ORCA SECURITY

2026

State of AppSec Report

When Development Velocity Exceeds Security Maturity

ORCA | RESEARCH POD

©2026 ORCA SECURITY. ALL RIGHTS RESERVED.

Application security has fundamentally changed, but many programs still operate as if it hasn't. Software is built on open-source dependencies, automated pipelines, and infrastructure as code, while AI is increasing both scale and risk. Yet security teams are expected to manage this complexity with outdated approaches.

Across real production environments, risk is visible but rarely actionable without context. AI is accelerating development and expanding the attack surface, from generated code to model dependencies, making prioritization essential. This report helps organizations understand where traditional approaches fall short and how to focus on the changes that materially reduce risk."

GIL GERON,
CEO AND CO-FOUNDER OF ORCA SECURITY

Inside This Report

Foreword 01

About the Orca Research Pod 02

Executive summary 03

Key findings 04

1. The Rise of Supply Chain Attacks 05
   - 1.1 Major Supply Chain Attacks 07

2. Vulnerabilities in AI Packages 08
   - 2.1 Critical Remote Code Execution Vulnerabilities 10
   - 2.2 Malicious Packages: Still Lurking in Production 11

3. Container Vulnerability Landscape 12
   - 3.1 High/Critical Vulnerability Patching Velocity 14

4. Secrets Management 15
   - 4.1 The AI/ML Secrets Crisis 17

5. CI/CD Pipeline Security 18
   - 5.1 CI/CD Platform Adoption 20
   - 5.2 GitHub Actions Security 21

6. Infrastructure as Code Security 22
   - 6.1 IaC Platform Adoption 24
   - 6.2 Storage and Data Protection 25
   - 6.3 Identity and Access Management 26
   - 6.4 Network Security 27
   - 6.5 Container Security in IaC 28

7. Repository and SCM Security 29
   - 7.1 Code Review and Approval Gaps 31
   - 7.2 Branch Protection Weakness 32
   - 7.3 Access Control and Hygiene 33

8. Key Recommendations 34
   - 8.1 Immediate Actions (0-30 Days) 35
   - 8.2 Short-term Initiatives (30-90 Days) 36
   - 8.3 Strategic Improvements (90+ Days) 37

9. Conclusion 38

Foreword

As organizations accelerate software delivery through cloud-native architectures, open-source dependencies, and automated pipelines, application attack surfaces are expanding faster than security practices can keep up. AI-assisted development is further increasing this velocity, generating code, dependencies, and configurations at a pace that traditional security processes were not designed to govern.

Modern applications are built from thousands of third-party components and deployed at machine speed. This velocity enables scale and innovation, but it also makes it impossible to fix everything once code reaches production. Vulnerable dependencies, exposed secrets, and insecure configurations are no longer edge cases; they are structural realities of how software is built today. At the same time, AI systems introduce new risks and enable the rapid propagation of insecure code patterns and model dependencies across environments.

These challenges are compounded by the rise of software supply chain attacks, which have proven to be one of the most effective paths to large-scale compromise. A single poisoned dependency or workflow can cascade across thousands of organizations, turning application security failures into operational risk.

This State of Application Security Report is designed to help teams understand where these risks are introduced and how to address them effectively. Grounded in real-world findings from the Orca Research Pod, it provides a clear view into the current Application Security landscape and practical guidance for securing modern applications at the speed today's businesses demand.

Gil Geron
CEO and Co-Founder of Orca Security

About the Orca Research Pod

The Orca Research Pod is a group of security researchers that discover and analyze security risks and vulnerabilities to strengthen the Orca Security Platform and promote CNAPP security best practices.

Research Methodology

This report is based on aggregated, anonymized security telemetry from more than 1,000 production organizations leveraging Orca Security's cloud security platform.

All metrics presented represent the percentage of organizations exhibiting each finding, calculated as a weighted average across the organizations. Data was collected between Q3 2025 and Q1 2026, focusing exclusively on production environments to ensure findings reflect real-world security postures rather than test or development configurations.

Security findings span multiple domains: CI/CD pipeline security, secrets management, repository configuration, software composition analysis (SCA), static application security testing (SAST), infrastructure as code (IaC), and container security.

Report Data Set:

- Cloud workload and configuration data
- Billions of real-world production cloud assets
- Data referenced in this report was collected from Q3 2025 to Q1 2026
- AWS, Azure, Google Cloud, Oracle Cloud, and Alibaba Cloud environments

Executive Summary

Leveraging real-world telemetry from more than 1,000 production organizations, this report examines the current state of application security across the modern software delivery lifecycle. The findings reveal a growing disconnect between how applications are built today and how application security is practiced, resulting in persistent risk, stalled remediation, and widening exposure across production environments. Our main findings summary includes:

- **Application risk is widespread and routinely reaches production**
  More than 78% of organizations have applications running with critical vulnerabilities. Even years after disclosure, high-impact issues like Log4Shell continue to affect nearly half of production environments, highlighting how traditional AppSec approaches struggle to keep pace with dependency sprawl.

- **Secrets exposure remains pervasive, and AI adoption is amplifying impact**
  Nearly one-third of organizations expose valid, active secrets in code, while over 41% have leaked AI/ML credentials. These exposures grant direct access to proprietary models, sensitive data, and usage-based services, increasing both security and financial risk.

- **Detection is not translating into remediation**
  While vulnerabilities are identified, they are rarely resolved. Over 77% of organizations still carry high or critical container vulnerabilities after 90 days, revealing that when teams can't effectively prioritize risk, vulnerabilities aren't remediated.

- **CI/CD Pipelines and Infrastructure-as-Code are scaling risk by default**
  With 74% of organizations deploying infrastructure through code, misconfigurations are no longer isolated mistakes, they are repeatable production risks. Over 80% of IaC environments lack proper logging and monitoring, and 84% deploy unencrypted storage, embedding security gaps directly into delivery pipelines.

- **Supply chain attacks are now a routine production risk**
  More than 11% of organizations have well known malicious packages in production, including dependencies publicly disclosed and removed years ago. Self-replicating attacks like the 2025 Shai-Hulud worm demonstrate how a single compromised dependency can cascade across thousands of downstream environments.

Key Findings

31%

of organizations expose valid secrets in source code repositories.

Hardcoded credentials provide attackers with direct access to systems and data, turning simple code exposure into immediate, potentially large scale, security incidents.

43%

of organizations have exposed AI or machine learning credentials.

As AI adoption accelerates, leaked model and API credentials introduce new risks, including unauthorized access, data leakage, and unexpected financial impact.

78%

of organizations run applications with critical vulnerabilities

High-severity issues still reach production, showing that CVSS scores alone, without exploitability context, no longer drive effective prioritization or remediation.

