
90
Days

The Growing Threat Landscape

The reported number of application security vulnerabilities has skyrocketed in recent years. The public Database CVEdetail identified 6,494 vulnerabilities in 2015, as opposed to 40,291 in 2024 — bringing the total number of known vulnerabilities to around 200,000. This trend shows no signs of slowing down. The Forum of Incident Response and Security Teams (FIRST) projects an additional 41,000-50,000 new weaknesses will emerge in 2025.

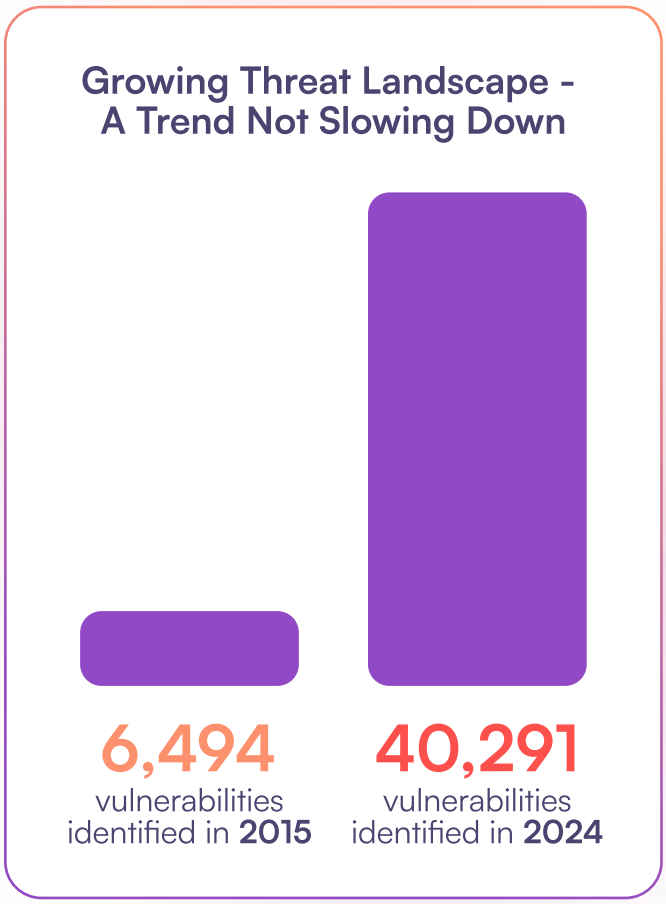
This growth of vulnerabilities, combined with the pressure to deploy software quickly, has strained security teams. While many commercial security tools excel at detecting issues, they often produce an overwhelming number of alerts, resulting in "alert fatigue." Further, when these abundant alerts are not assessed for criticality based on the specific organization's digital environment(s) and implemented security controls, security teams — or the developers charged with fixing software issues — cannot appropriately prioritize which issues to tackle first.

Chasing non-critical issues and wading through endless alerts desensitizes security and development professionals. This often results in an increased likelihood of missing genuine threats and putting the organization at greater risk than need be.

Citing again from Gartner Innovation Insight report:

> ***"If the number of findings becomes too large, organizations dismiss them as a difficult and unsolvable problem. Rigid deployment controls that fail to adequately reflect the risk associated with different findings can create friction between security and the rest of the organization. Practitioners will waste time and energy pursuing issues that could safely be ignored or deprioritized".***

The volume of security findings and the inability to prioritize issues have created a concerning pattern in AppSec: vulnerabilities go unaddressed in the early stages of development — when fixes are simpler and less costly. Some teams may delay remediation until after deployment — when an exploited weakness can have far-reaching consequences.



The Need for A Holistic Prioritization Approach

To combat the ever-growing attack surface, organizations must adopt a more sophisticated approach to application security based on evidence-driven prioritization. This requires a shift from generic alert handling to a comprehensive model that covers code from design stages to runtime, and includes multiple elements:

Reachability Analysis:
Determines whether a vulnerability or weakness can be reached by an attacker based on code structure and logic, the application's architecture, and environmental factors such as access controls, configuration settings, and deployment environment.

