ARMIS®

WHITE PAPER

From Code to Consequence:

Application Security's Role in Cyber Exposure Management

Abstract

Applications have quietly become the most reliable entry point for attackers. As organizations modernize through cloud, microservices, APIs, and AI-assisted development, the application layer now represents the largest and least controlled attack surface in the enterprise. Security teams have made progress securing infrastructure and identities, yet breaches continue to accelerate because adversaries are no longer starting with networks. They are starting with code.

Application security cannot be deemed effective when it exists as an "edge case" tool focused on developer hygiene. It must be treated as a first-class exposure domain within the Cyber Exposure Management (CEM) practice. Organizations that fail to integrate application security into their CEM programs lack operational intelligence into how exploitable code paths connect to real business impact. This gap is increasingly being exploited at scale.

The Current Application Attack Landscape

Attackers follow efficiency.

Over the past decade, defensive investments in the organizational security stack have pushed them away from hardened perimeters and toward softer, more scalable entry points. Applications offer exactly that. In fact, industry data consistently shows that the majority of breaches now originate in the application layer. Verizon's DBIR has reported for several years now that web applications are among the top initial access vectors for breaches, particularly in cloud-first organizations. Meanwhile, software supply chain incidents have grown dramatically, with reports showing year-over-year increases of more than 100% in malicious open source package activity.

The introduction of AI-assisted development has accelerated this risk. Developers are shipping more code than ever, faster than ever, often without understanding the security implications of generated logic or inherited dependencies. Vulnerabilities that once took weeks to introduce can now be replicated across hundreds of services in days. From an attacker's perspective, applications offer reach and leverage. A single vulnerable API can expose sensitive data, enable lateral movement, or provide access to privileged backend systems. Unlike infrastructure vulnerabilities, application flaws are often logic-based, harder to detect with signatures, and easier to exploit once discovered.

Why Hackers Choose Applications as Their Beachhead

Applications sit at the intersection of users, data, and infrastructure.

They are trusted by design and deeply integrated into business workflows. This makes them ideal for attackers seeking persistence and impact. Attackers are not simply hunting for common coding errors. They are targeting authentication flows, authorization gaps, insecure dependencies, exposed secrets, misconfigured infrastructure-as-code templates, and business logic flaws that allow abuse without triggering traditional alerts.

Once access is gained through an application, attackers can move laterally using legitimate APIs, harvest credentials embedded in code or pipelines, and escalate privileges through overly permissive service accounts. From there, the blast radius expands quickly into cloud control planes, data stores, and production systems. Recent breaches illustrate this pattern clearly. In multiple high-profile incidents, attackers exploited a single exposed token or vulnerable dependency to gain access to CI/CD systems, from which they injected malicious code downstream. In others, insecure APIs enabled direct access to customer data without triggering intrusion detection systems because the traffic looked legitimate.

Applications provide attackers with something even more valuable than access, namely context. Understanding how an application works allows attackers to blend in, persist longer, and cause damage that is harder to detect, contextualize, prioritize and mitigate.

The Anatomy of an Application-Based Attack

Most application-driven attacks follow a predictable lifecycle, even if the techniques vary.

It begins with discovery where attackers scan public repositories, package registries, and exposed endpoints looking for vulnerable libraries, misconfigurations, or leaked secrets. AI tooling makes this step trivial and continuous. Next comes initial access. This is often achieved through a vulnerable dependency, a misconfigured API, an insecure IaC template, or hardcoded credentials. Increasingly, these weaknesses are introduced unintentionally through AI-generated code or copied snippets. Once inside, attackers establish persistence. They may modify code, inject malicious dependencies, create backdoor accounts, or abuse CI/CD permissions to ensure continued access even after the original vulnerability is patched.

Using application privileges, attackers laterally pivot into cloud resources, data stores, or identity systems. Because this activity often uses legitimate credentials and APIs, it frequently bypasses traditional security controls completely unnoticed. Finally, impact is realized. This may involve data exfiltration, ransomware deployment, intellectual property theft, or disruption of business operations. At this stage the damage is done, remediation is costly and the incident highly visible. What makes application-based attacks especially dangerous is that many organizations only detect them late in this lifecycle, long after the initial exposure could have been addressed during development.

Why Organizations Are Vulnerable Today

Application security has grown organically, often driven by developer teams adopting point solutions to solve specific problems.