80%

of organizations lack logging or monitoring in Infrastructure deployed as code

When infrastructure is defined programmatically without visibility, misconfigurations and compromises can persist undetected across multiple environments.

77%

of organizations have high or critical container vulnerabilities unpatched after 90 days.

While vulnerabilities are detected early, remediation stalls when teams lack the context needed to determine which findings pose real production risk.

50%

of organizations remain vulnerable to Log4Shell-affected dependencies in production.

The persistence of widely publicized, high-impact vulnerabilities years after disclosure shows how deeply embedded and hard to remediate transitive dependencies are.

11%

of organizations have well known malicious packages running in production.

Software supply chain attacks are no longer edge cases, with malicious dependencies continuing to propagate long after they are publicly disclosed and removed from registries.

75%

of organizations manage infrastructure through code.

When infrastructure is defined and deployed programmatically, misconfigurations such as unencrypted storage and missing logging are repeatedly pushed into production at scale.

01 The Rise of Supply Chain Attacks

18,000+

DOWNSTREAM ORGANIZATIONS

The Rise of Supply Chain Attacks

Software supply chain attacks have shifted from edge-case threats to one of the most reliable paths to large-scale compromise. By targeting shared dependencies, build systems, and automation workflows, attackers can achieve exponential impact from a single intrusion.

The SolarWinds breach in 2020 marked a turning point, demonstrating how compromising one build pipeline could grant access to **18,000+ downstream organizations**, including Fortune 500 companies and government agencies. Since then, attackers have increasingly focused on package registries, CI/CD platforms, and maintainer credentials, where trust is implicit and validation is often limited.

This evolution accelerated in 2025 with the emergence of self-replicating supply chain malware. The Shai-Hulud campaigns introduced a new class of attack that spreads autonomously by harvesting npm tokens and GitHub credentials from infected environments. Shai-Hulud 2.0 alone compromised over 796 npm packages with more than 20 million weekly downloads, exposing 14,000 secrets across 487 organizations.

These attacks underscore a new reality, that modern software is only as secure as the weakest dependency, maintainer account, or automation workflow it relies on.

Major Supply Chain Attacks

Software supply chain attacks have evolved from isolated incidents into one of the most effective paths to large-scale compromise. Over the past five years, their sophistication and impact have increased sharply, as attackers exploit the force-multiplier effect of compromising shared dependencies, build systems, and automation workflows.

The 2020 SolarWinds breach marked a turning point, demonstrating how a single compromised build pipeline could grant access to **more than 18,000 downstream organizations**, including government agencies and Fortune 500 companies. Since then, attackers have increasingly targeted package registries, CI/CD platforms, and maintainer credentials, as these are areas where trust is implicit and security controls are often weakest.

This escalation accelerated in 2025 with the emergence of self-replicating supply chain malware. Campaigns like Shai-Hulud introduced attacks capable of spreading autonomously by harvesting npm tokens and GitHub credentials from infected environments. Shai-Hulud 2.0 alone compromised over 796 npm packages with more than 20 million weekly downloads, exposing 14,000 secrets across 487 organizations. This signals a shift toward persistent, scalable operations that weaponize trust across the software ecosystem.

	ATTACK / CAMPAIGN	TYPE	IMPACT
2025	Shai-Hulud 2.0	npm Self-Replicating Worm	**796+** npm packages, **20M+** weekly downloads
2025	React shell (CVE-2025-55182)	RCE Vulnerability	**CVSS 10.0, 55.8M+** weekly downloads
2025	tj-actions/changed-files	GitHub Actions Compromise	**23,000+** repositories
2024	XZ Utils Backdoor	Maintainer Social Engineering	**CVSS 10.0,** multi-year operation
2024	3CX / SmoothOperator	Double Supply Chain Attack	**242,519+** IPs compromised
2023	MOVEit	Zero-Day Exploitation	**600+** organizations
2021	Log4Shell (Log4j)	Critical Library Vulnerability	**CVSS 10.0,** billions of devices
2020	SolarWinds / SUNBURST	Build System Compromise	**18,000+ customers**

02 Dependency & Vulnerability Management

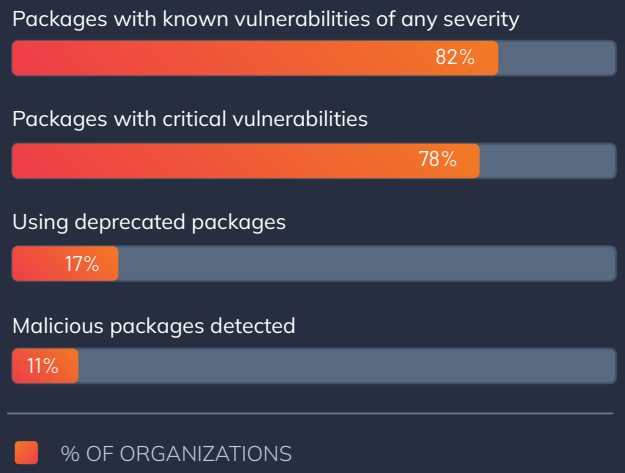
Dependency and Vulnerability Management

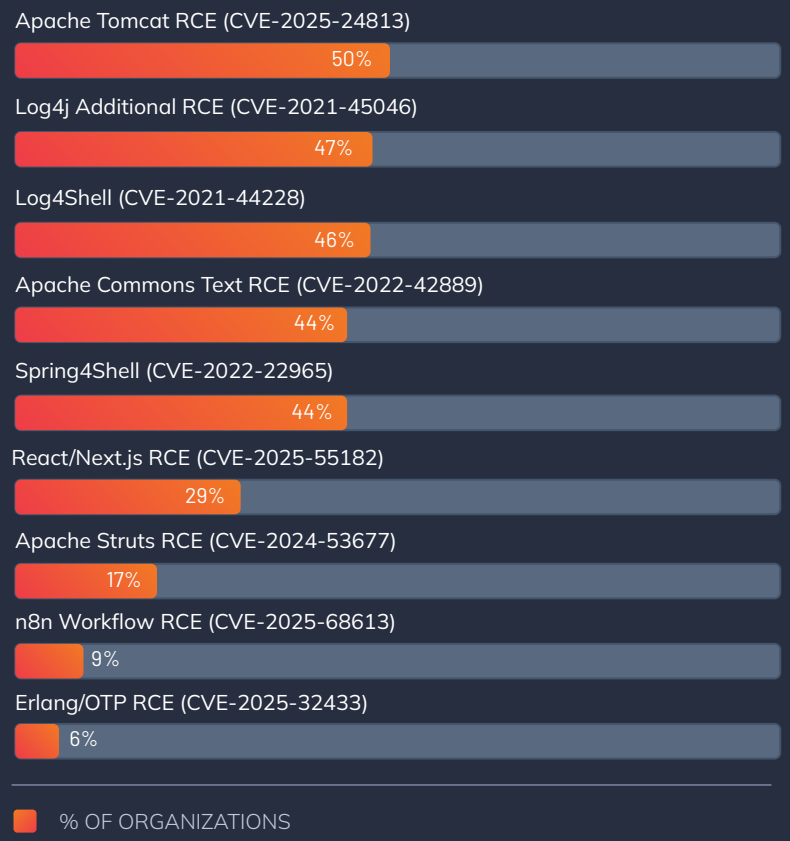
Modern applications are built on layers of third-party code, with much of it shipping with known risk. Open-source dependencies accelerate development, but they also introduce vulnerabilities that are difficult to track, prioritize, and remediate across complex dependency trees. Our research shows that vulnerable packages are not isolated findings, but pervasive across production environments.