Exploit Probability:
Focuses on understanding if the conditions required to exploit a vulnerability exist within each customer's unique environment, allowing for verification of specific exploitability traits in their particular context. This analysis is supported by established scoring systems like the Exploit Prediction Scoring System (EPSS) and CISA Known Exploitability Vulnerabilities Catalog (CISA KEV), which help assess the overall likelihood of a vulnerability being exploited.

Business Impact:
Analyzes the potential consequences of a successful attack, including financial loss, reputational damage, and disruption of operations.

Environmental Context:
Evaluates the operational environment where vulnerabilities reside, including the application's role, its exposure to external threats, and the organization's overall risk tolerance.

Dependency Mapping:
Identifies dependencies between components and prioritizes vulnerabilities that impact critical systems or services.

By implementing such a framework, organizations can effectively filter out the noise and focus their efforts on the small percentage of alerts that pose a genuine threat. This improves security effectiveness frees up valuable resources and enables more confident development practices.

1. Overall Findings and Organizational Security Profile

Finding Critical Needles in Alert "Hay-Stacks"

Our analysis revealed several important findings that have profound implications for application security and, ultimately, business risk reduction.

Overall Findings

Active Issues

101,344,969
Before Context-Based Prioritization

↓2,106,726
After Context-Based Prioritization

Active High & Critical Issues

11,096,892
Before Context-Based Prioritization

310,364
After Context-Based Prioritization

Active Critical Issues Only

1,072,086
Before Context-Based Prioritization

35,877
After Context-Based Prioritization

Alert Volume Reduction:
A dramatic **97.92%** reduction in overall active alerts was achieved - from over 101 million to approximately 2.1 million. This demonstrates the significant potential for reducing alert fatigue while maintaining comprehensive security visibility.

Critical Issue Refinement:
Initial critical issues were reduced by **96.65%.** "High" and "critical" issues combined decreased from 11 million to 310,364, allowing for more precise identification of truly critical issues and eliminating false positives.

Organizational level Impact:
The average organization uses 469 active applications and faces 569,354 security alerts, of which 6,023 are deemed critical. After implementing context-based prioritization, the number of issues per organization dropped dramatically to 11,836 — a mere **2.08%** of its original number.

Similarly, the number of critical issues dropped from 6,023 to 202 (**3.35%**) — bringing the alerts to manageable levels.

Security Profile of the Average Organization

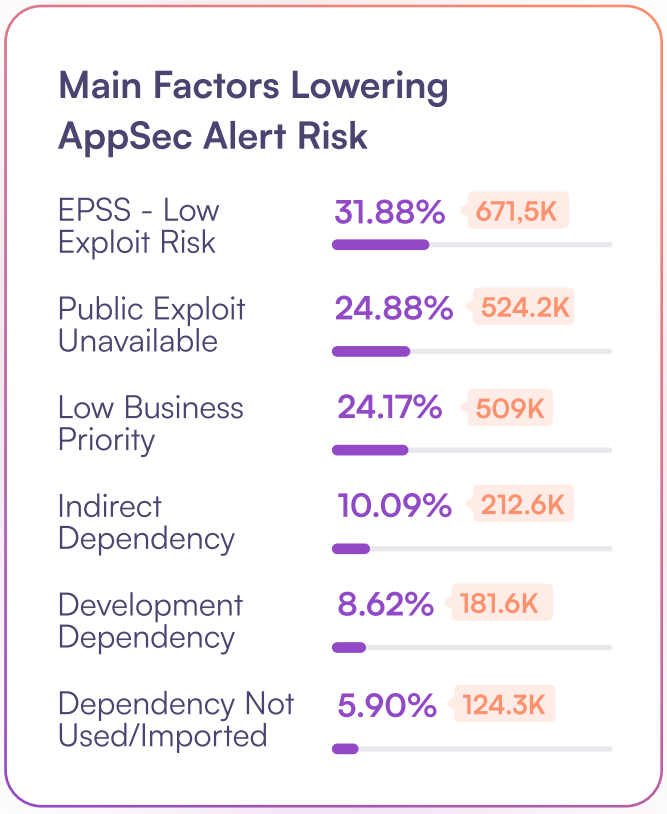
Active Apps Per Org **469**

	Before Prio.	After Prio.
Issues	569,354	11,836
	97.92%	2.08%
Critical issues	6,023	202
	96.65%	3.35%