SAST, SCA, secrets scanning, IaC scanning, and container security are frequently owned by different teams, run at different times, and produce overlapping or conflicting results. This fragmentation creates blind spots. Static tools lack runtime context and over-report issues that are not exploitable. Template-based scanners miss vulnerability variants and custom logic flaws. Alerts often stack up and remain unaddressed for long periods of time because they may lack ownership and business relevance, leading to long periods of unaddressed exposure and risk. False positives exacerbate the problem. Industry studies show that developers can spend more time triaging security alerts than fixing real issues, eroding trust between security and engineering. Over time, findings are deprioritized, and exposure accumulates.

CEM programs often compound this issue by focusing primarily on infrastructure, assets, and known vulnerabilities while treating applications as an external input rather than a core exposure domain. Without application context, exposure management remains incomplete. The result is an environment where attackers move faster than defenders, because defenders lack holistic integration, prioritization, and context.

Application Security as a Core CEM Domain

Cyber Exposure Management is fundamentally about understanding what matters, what is exposed, and what should be fixed first.

Applications must be treated as a primary source of exposure, not a downstream concern. To achieve this, application security findings must be tied to real-world exploitability and business impact. Best-in-class organizations integrate application security telemetry directly into their exposure management platforms. This allows them to correlate code-level issues with runtime behavior, asset criticality, and threat intelligence. The result is fewer alerts, clearer priorities, and faster remediation.

Application security also enables proactive exposure reduction. By identifying systemic issues in development pipelines, such as insecure templates, objects or risky AI-generated patterns, organizations can prevent entire classes of vulnerabilities before they reach production.

Best Practices for Including Application Security in CEM

Effective application security within CEM starts with coverage. Organizations must secure the entire software supply chain, including source code, dependencies, infrastructure definitions, secrets, and runtime environments. Gaps in coverage create opportunities for attackers.

Context is equally critical. Findings must be enriched with information about reachability, exploitability, and business relevance. This is the difference between theoretical risk and actionable exposure.

Integration into developer workflows should be a top priority. Security controls that slow delivery or operate outside CI/CD pipelines will be bypassed. AppSec must meet developers where they work and provide guidance that is specific, accurate, and timely.

Ownership and accountability must be automated. Issues should be routed to the right teams with context, prioritization and clear remediation paths, reducing friction and mean time to remediation.

Finally, consolidation matters. Maintaining multiple overlapping tools increases cost and complexity while reducing signal integrity. A unified platform approach enables consistent policy enforcement, shared context, and meaningful reporting to executive stakeholders.

Why a Platform Approach Is A Must

Point solutions cannot keep pace with modern application risk.

They lack the "completeness of vision" needed to detect novel vulnerabilities, particularly those introduced by AI-generated code or complex dependency chains. A unified, AI-powered platform enables deeper detection by identifying vulnerability variants that siloed based technologies miss. It reduces noise by correlating and contextualizing findings across the software lifecycle and eliminating redundant alerts. Most importantly, it connects application risk to enterprise exposure and business outcomes.

Armis Centrix™ for Application Security was built for this reality. By integrating directly into Git, CI/CD pipelines, containers, and runtime environments, it provides continuous visibility across the entire application lifecycle. Proprietary AI models detect known vulnerabilities and previously unseen variants, while contextual intelligence reduces false positives by up to 70 percent.

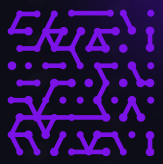
Because it is natively connected to Armis Centrix™ for Vulnerability Prioritization and Remediation, application findings are automatically tied to business context. Security teams can see not just what is vulnerable, but what matters most. The result is faster remediation, lower cost, and stronger alignment between security, development, and the business.

READ MORE

- Press Release →
- Solution Brief →
- Trailer →

Conclusion

As software becomes the primary interface to customers, partners, and critical systems, application security must evolve from fragmented tooling into a core pillar of exposure management. Organizations that fail to make this shift will continue to chase breaches after the fact, while those that succeed will reduce risk by design. Incorporating application security into CEM is not about adding more tools. It is about unifying visibility, intelligence, and action across the software lifecycle. With a platform approach that Armis Centrix™ provides, application security becomes a force multiplier with security being built into the development process thereby enabling secure-by-default software delivery at enterprise scale.



ARMIS®

Go deep into the platform powering the future of security.

Explore Armis Centrix™

Armis, the cyber exposure management & security company, protects the entire attack surface and manages an organization's cyber risk exposure in real time.

In a rapidly evolving, perimeter-less world, Armis ensures that organizations continuously see, protect and manage all critical assets - from the ground to the cloud. Armis secures Fortune 100, 200 and 500 companies as well as national governments, state and local entities to help keep critical infrastructure, economies and society stay safe and secure 24/7.

Armis is a privately held company headquartered in California.

+1 888 452 4011

armis.com