opentext™

WHITE PAPER

State of application security: Trends, challenges, and upcoming threats

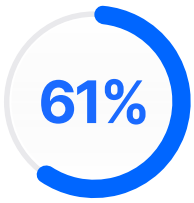
Contents

Application security (AppSec) has become an executive-level priority, as organizations face an increasingly complex threat landscape and rising regulatory scrutiny. Applications now account for a significant portion of security breaches—roughly **25% of all breaches involve application vulnerabilities or stolen credentials**.[1] This is a global concern across web, mobile, and cloud-native apps. High-profile incidents and new regulations (from GDPR fines to US SEC disclosure rules) are reinforcing the need for robust AppSec programs. CISOs are expected to ensure software is secure-by-design, with strategies spanning the entire software development lifecycle (SDLC).

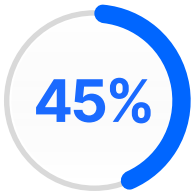
Key trends include the explosion of **AI and machine learning (ML)** within applications and development, an intensified focus on **software supply chain security**, and a surge in **API-driven architectures**—all of which expand the attack surface. **DevSecOps** practices are also maturing: Development teams increasingly drive security tool adoption, emphasizing developer-friendly solutions and integration into CI/CD pipelines. Traditional AppSec technologies like SAST, DAST, and SCA remain cornerstones, but their roles are evolving to keep pace with modern development and threats. Security leaders are also grappling with tool sprawl and skills gaps, leading nearly half of enterprises to plan **consolidation of AppSec tools for efficiency**.

Overall, the **current AppSec landscape is defined by complexity and opportunity**. Attackers are exploiting open-source flaws and API weaknesses with greater frequency, even leveraging AI to discover and weaponize vulnerabilities. Yet, defenders also have new tools, from AI-driven anomaly detection to automated code scanning, to bolster their posture. With global regulations mandating stronger security governance (e.g., **mandatory breach disclosures and software bill of materials (SBOM)** reporting), executive support for AppSec has never been higher. This paper provides a comprehensive look at the state of application security, the core challenges and emerging trends, and an outlook into 2026 and beyond. It offers AppSec leaders and CISOs a global perspective on how to safeguard enterprise applications across web, mobile, and cloud environments, balancing innovation with security and compliance.

1 AIMultiple, *20+ Application Security Statistics & Trends in 2025,* July 10, 2025



of organizations now leverage AI/ML for threat detection in applications and APIs.



Gartner predicts 45% of organizations will have experienced a software supply chain attack by 2025.

Key trends

Generative AI in applications

Organizations are embedding generative AI (GenAI) and ML models into products and workflows at scale. More than **33% of enterprises report using GenAI in production applications**.[2] This introduces new vulnerabilities—for example, prompt injection attacks on AI models and exposure of sensitive data via AI APIs. Security teams are only beginning to understand how **LLM usage expands the application attack surface**. Conversely, defenders are adopting AI for defense; **61% of organizations now leverage AI/ML for threat detection**[3] in applications and APIs. The dual use of AI is redefining AppSec, demanding vigilance against AI-specific threats and creative use of AI to enhance security monitoring.

Software supply chain transparency

After breaches like SolarWinds and Log4j, attackers continue to target upstream libraries and build processes. Governments worldwide responded with regulations for transparency. For example, US federal agencies and even the FDA now require software vendors to provide SBOMs (software bills of materials), and the **EU's Cyber Resilience Act (effective late 2024) mandates SBOMs for digital products**. Australia's cyber guidelines similarly recommend SBOM usage. As a result, enterprises are pressuring suppliers for open-source component lists and vulnerability data. Gartner predicts **45% of organizations will have experienced a software supply chain attack by 2025**,[4] underscoring why software composition analysis and supply chain risk management are top of mind. Indeed, modern applications are largely built, not written: A Synopsys study found **97% of apps contain open-source components**,[5] with transitive dependencies (indirect open-source libraries) making up 64% of the components. This ubiquity of open source has driven demand for greater supply chain visibility and secure dependency management as a strategic priority.