2. The 95% Problem: Security Teams Beating a Majority of Dead Horses

2.1. The Anatomy of Alert Fatigue

The industry-wide alert fatigue crisis is widely covered and discussed. This phenomenon takes shape and form and is easily recognizable in our data:



Low Exploit Risk:
Over 31% of all issues were considered low exploit risk based on EPSS scores, meaning that for roughly a third of all vulnerabilities, attackers don't have a straightforward way to exploit them. Even when vulnerabilities exist, the attacker's ability to take advantage of them is very low.

Public Exploit Unavailable:
Nearly 25% of issues were deprioritized because no public exploit was available, which means there is currently no known method for exploiting these vulnerabilities. While these vulnerabilities exist in theory, they remain abstract without practical exploitation techniques. It's possible that some individuals or organizations have privately developed exploitation methods, but such knowledge is not widely available to the public.

Low Business Priority: Approximately 25% of findings were categorized as low business priority due to their limited impact on critical systems or processes. Analysis of both business impact and the development team's workload for the affected applications determined these findings offered minimal value to the business

Indirect/Development Dependencies:
A significant portion of alerts (25%) were related to indirect dependencies, development dependencies, or dependencies that were not used or imported. When analyzing source code, these dependencies typically don't reach production environments and are less exposed to malicious actors. They remain deeply buried within the code, with low chances of being exploited.

Put together, these findings show that organizations need intelligent filtering mechanisms to focus their limited security resources on the minority of issues that represent genuine, exploitable risks to business-critical assets.

> Organizations need intelligent filtering mechanisms to focus their limited security resources on the minority of issues that represent exploitable risks to business-critical assets

2.2.Code Hygiene: The Added Weight

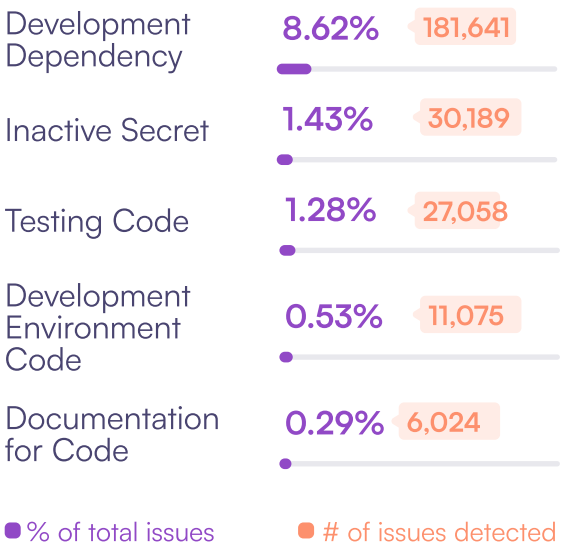
Code hygiene issues add unnecessary weight to security assessments. The following table demonstrates cases when a single risk label was sufficient to classify an alert as "info" level, meaning no remediation or intervention was required. These findings were, in fact, not vulnerabilities at all, making security team involvement unnecessary.

Currently, security scanners struggle to distinguish between actual security vulnerabilities and poor coding practices.

The **95%** problem, as demonstrated in the data, highlight the importance of contextual analysis and evidence-based prioritization. By considering factors beyond the initial vulnerability assessment, organizations can effectively filter out non-critical alerts and focus on those that pose a genuine threat.

In real-world conditions, high rates of false positives capture too much of security teams' attention, pulling them into reactive firefighting instead of strategic risk management. This inefficiency might signal the organization that security lacks the tools and maturity to handle modern threats effectively.