More than 78% of organizations have applications running with critical vulnerabilities. As dependency sprawl grows, traditional AppSec approaches that rely on severity-based alerting struggle to distinguish between theoretical risk and issues that pose real exposure in production.

The result is a growing backlog of vulnerabilities that teams are aware of but lack the context to confidently and efficiently address.

Dependency Security Findings





2.1 Critical Remote Code Execution Vulnerabilities

Certain vulnerabilities demand immediate attention due to severity and ease of exploitation, however many evade remediation.

The persistence of Log4Shell nearly four years after disclosure, still **affecting 46% of organizations**, underscores a broader challenge in modern AppSec that even when vulnerabilities are widely publicized and patches are available, remediation can still stall. At the same time, new RCE vulnerabilities like React2Shell (CVE-2025-55182), which affects 29% of organizations and applies to all React 19 and Next.js 15/16 applications using React Server Components by default, are rapidly expanding the attack surface of modern application frameworks.

Together, these findings show that severity alone is not enough to drive effective remediation. Without validation, prioritization, and production context, even the most critical vulnerabilities remain open because teams struggle to gather the insight needed to determine which risks are truly exploitable and demand immediate action.

2.2 Malicious Packages: Still Lurking in Production

Despite public disclosure and removal from package registries, confirmed malicious packages continue to persist in production, in some cases dating back to 2018. A single malicious package in production can enable credential theft, data exfiltration, or full system compromise. This makes even limited exposure a significant organizational risk. Our analysis shows that organizations are still running known supply chain threats, exposing persistent gaps in dependency governance and remediation practices.

Notably, the most prevalent threats identified originated from the same trusted maintainer, reinforcing that modern supply chain attacks do not always come from unknown external actors. In many cases, the threat is introduced through trusted dependencies already embedded within the application ecosystem.

PACKAGE	VERSION(S)	ECOSYSTEM	ATTACK TYPE	% Of ORGS
Faker	6.6.6	npm	Maintainer Sabotage	7%
Event-stream	3.3.6	npm	Account Takeover / Bitcoin Stealer	4%
flatmap-stream	0.1.1	npm	Malicious Dependency Injection	3%
colors	1.4.1, 1.4.2, 1.4.44-liberty-2	npm	Maintainer Storage	2%
CTX	0.1.2, 0.2.2, 0.2.6+	PyPI	AWS Takeover / AWS Credential Stealer	1%
larpexodus	0.1	PyPI	Typosquatting / Infostealer	<1%

03 Container Vulnerability Landscape

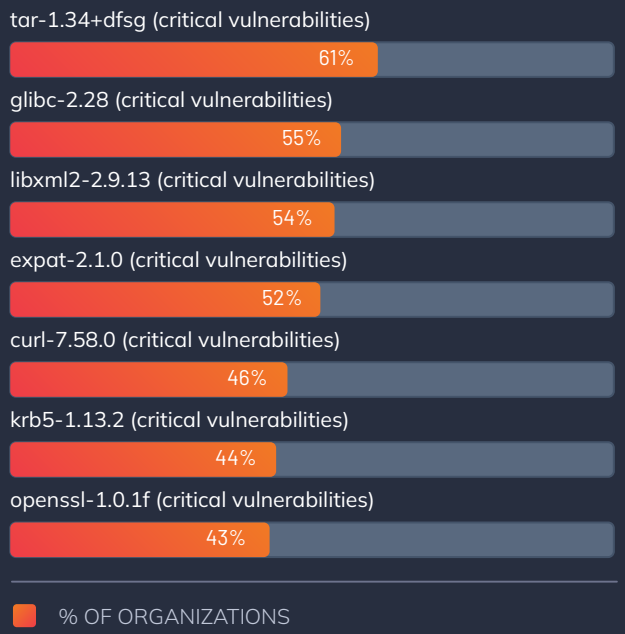
Container Vulnerability Landscape

Containers are a foundational component of modern application delivery, but many production images are built on outdated and vulnerable base layers. Our research shows that critical vulnerabilities are deeply embedded in commonly used container packages.

Over 60% of organizations run containers with critical vulnerabilities in core system packages such as tar (61%) and glibc (55%). These components sit at the foundation of container images and are inherited across applications, meaning a single vulnerable base image can propagate risk across entire environments.

43% of organizations still run containers with OpenSSL 1.0.1f, a version affected by Heartbleed and other high-impact vulnerabilities, reflecting the widespread use of base images that are not regularly updated. This allows well-known, high-severity vulnerabilities to persist in production workloads.

Vulnerable Container Packages

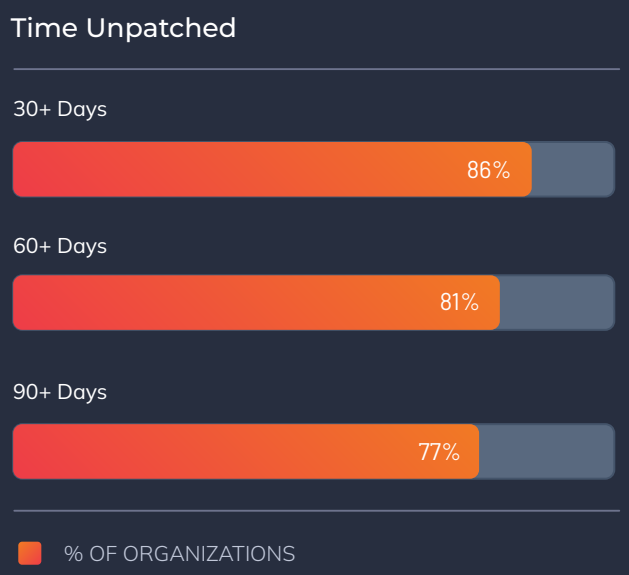


High/Critical Vulnerability Patching Velocity

While container vulnerabilities are widely detected, remediation remains slow and inconsistent. Our research found that **77% of organizations still have high or critical container vulnerabilities unpatched after 90 days**, revealing a significant gap between detection and action.