API explosion and security focus

API use is surging—driven by microservices, mobile apps, and third-party integrations—with one report citing **167% growth year-over-year**.[6] Security maturity lags: 95% of organizations reported API security issues in production, and 23% experienced breaches.[7] Common issues include broken authentication, excessive data exposure, and missing rate limits. In 2025, API security became a C-level priority at 46% of companies, spurring investment in testing, monitoring, and protection. Yet only ~7.5% have mature API testing programs.[8] **Cloud-native**

2 Forrester, *The State Of Application Security, 2025: Yes, AI Just Made It Harder To Do This Right,* May 15, 2025
3 Fortinet, *2025 Web Application Security Report,* 2025
4 Snyk, *Software Supply Chain Security White Paper,* [n.d.]
5 Black Duck, *Six takeaways from the 2025 "Open Source Security and Risk Analysis" report,* February 25, 2025
6 Exclusive Networks (Salt Security), *Salt Security – API Security Report 2024,* July 9, 2024
7 Ibid
8 Ibid

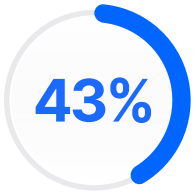
and container security ("Shift Left"): The shift to cloud-native architectures, such as containerized microservices, serverless functions, and Infrastructure as Code, is accelerating. Misconfigured cloud storage or unpatched container images can lead to serious breaches. In 2025, **container and Kubernetes security** has solidified as a key AppSec trend, with businesses embedding security checks into DevOps pipelines ("shift-left" security) to scan images and IaC templates before deployment. DevSecOps practices have organizations scanning code, container artifacts, and cloud configs early and often, to prevent vulnerable software from ever reaching production.

In 2025, API security became a C-level priority at 46% of companies, spurring investment in testing, monitoring, and protection. Yet only ~7.5% have mature API testing programs.

Developer-centric security and DevSecOps

A noteworthy shift is the empowerment of development teams in security decision-making. **Sixty-two percent of security leaders say development teams are now the final decision makers for AppSec tool purchases**.[9] **Developer experience is prioritized over sheer feature count:** If a scanning tool is too slow or noisy, developers won't use it, rendering its findings moot. Successful AppSec programs embraced this ethos by embedding security controls in IDEs, CI/CD pipelines, and version control, and providing guardrails (secure coding training, linting, automated scans) rather than gatekeeping. The trend has also led to growing use of **"self-service" security tooling** (APIs for security testing, automated fix pull-requests, etc.) and more cross-functional collaboration.

Tool consolidation and platform approaches



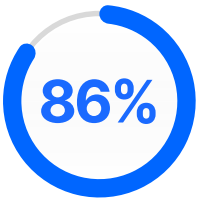
of organizations plan to consolidate tools to cut complexity and improve integration

Enterprises have long relied on numerous point solutions (SAST, DAST, SCA, WAFs, container scanners), creating inefficiencies, siloed data, and high costs. A 2025 survey found that 43% of organizations plan to consolidate tools to cut complexity and improve integration.[10] The trend is toward unified AppSec platforms covering code, open source, runtime protection, and API security in one suite, offering better visibility and streamlined workflows. Automation is also increasing, linking scanners to ticketing and CI pipelines for faster remediation, to maximize efficiency with limited security staff. **Many AppSec leaders view consolidation and smarter tooling as the answer to both resource constraints and the need for faster remediation.**

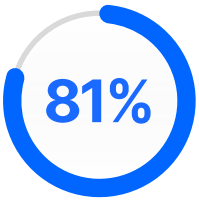
These key trends highlight a landscape where innovation (AI, cloud-native tech) is running ahead of security in some areas, even as security teams innovate with new approaches (DevSecOps, unified platforms) to catch up.

9 Forrester, *The State Of Application Security, 2025: Yes, AI Just Made It Harder To Do This Right*, May 15, 2025
10 Fortinet, *2025 Web Application Security Report*

The 2025 OSSRA report found:



of audited apps contained open-source vulnerabilities



included at least one high/critical open-source vulnerability.