Security Alerts Reclassified as Code Hygiene Issues

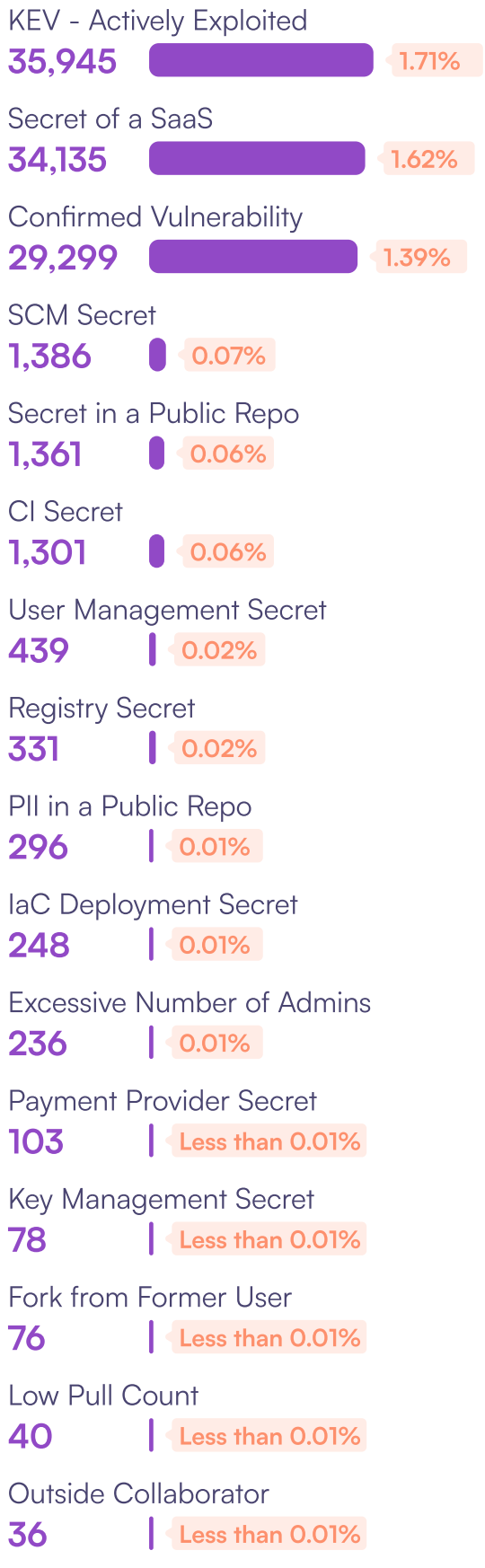


Currently, security scanners struggle to distinguish between **actual security vulnerabilities and poor coding practices**

3. The 5%: Rare Gems of Bad Practice, Management Issues & One Surprising Champion

While most alerts can be safely deprioritized, it is important to accurately identify the 2-5% that require immediate attention. Our research revealed several factors that contribute to risk increase, as detailed in the table chart:

Main Factors Increasing AppSec Alert Risk



* Fork from former user: meaning, to create a copy - fork- of a repository originally maintained by a previous owner. It also indicates a transition of ownership

* Low pull count: meaning, a small number of downloads or retrievals of a resource, potentially indicating that a resource is untested or unmaintained.