The data shows that delays begin early and compound over time. 86% of organizations retain high or critical vulnerabilities for more than 30 days, and 81% continue to carry them beyond 60 days. Organizations that don't patch vulnerabilities quickly are unlikely to patch at all.

With the vast majority of organizations (70%) tolerating critical and high-risk vulnerabilities for months, container vulnerability management has become less about remediation and more about risk acceptance. Closing this gap requires not just identifying vulnerabilities, but enabling teams to prioritize and act on the risks that truly demand immediate attention.



04 Secrets Management

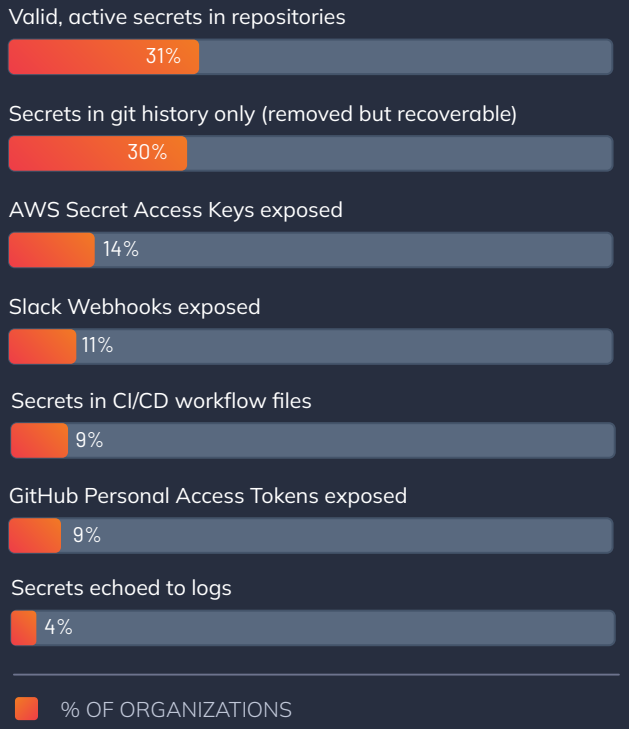
Secrets Management

Modern applications rely on secrets, such as API keys, tokens, passwords, and certificates, to connect services, access data, and automate workflows. As development velocity increases, secrets management remains a fragile and immature area of AppSec.

Our research shows that secrets are frequently hardcoded, committed to repositories, and retained long after they should be revoked. These exposures are routinely found in active production code, which creates a direct path to compromise.

More than 31% of organizations expose valid secrets in source code repositories, while 30% retain secrets in git history, leaving credentials recoverable by anyone with repository access even after they appear to have been removed. Once exposed, these secrets often remain active, granting attackers persistent access to internal systems, cloud services, and sensitive data.

Secrets Exposure Findings



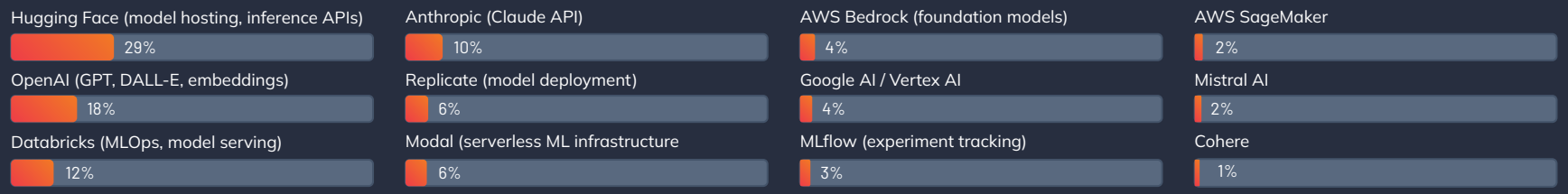
The AI/ML Secrets Crisis

The rapid adoption of AI and machine learning has introduced a new category of secrets exposure. Our analysis found that **43% of production organizations have exposed AI/ML credentials**, reflecting how quickly teams are integrating AI services without establishing proper secrets hygiene.

The risk is amplified by the nature of these services. Hugging Face tokens, for example, can grant access to private models, internal datasets, and inference endpoints. A single compromised token can enable intellectual property theft, model poisoning, or the unauthorized consumption of expensive GPU resources.

Unlike traditional API keys, AI/ML secrets introduce compounding risk. Usage-based billing models can lead to massive, unexpected costs. Access to proprietary models represents direct intellectual property theft. And compromised inference endpoints can be abused to generate malicious or harmful content attributed to the victim organization.

AI/ML Secrets Exposure by Service



% OF ORGANIZATIONS

05 CI/CD Pipeline Security

CI/CD Pipeline Security

CI/CD pipelines sit at the center of modern software delivery, automating the build, test, and deployment of applications at speed. As organizations push more logic and credentials into these pipelines, CI/CD systems have become high-value targets for attackers, yet security controls have not kept pace with their growing privilege and reach.

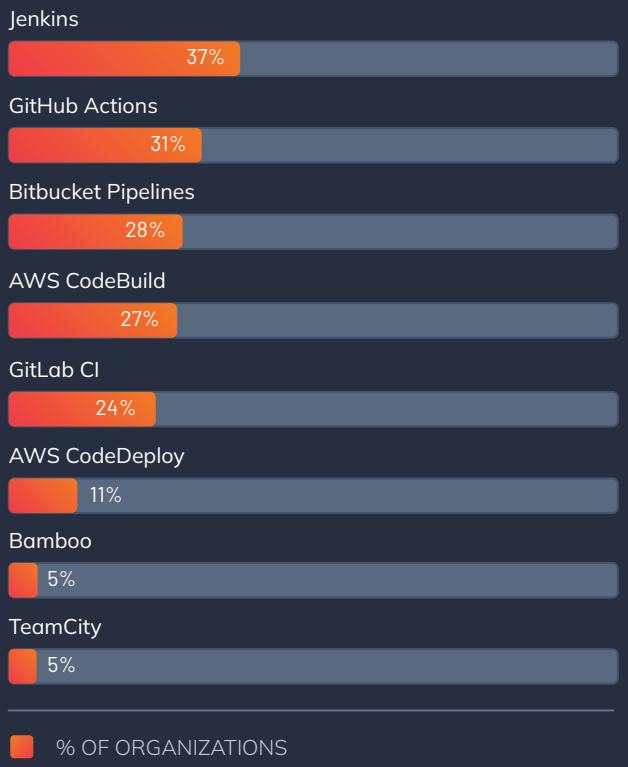
Our research shows that CI/CD pipelines frequently operate with excessive permissions, limited oversight, and minimal validation of third-party actions. When compromised, these systems provide attackers with a powerful launch point into source code, production infrastructure, and sensitive credentials.