Core challenges and threat landscape

Despite heightened awareness and investment, the threat landscape is both vast and rapidly evolving, straining traditional defenses. Core challenges and threats include:

Expanding attack surface and vulnerability deluge

Modern applications combine legacy code, microservices, open-source libraries, and third-party APIs—expanding the attack surface. In the past year, 26,447 vulnerabilities were disclosed,[11] with over 75% of applications having at least one flaw and 61% containing a high-severity issue outside the OWASP Top 10,[12] pointing to the breadth of issues beyond the "usual suspects." Security teams face challenges prioritizing and patching amid continuous deployments, and unpatched vulnerabilities remain a leading cause of breaches. In 2025, about 60% of breaches[13] involved known, unpatched flaws. **Timely patching is critical, but operational constraints and fear of system disruption often delay fixes.**

Open source and supply chain risks

The ubiquity of open-source software in applications introduces significant risk if not managed. Nearly all commercial apps use open-source components, many with known vulnerabilities or license risks.

The 2025 OSSRA report found 86% of audited apps contained open-source vulnerabilities, and 81% included at least one high/critical open-source vulnerability.

Compounding the issue, 91% of applications contain outdated open-source components[14]—most more than 10 versions behind. This threat backdrop makes it challenging for enterprises to **track their software supply chain**—many lack an up-to-date inventory of which components (and versions) are in their code, especially transitive dependencies that are "hidden" inside other packages. Visibility and governance over open-source usage remain a core challenge, requiring organizations to adopt SCA tools, maintain SBOMs, and stay alert to third-party advisories.

API and microservice vulnerabilities

As companies pivot to API-first architectures, APIs have become prime targets, with threats like broken object level authorization (BOLA), where attackers manipulate object IDs in an API call to access data they shouldn't. Other common API issues include lack of input validation on JSON/XML payloads, overly permissive CORS configurations, and inadequate authentication on internal microservice APIs. **Business logic flaws** are especially challenging—

11 AIMultiple, *20+ Application Security Statistics & Trends in 2025*, July 10, 2025
12 Ibid
13 Ibid
14 Black Duck, *Six takeaways from the 2025 "Open Source Security and Risk Analysis" report*, February 25, 2025

these are not simple coding bugs, but design issues (like an e-commerce API not limiting quantity in an order, enabling abuse). Traditional scanners have difficulty catching these, meaning they often slip through to production. Additionally, bots and automated scripts hammer APIs with credential stuffing and DDoS attacks; in fact, **DDoS was ranked the top bot-driven threat to web apps/APIs by many security professionals.**[15] Yet, many organizations lack full **visibility into their API inventory**—a significant number report struggling to even identify all active APIs and applications in their environment.

Cloud misconfigurations and secrets management

The move to cloud and DevOps has shifted some attack focus to configuration issues rather than code flaws. A single misconfiguration in cloud infrastructure can have dire consequences—for instance, an open S3 bucket or an overly broad IAM role can lead to massive data leaks or privilege escalation. Similarly, hardcoded secrets or tokens in app code (or CI pipelines) are a major risk; attackers actively scan public repos for leaked API keys. Many breaches in recent years trace back to mismanaged cloud app configs or exposed credentials. The challenge for AppSec teams is working with cloud security and DevOps teams to ensure secure defaults and automated checks.

Sophisticated and automated attacks

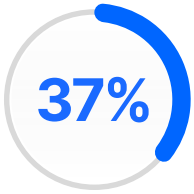
We see **ransomware gangs exploiting web app flaws** to gain footholds and **botnets systematically probing web applications** for common vulnerabilities. Attackers are also using **AI-driven tools to find and exploit vulnerabilities faster**.[16] For example, machine learning can help malware adapt to defenses or help attackers fuzz software to discover zero-days more efficiently. There is concern that generative AI could enable less-skilled adversaries to produce sophisticated exploits or phishing content at scale. These tools mean attackers can quickly weaponize new vulnerabilities and often coordinate attacks, underscoring the need for **continuous testing and monitoring**.