3.1. Our Champion, KEV (Funny to See You Here)

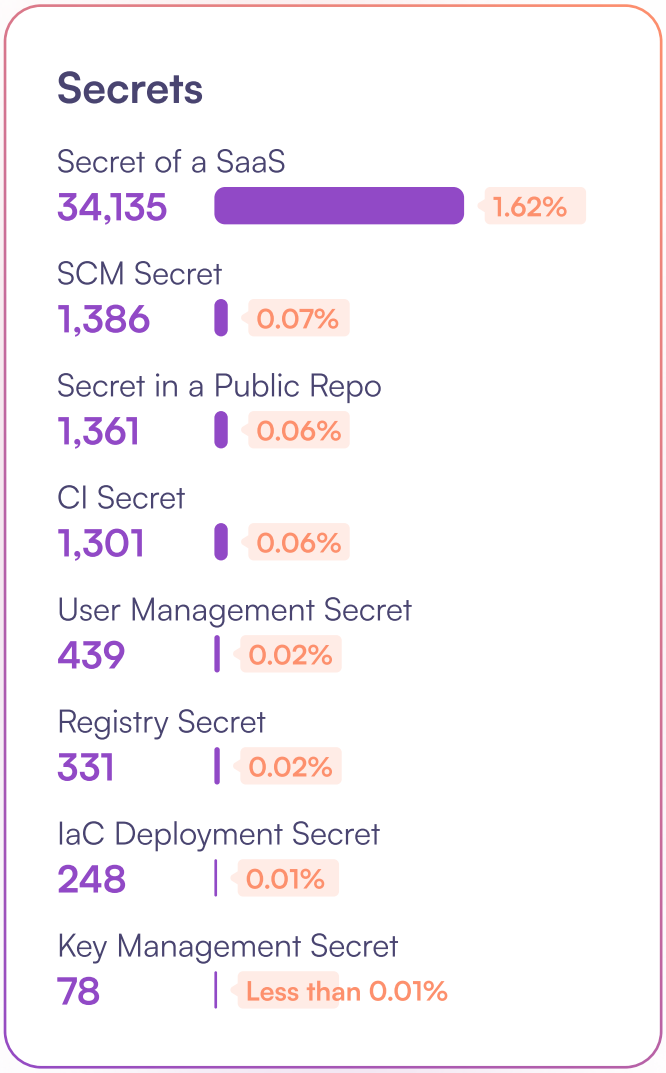
1.71% of all issues (about 36,000) were identified as Known & Exploited Vulnerabilities (KEV).

The KEV catalog, managed by CISA, compiles vulnerabilities that we know have been exploited and are actively being used in attacks.

These are no longer theoretical vulnerabilities; they have been practically exploited and can be exploited again. As a result, they represent high risk. We would expect that critical vulnerabilities that appear in the catalog would be treated differently - and given priority over vulnerabilities that are not identified in KEV.

In the next report, it will be interesting to see if the percentage of issues that are KEVs is decreasing or increasing. Is the rate of exploitation outpacing the rate of remediation among organizations?

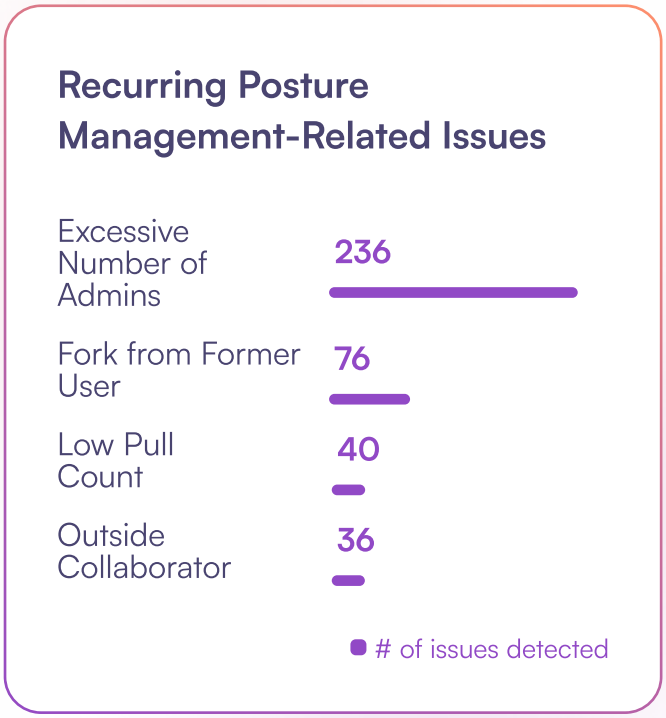
3.2. Secrets: Bad Practice Turned into Risk



Stemming from poor development practices, the exposure of secrets is one of the most prevalent high-risk security findings. Developers often take the convenient route of embedding sensitive credentials (API keys, passwords, tokens) directly into code. This creates a significant vulnerability: anyone with code access effectively gains keys to the entire system—similar to handing someone your house key along with your address.

Best practice requires using dedicated key management services that securely store credentials, such as AWS Key Management Service or Azure Key Vault. **Moving forward, we expect to see a decline in this vulnerability as security teams prioritize addressing secrets in code due to both the ease of exploitation and the painless remediation process.** Additionally, more organizations are implementing pipeline-level controls that block the release of applications containing hardcoded secrets.