5.1 CI/CD Platform Adoption

Our research shows broad adoption of CI/CD platforms across production organizations, reinforcing the importance of securing these systems as critical infrastructure. As pipelines become more interconnected with cloud environments, identity systems, and deployment tooling, misconfigurations in CI/CD platforms increasingly translate into production risk.

Without consistent security controls, visibility, and governance, CI/CD platforms can unintentionally amplify application risk rather than reduce it.

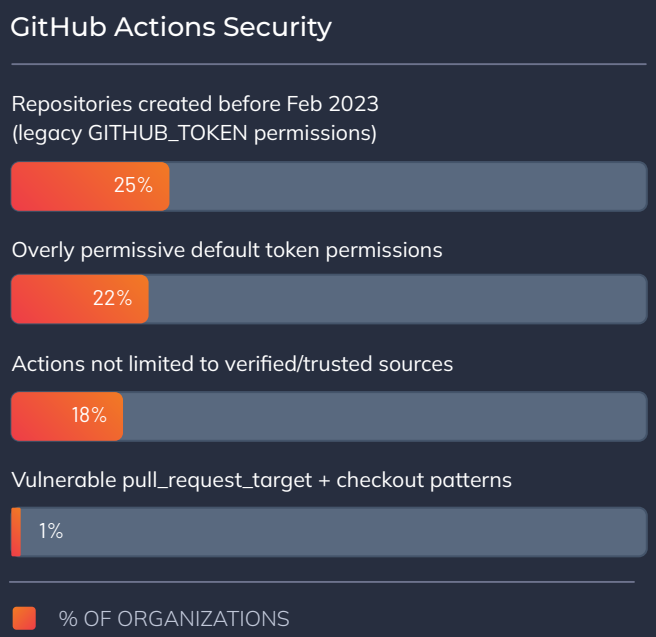
CI/CD Platform Usage



5.2 GitHub Actions Security

GitHub Actions has become a foundational component of modern CI/CD pipelines, enabling teams to automate workflows through reusable actions and third-party integrations. However, this flexibility introduces new security challenges when workflows are built on overly permissive permissions and unvalidated components.

Our research found that **25% of repositories still rely on legacy or overly permissive GitHub token permissions**, increasing the risk of credential abuse and unauthorized workflow execution. When combined with third-party actions and automated deployment privileges, these permissions can allow attackers to move from code changes to production access with minimal resistance.



06 Infrastructure as Code Security

Infrastructure as Code Security

Infrastructure as Code (IaC) has fundamentally transformed how organizations deploy and manage cloud resources. As IaC adoption continues to accelerate, however, security controls have not evolved at the same pace.

Our research shows that **75% of organizations manage infrastructure through code**. IaC includes cloud-native templates, Kubernetes manifests, and configuration-driven container deployment. These misconfigurations introduced through IaC are repeatable, automated, and often deployed across multiple environments simultaneously.

Without strong validation and governance, IaC can turn small configuration gaps into large-scale production risk.

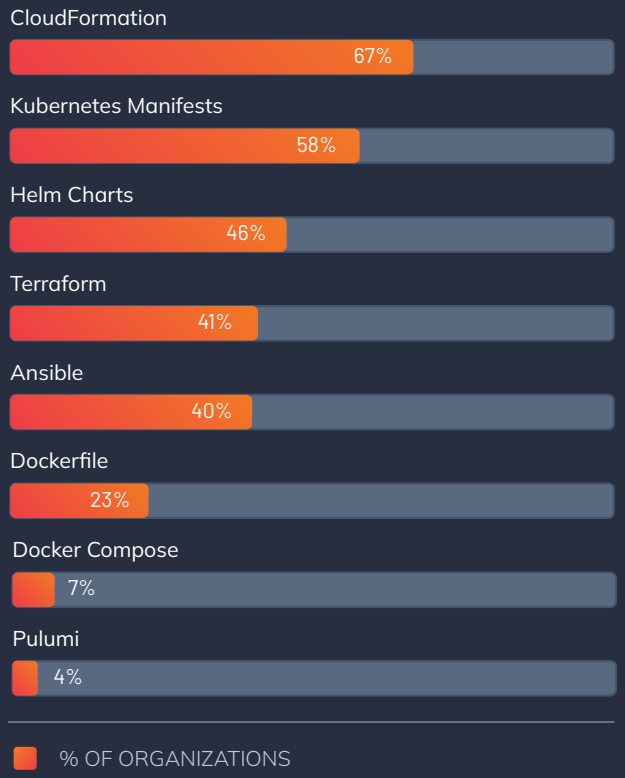
6.1 IaC Platform Adoption

IaC platforms are widely adopted across production organizations, reflecting the shift toward automated, cloud-native infrastructure management. **75% of organizations use Infrastructure as Code to deploy cloud resources**, reflecting the industry-wide shift toward automated infrastructure management.

This standardization accelerates deployment, but also concentrates risk when IaC templates are copied, shared, and reused across environments. In complex environments, teams may not fully understand the security implications of infrastructure changes until after deployment.

Securing IaC platforms is therefore critical to preventing systemic risk across cloud environments.

Infrastructure as Code Platform Usage



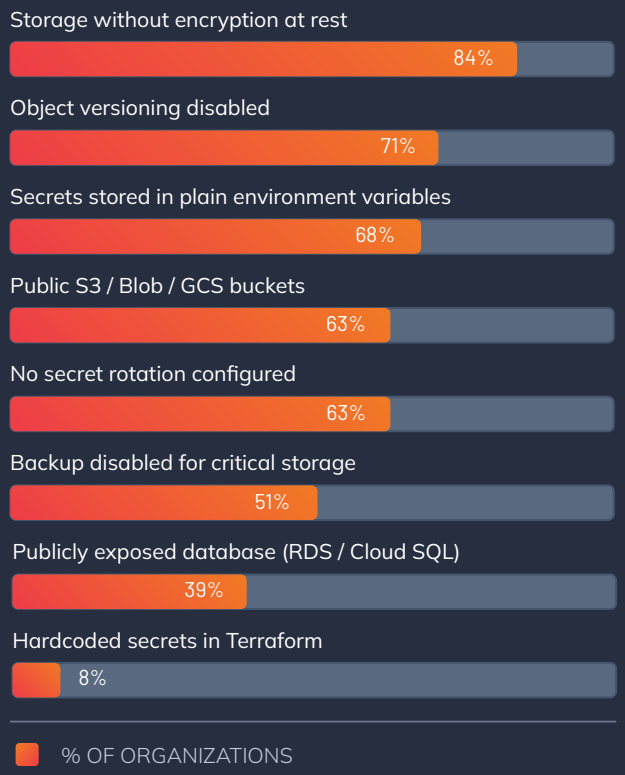
6.2 Storage and Data Protection

Storage misconfigurations remain one of the most common and impactful risks introduced through IaC, with potential for data breaches and compliance violations. Our research found that **84% of organizations deploy unencrypted storage resources** through infrastructure code, embedding data exposure directly into production environments.