3.3 Posture Management: Infrequent but Potentially Devastating



Some of the issues presenting high risk point to improper security practices, rather than actual weaknesses. These practices increase the attack surface and expose the organization further to attacks. **Given that the frequency of issues related to posture management is low, we can conclude that the industry generally maintains good control over this issue. However, the potential risk remains very high.** If proper access restrictions aren't implemented using a Zero Trust framework, each of these issues could lead to an incident with significant impact.

4. Industry Benchmark & Real World Examples

4.1. AppSec Alerts Industry Benchmark

Software & Technology

	Before Prio.	After Prio.	Noise	Real Risk
Industry Metrics	728,338	15,859	97.82%	2.18%
Entreprise	3,393,928	60,996	98.20%	1.80%
SMB	601,405	13,760	97.71%	2.29%

Financial Services

	Before Prio.	After Prio.	Noise	Real Risk
Industry Metrics	1,674,948	29,862	98.22%	1.78%
Entreprise	5,258,594	98,087	98.13%	1.87%
SMB	1,140,230	16,217	98.58%	1.42%

Transportation & Logistics

	Before Prio.	After Prio.	Noise	Real Risk
Industry Metrics	1,878,408	32,495	98.27%	1.73%
Entreprise	3,554,915	63,390	98.22%	1.78%
SMB	1,319,572	22,197	98.32%	1.68%

Other | Including Healthcare, Energy, Communications & Media, Gaming

	Before Prio.	After Prio.	Noise	Real Risk
Industry Metrics	925,262	17,252	98.27%	1.86%
Entreprise	1,254,860	21,379	98.22%	1.70%
SMB	784,006	15,483	98.32%	1.97%

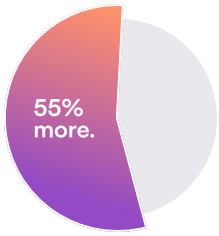
Industry Benchmark Insights

Enterprise security complexity:
Enterprise security environments face significantly greater challenges due to their broader tool ecosystem, larger application footprint, higher volume of security events, more frequent incidents, and elevated overall risk exposure.

Baseline noise thresholds remain remarkably similar - around **98%** - across different industries and company sizes

Consistent baseline noise thresholds:
Industry benchmarks reveal that baseline noise thresholds remain remarkably similar across different environments—whether enterprise or commercial—regardless of industry.

Financial institutions
Higher alert volumes



Financial institutions experience distinctively higher alert volumes - **up to 55% more.** Their proximity to monetary transactions and sensitive data makes them high-value targets

The Financial sector's unique vulnerability:
Financial institutions experience distinctively higher alert volumes. Their processing of financial transactions and sensitive data makes them high-value targets. As the Verizon Data Breach Investigations report indicates, 95% of attackers are motivated primarily by financial gain rather than espionage or other reasons. Financial institutions's proximity to monetary assets creates direct profit opportunities for attackers.

4.2. Zoom in: Three Companies and Their Findings

Examples of real organization metrics before and after context-based prioritization

Media Enterprise

Active Apps: 6,472

	Before Prio.	After Prio.	Noise	Real Risk
Issues	465,650	4,997	98.93%	1.07%
Critical issues	15,705	40	99.75%	0.25%

FinTech Enterprise

Active Apps: 2,090

	Before Prio.	After Prio.	Noise	Real Risk
Issues	681,167	19,528	97.13%	2.87%
Critical issues	12,140	2	99.98%	0.02%

Edge Computing Enterprise

Active Apps: 808

	Before Prio.	After Prio.	Noise	Real Risk
Issues	321,837	7,203	97.76%	2.24%
Critical issues	24,742	10	99.96%	0.04%

5. Environmental CWE Landscape vs. MITRE Top 25

Top 10: Most Common CWEs in the Researched Environment

	# of Issues detected	Rank in MITRE top 25	Description
1 CWE - 400	277,937	24	Uncontrolled Resource Consumption*
2 CWE - 20	125,310	12	Improper Input Validation
3 CWE - 798	123,518	22	Use of Hard-coded Credentials
4 CWE - 200	104,827	17	Exposure of Sensitive Information to an Unauthorized Actor
5 CWE - 798	100,207	2	Out-of-bounds Write
6 CWE - 79	81,717	1	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
7 CWE - 22	80,231	5	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
8 CWE - 94	70,828	11	Improper Control of Generation of Code ('Code Injection')
9 CWE - 476	66,859	21	NULL Pointer Dereference
10 CWE - 190	64,877	23	Integer Overflow or Wraparound