In addition, **80% of organizations lack sufficient logging and monitoring controls** for IaC-managed storage. Without encryption and visibility, sensitive data can be exposed or accessed without detection for extended periods of time.

When insecure storage configurations are codified, data protection failures scale as efficiently as infrastructure itself.

Storage and Data Protection Issues



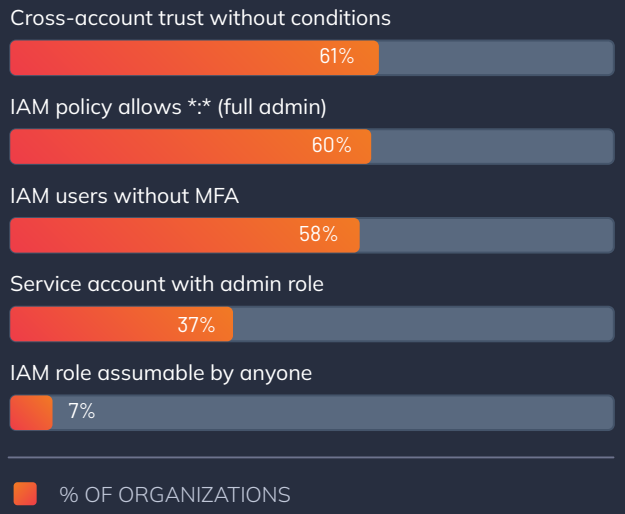
6.3 Identity and Access Management

Identity and access misconfigurations in IaC templates continue to be a leading source of cloud risk, granting excessive permissions that persist across deployments. Our research shows that overly permissive IAM roles and wildcard permissions are frequently deployed via infrastructure templates, increasing the blast radius of credential compromise.

Once deployed, these permissions are automatically propagated across environments, allowing attackers to escalate privileges and move laterally at speed. In IaC-driven environments, a single misconfigured role can grant excessive access across large portions of the cloud estate.

Embedding least-privilege IAM controls into infrastructure definitions is essential to reducing large-scale compromise risk.

IAM Configuration Issues

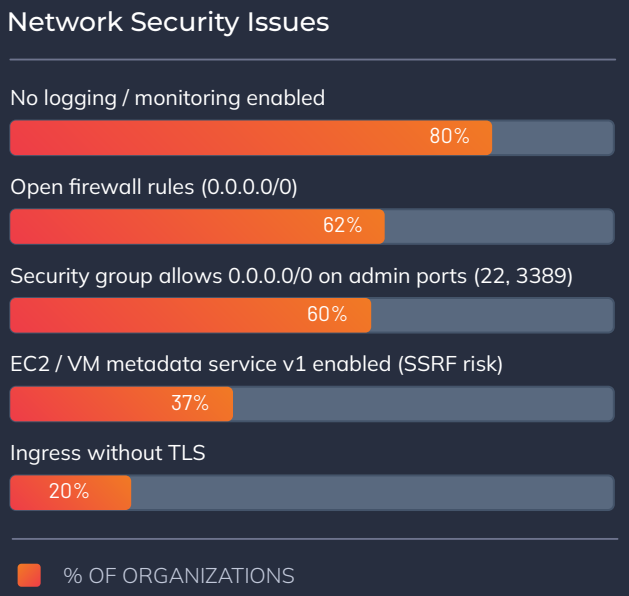


6.4 Network Security

Network controls such as security groups, firewall rules, and routing policies are increasingly managed through IaC. While this enables consistency and efficiency, it also increases the risk of deploying overly permissive network access at scale.

Our research found that open ingress rules and unrestricted management ports are commonly introduced through IaC. **80% of organizations have no logging or monitoring enabled**, and **62% have open firewall rules**, expanding the external attack surface of cloud environments.

Without validation and segmentation, network misconfigurations defined in code can expose entire environments to compromise.



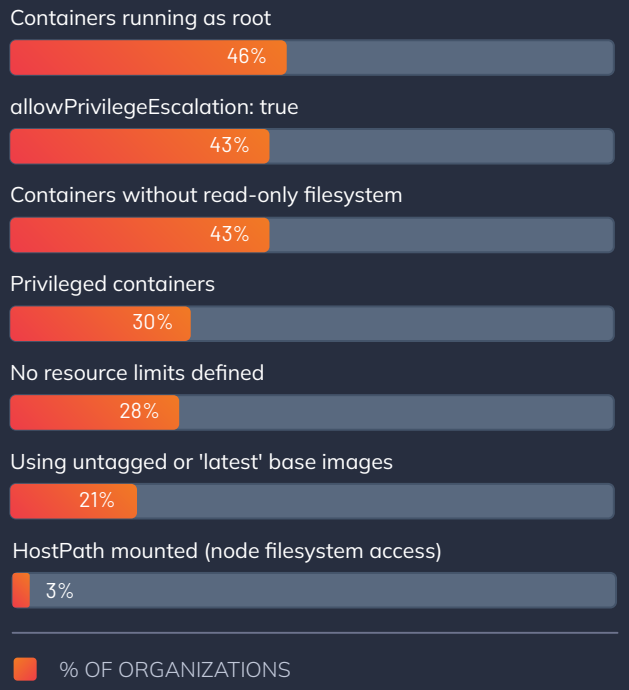
6.5 Container Security in IaC

Containers are frequently deployed and managed through IaC, which connects container security to infrastructure definitions. Our research shows that **77% of organizations retain high or critical container vulnerabilities for more than 90 days**, indicating persistent gaps in remediation and runtime security.

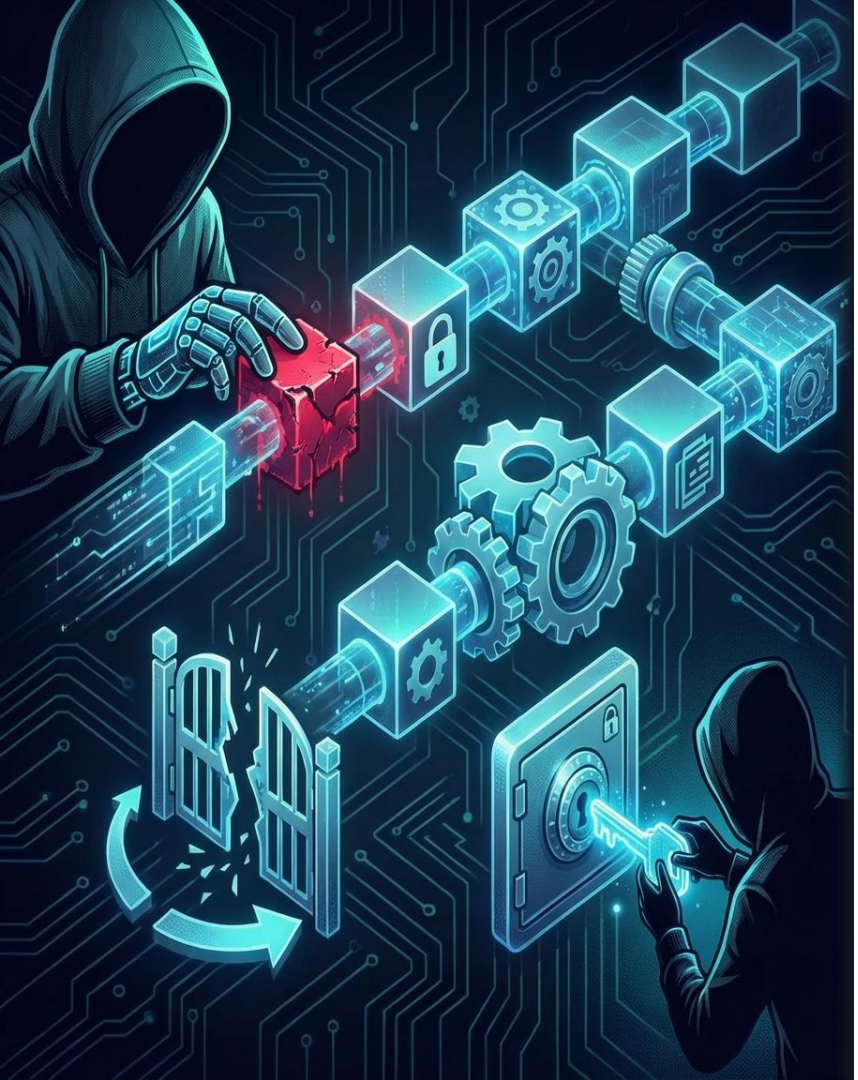
In addition, insecure container configurations are often embedded directly into IaC templates. Once defined, these insecure patterns propagate to every deployment across clusters and environments.

> **The Shift-Left Gap**
>
> Despite widespread IaC adoption (75%), security has not kept pace. With 84% of organizations deploying unencrypted storage and 80% lacking logging, misconfigurations are being codified and replicated at scale. Every terraform apply or kubectl apply propagates these vulnerabilities across environments.

Container Security in IaC



07 Repository and SCM Security



Repository and SCM Security

Source code repositories and source control management (SCM) platforms form the foundation of modern application development. They store proprietary code, infrastructure definitions, and credentials and automation logic that directly impact production environments.

Our research shows that security controls across SCM platforms often lag behind their critical role in the software supply chain, creating opportunities for unauthorized code changes, bypassing code review requirements, or credential theft.

Without consistent governance, validation, and access hygiene, weaknesses in repositories can propagate rapidly across CI/CD pipelines and into production systems.

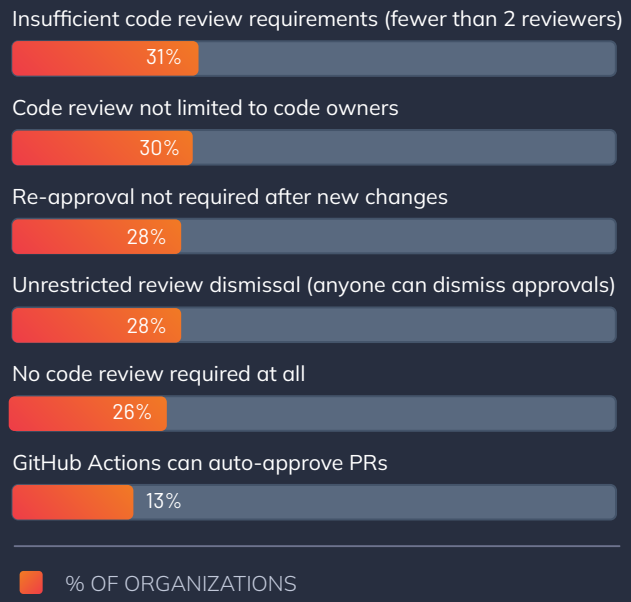
7.1 Code Review and Approval Gaps

Code review and approval processes are the final line of defense against malicious or accidental changes entering production. However, our research indicates that many repositories lack sufficient enforcement of review requirements, with **over 1 in 4 organizations requiring no code reviews at all**. This allows risky changes to be merged with limited to no oversight.

When pull requests are merged without consistent review, attackers can introduce backdoors, malicious dependencies, or credential leaks directly into trusted codebases. In fast-moving development environments, these gaps are often exploited through compromised accounts or social engineering rather than sophisticated exploits.

Without mandatory, enforceable review controls, organizations rely heavily on trust and developer vigilance, which does not scale with modern development velocity.

Code Review Configuration Issues

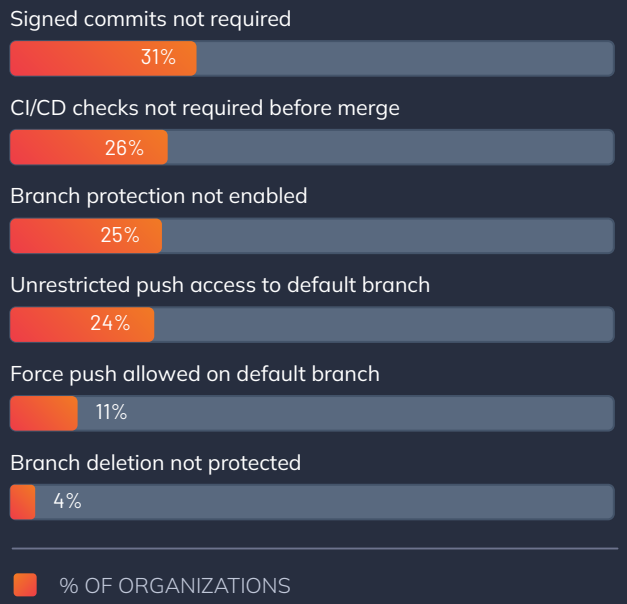


7.2 Branch Protection Weakness

Branch protection rules are critical to prevent unauthorized or unsafe changes to critical branches. Despite their importance, our research shows that branch protection is frequently incomplete or inconsistently applied across repositories. In fact, **25% of organizations studied do not have branch protection enabled**.

Weak or missing branch protection allows changes to bypass testing, review, and policy checks, increasing the likelihood that vulnerable or malicious code reaches production. In organizations with large repository footprints, inconsistent enforcement creates uneven security posture and blind spots attackers can exploit.