Examining the most common CWEs (Common Weakness Enumerations) in our database and comparing them to MITRE's Top 25 CWE rankings reveals that industry-wide trends might not always apply to individual cases. While all our top 10 CWEs are also present in MITRE's Top 25, their order differs.

This suggests that MITRE's overall ranking may not always be accurate or relevant at the organizational or industry level. For instance, the most commonly found CWE in our study is CWE-400 (Uncontrolled Resource Consumption) with over 277,000 occurrences, yet it is ranked 24th in MITRE's list.

Conversely, MITRE's top-ranked CWE, CWE-79 (Improper Neutralization of Input During Web Page Generation, also known as Cross-site Scripting), ranks sixth in our environments. This gap underscores the need for organizations to analyze their own most common vulnerabilities instead of relying solely on industry rankings.

It will be interesting to see if in future editions, these differences persist or if industry rankings begin to align with prevalence across organizations. Organizations should continue monitoring their specific vulnerability landscapes to ensure their security priorities match their actual risk profiles.

The Cost of AppSec Inefficiency

The implications of the report's findings are numerous and far-reaching. If less than 95% of application security fixes are critical to the organization, then all organizations invest enormous resources in triage, programming, and cybersecurity hours in vain.

This waste extends to payments for bug-bounty programs where white-hat hackers find vulnerabilities to fix, as well as the costs of complicated fixes for vulnerabilities that weren't discovered early and reached production.

The final significant cost is the tension created within organizations between development teams and security teams who demand fixes for vulnerabilities that aren't relevant. It's difficult to accurately estimate the costs of these unnecessary security fixes across the industry. However, in our next edition, we will present an estimated calculation based on a standard metric of 100 programmers to quantify this inefficiency.

Organizations invest enormous resources in triage, programming, and cybersecurity hours in vain. The waste extends to payments for bug-bounty programs

Conclusions: From Detection Overload to Strategic Defense; The Path Forward

Our Analysis of 101 million security findings reveals the traditional "detect everything" approach has reached its breaking point: security teams are crippled by unmanageable alert volumes, while critical vulnerabilities often go unaddressed. The data is unequivocal: only 2-5% of security alerts require immediate action, yet organizations continue to waste valuable resources on the remaining +95% of non-critical issues.

This leads to tools becoming a part of the problem instead of increasing security: Creating tensions between teams and leaving real issues untreated to later stages, when remediation is costly and exposure high.

Key Recommendations

1.
Shift from Vulnerability Management to Risk Management

Focus on high-impact vulnerabilities: Prioritize issues based on actual business risk rather than treating all CVEs equally

Implement automated risk prioritization: Leverage tools that analyze exploitability, reachability, and business impact to identify the critical 2-5% of vulnerabilities that pose genuine threats

Integrate business context: Align security decisions with application criticality, revenue impact, and organizational risk tolerance

2.
Build an End-to-End Application Security Pipeline

Create an integrated security approach: Connect security tools across the entire code-to-cloud lifecycle instead of relying on isolated scans

Adopt evidence-based security: Move from "detect everything" to a context-aware model that considers your organization's specific threat landscape

Correlate security data across SDLC: Link code vulnerabilities with runtime behavior, API risks, and infrastructure misconfigurations to identify real vs. theoretical threats

> The traditional "detect everything" approach has reached its limit: security teams are crippled by alerts, while critical vulnerabilities often go unaddressed.

3.
Assist Developers Instead of Overwhelming Them

hift security left: Integrate security tools and practices directly into development workflows

Provide actionable intelligence: Replace overwhelming vulnerability lists with contextual feedback that includes remediation guidance

Automate low-risk analysis: Free developer resources to focus on critical security issues requiring human judgment

4.

Foster Cross-Team Collaboration

Align metrics and objectives: Create standard success measures between security and development teams

Develop security awareness: Make security knowledge a core engineering competency through training and collaborative problem-solving

The future of application security lies not in addressing every possible vulnerability but in intelligently identifying and focusing on the issues that pose real risks. As AI increases in both attack and defense capabilities and software grows in complexity, teams that adopt this paradigm shift will effectively navigate the modern threat landscape and build resilience for the challenges ahead.

The future of application security lies not in addressing every possible vulnerability but in intelligently identifying and focusing on the issues that pose real risks.

About OX

OX Security revolutionizes application security by eliminating the distraction of generic prioritization and enabling teams to focus on the 5% of risks that truly matter. Traditional ASPM tools overwhelm security and development teams with commoditized alerts, wasting resources and slowing progress.

With OX's proprietary technology, organizations gain precise risk prioritization based on exploitability, reachability, and business impact — ensuring vulnerabilities are addressed before they reach runtime.

By integrating security seamlessly into the development lifecycle, OX replaces fragmented, siloed tools with a unified approach that empowers AppSec teams to act faster, reduce risk efficiently, and support developers without unnecessary friction.

Application Security Technical Terms Glossary

Alert Fatigue
A condition where security teams become desensitized to security alerts due to their high volume, leading to potentially missed critical threats.

AppSec (Application Security)
The process of finding, fixing, and preventing security vulnerabilities in software applications.

ASPM (Application Security Posture Management)
A category of security tools that provide visibility and control over an organization's application security risk posture across all applications and infrastructure.

CISA (Cybersecurity & Infrastructure Security Agency)
A United States federal agency responsible for improving cybersecurity across all levels of government.

CI Secret
Confidential information (like passwords or API keys) used in Continuous Integration systems.

CVE (Common Vulnerabilities and Exposures)
A system providing a reference method for publicly known information-security vulnerabilities and exposures.

CWE (Common Weakness Enumeration)
A community-developed list of software and hardware weakness types.

EPSS (Exploit Prediction Scoring System)
A data-driven effort to estimate the probability that a software vulnerability will be exploited in the wild.

FIRST (Forum of Incident Response and Security Teams)
A global organization that brings together computer security incident response teams.

Fork from Former User
A copy of a code repository that was created by a user who is no longer part of the organization, representing a potential security risk as the code may contain sensitive information or vulnerabilities that are no longer under direct organizational control.

IaC (Infrastructure as Code)
The practice of managing and provisioning infrastructure through code instead of manual processes.

KEV (Known Exploited Vulnerabilities)
A catalog maintained by CISA that lists vulnerabilities that are actively being exploited by threat actors.

Low Pull Count
A security risk indicator where a code repository or package has very few downloads or pull requests, which could indicate it's either untrusted, abandoned, or potentially malicious. Low engagement from the developer community may suggest the code hasn't been thoroughly vetted for security issues.

MITRE
A non-profit organization that operates research and development centers sponsored by the federal government, known for maintaining the CVE list.

MITRE Top 25
A list of the most dangerous software weaknesses, as determined by MITRE Corporation.

PII (Personally Identifiable Information)
Any data that could potentially identify a specific individual.

SaaS (Software as a Service)
A software licensing and delivery model where software is centrally hosted and licensed on a subscription basis.

SAST (Static Application Security Testing)
A testing methodology that analyzes source code to find security vulnerabilities.

SCA (Software Composition Analysis)
A tool or process to identify and track third-party and open source components in applications.

SCM Secret
Confidential information stored in Source Code Management systems.

Vulnerability
A weakness in a system that could be exploited to perform unauthorized actions.

Additional Terms
Active Apps
Applications that are currently in use and being monitored for security issues.

Critical Issue
A security vulnerability that poses immediate and serious risk to the application or organization.

Development Dependency
A software component required only during the development phase, not in production.

Indirect Dependency
A secondary dependency that is required by a direct dependency but not explicitly declared in the project.

Reachability
The determination of whether a vulnerability can be accessed or exploited by an attacker based on the application's architecture and environmental factors.

Secret
Any confidential information such as passwords, API keys, or authentication tokens that should be protected from unauthorized access.