Branch Protection Findings



7.3 Access Control and Hygiene

Access to source code repositories is often broader than necessary, with users, service accounts, and integrations retaining permissions long after they are needed, resulting in an expanded attack surface. Our research highlights persistent gaps in access hygiene, including excessive privileges and stale accounts.

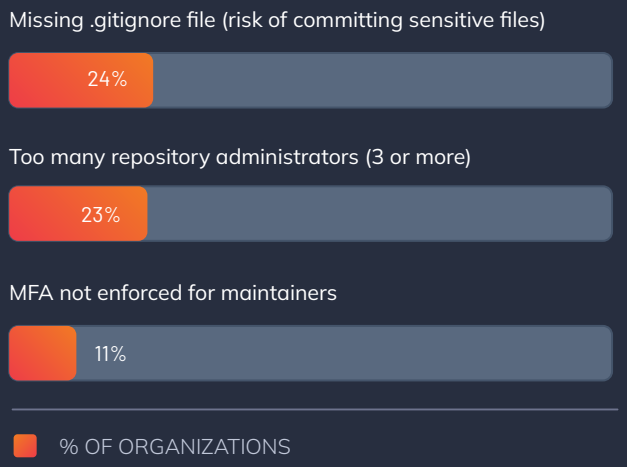
Over-privileged access increases the blast radius of account compromise, while unused or unmanaged accounts create silent entry points into the software supply chain. When combined with limited monitoring and infrequent access reviews, these conditions make it difficult to detect or respond to unauthorized activity.

This is not a novel idea, however it is important to emphasize that maintaining strong access hygiene, such as least-privilege permissions, regular audits, and timely offboarding is critical to reducing SCM risk and protecting application code integrity.

The XZ Utils Wake-Up Call

The 2024 XZ Utils backdoor demonstrated how weak repository controls enable supply chain attacks. The attacker spent two years building trust before introducing malicious code. With 31% of organizations not requiring signed commits and 26% requiring no code review at all, many repositories remain vulnerable to similar patient, sophisticated attacks.

Access Control and Hygiene Issues



08 Key Recommendations

Key Recommendations

This research highlights a growing gap between risk visibility and effective remediation. The recommendations below provide a prioritized roadmap for closing that gap, focusing on the actions that will most immediately reduce exposure and strengthen long-term application security.

8.1 Immediate Actions (0-30 Days)

Secrets Remediation

Identify and rotate all valid secrets across code repositories, commit history, and CI/CD environments. Deploy automated secret scanning with blocking controls to prevent new credentials from being committed or exposed.

Critical Vulnerability Patching

Prioritize remediation of actively exploitable CVSS 10.0 vulnerabilities, including React2Shell (CVE-2025-55182), Log4Shell, and Spring4Shell. Focus first on vulnerabilities reachable from production workloads.

CI/CD Token Permissions

Audit CI/CD authentication tokens and restrict permissions to the minimum required access. Reduce GITHUB_TOKEN scopes, remove legacy default permissions, and limit workflows to verified and trusted actions.

Key Recommendations

8.2 Short-term Initiatives (30-90 Days)

Supply Chain Security Controls

Implement dependency scanning with enforcement for known malicious packages. Audit CI/CD workflows for unpinned actions and transitive dependencies. Require lock files and integrity verification for third-party packages.

Repository Hardening

Enforce branch protection on critical repositories, require signed commits, and mandate MFA for all maintainers. Implement CODEOWNERS to ensure sensitive paths require explicit approval from trusted reviewers.

IaC Security Gates

Integrate Infrastructure as Code scanning into CI/CD pipelines with blocking policies for high-risk misconfigurations, including public storage, open firewall rules, unencrypted resources, and overly permissive IAM roles.

Key Recommendations

8.3 Strategic Improvements (90+ Days)

Container Image Hygiene

Establish standards for approved base images and enforce regular rebuilds and updates. Implement automated container vulnerability scanning in registries and continuously monitor runtime workloads for high and critical risks.

Zero Trust for CI/CD Pipelines

Adopt ephemeral credentials, integrate secrets managers, and eliminate long-lived tokens from pipelines. Enable full audit logging for all CI/CD actions and credential usage.

Continue Risk Monitoring

Deploy runtime security monitoring across applications and infrastructure. Establish baseline behavior, enable comprehensive logging, and configure alerting to detect anomalous activity and emerging threats in real time.

09 Conclusion

Conclusion

The state of application security in 2025–2026 reflects an industry in transition. Organizations have embraced modern development practices such as cloud-native architectures, infrastructure as code, and automated pipelines, but security maturity has not kept pace with the velocity of change.

The numbers are sobering. 78% of organizations ship applications with critical vulnerable dependencies, 31% expose valid secrets in code, and 80% lack adequate logging across their infrastructure. These are not edge cases, they represent the norm across more than a thousand production environments.

Container security paints an equally stark picture. With 77% of organizations carrying unpatched high or critical container vulnerabilities after 90 days, the data reveals a hard truth: organizations that don't patch quickly often don't patch at all. These vulnerabilities are being tolerated instead of remediated.

At the same time, software supply chain attacks have emerged as the defining threat of this era. The progression from the SolarWinds compromise to self-replicating attacks like Shai-Hulud shows how attackers are exploiting trust at scale. The 11% malicious package detection rate demonstrates that even mature organizations remain vulnerable. Incidents like the tj-actions/changed-files compromise further highlight how trusted automation can turn into an attack vector overnight.

The good news is that the most impactful defenses are also the most practical. Enforcing MFA, scanning and rotating secrets, updating dependencies, restricting permissions, pinning GitHub Actions to commit SHAs, and monitoring continuously can dramatically reduce exposure across the software supply chain.

The threat landscape will continue to evolve. Attacks will grow more sophisticated, new vulnerability classes will emerge, and attackers will continue to abuse trust relationships embedded in modern development workflows.

Organizations that thrive will be those that build application security into the DNA of their development process, not as an afterthought, but as a core competency.

About Orca Security

Orca enables organizations to make cloud security a strategic advantage. With the most comprehensive coverage and visibility across multi-cloud environments, the agentless-first Orca Platform unites teams to eliminate complexities, vulnerabilities and risks.

Backed by Temasek, CapitalG, ICONIQ Capital, Redpoint Ventures and others, Orca is trusted by hundreds of organizations, including SAP, Gannett, Autodesk, Unity, Lemonade and Digital Turbine.

To find out more, schedule a
personalized demo of the Orca platform

The path forward requires treating application security not as a checkbox, but as a fundamental component of software quality. Every commit, dependency, and configuration change shapes risk and must be evaluated through a security lens, with clear context, prioritization, and accountability across the software lifecycle."

YOAV ALON
CTO OF ORCA SECURITY

©2026 ORCA SECURITY. ALL RIGHTS RESERVED.