Regulatory and compliance pressures

Worldwide privacy laws (GDPR, CCPA) and industry standards (PCI DSS 4.0, HIPAA) demand robust protection, with GDPR fines exceeding $1.2 billion in a single year (2021) and that trend continues upward.[17] New SEC rules require public companies to disclose material cyber incidents within four business days and report annually on cyber risk governance. The positive side is that regulation has pushed AppSec higher on the agenda, but achieving and proving compliance remains a non-trivial challenge that requires process maturity.

15 Fortinet, *2025 Web Application Security Report,* [n.d.]
16 Dark Reading (Jai Vijayan), *6 AI-Related Security Trends to Watch in 2025*, December 31, 2024
17 AIMultiple, *20+ Application Security Statistics & Trends in 2025,* July 10, 2025

The threat landscape is marked by a high volume of vulnerabilities, automated attacks, and new frontiers like APIs and open-source supply chain risks. Organizations must reduce their internal risk (secure coding, proper configuration, prompt patching) and defend against external threats (skilled adversaries and bots)—all under the watchful eye of regulators and customers.



of organizations now perform SCA during development to catch vulnerable open-source use before it progresses.

The evolving role of SAST, DAST, and SCA technologies

Static Application Security Testing (SAST), Dynamic Application Security Testing (DAST), and Software Composition Analysis (SCA) have long been the pillars of application security tooling. These technologies continued to mature and adapt, often used in concert as part of an integrated DevSecOps toolchain. Below is how each is evolving in the enterprise context:

Static Application Security Testing (SAST)

SAST tools analyze source code (or compiled bytecode) for vulnerabilities *without executing the program*. The role of SAST is shifting **earlier in the development lifecycle** ("shift-left") so issues are caught at commit or build time. SAST now has to rise to new challenges, as developers using tools like GitHub Copilot can unintentionally introduce vulnerabilities or licensing issues into code. SAST detects these AI-introduced flaws. Additionally, SAST is expanding to cover more languages and frameworks (for example, infrastructure-as-code templates, mobile app code, and even binary scanning when source is not fully available). It is becoming a continuous, invisible part of coding, often running in the background on every pull request.

Dynamic Application Security Testing (DAST)

DAST solutions test running applications by simulating attacks and analyzing responses. DAST tools have evolved to be more **API-aware and automated**. Modern DAST can import API definitions (Swagger/OpenAPI) to systematically test REST and SOAP endpoints, which is vital given APIs are a growing attack surface. **Interactive DAST** and cloud-based DAST services can run continuously against staging environments or even production to detect changes or emerging vulnerabilities. Modern DAST engines often leverage headless browsers to better analyze complex single-page applications and have improved in crawling modern web interfaces (e.g., SPAs, which traditional scanners struggled with). They also use **machine learning to reduce false positives**, differentiating intentional app behavior from genuine security issues. Despite advancements, DAST faces challenges like deep logic testing, which is why many organizations complement DAST with manual pen-testing for business logic flaws. Still, DAST's ability to find exploitable conditions (like an open admin interface or a XSS vulnerability) in a running app makes it indispensable.

Software Composition Analysis (SCA)

SCA tools scan applications for open-source components and third-party libraries, identifying known vulnerabilities) and license compliance issues. SCA has arguably become as crucial as SAST/DAST due to the surge in supply chain attacks. Modern SCA solutions can automatically produce an **SBOM** for each build, flag any component with known CVEs, and even suggest or automate upgrading to safer versions. SCA is also used to enforce corporate open-source policies. An interesting trend is integrating SCA early in development: Nearly **37% of organizations now perform SCA during development to catch vulnerable open-source use before it progresses**. This reflects a shift-left for SCA akin to SAST's. Additionally, the importance of SCA is underscored by

regulatory pushes. With requirements for SBOMs and proof of vulnerability management, SCA reports are becoming deliverables in contracts and compliance audits. Some SCA tools now tie into repository management, automatically checking pull requests that introduce new dependencies. The evolution of SCA is also towards **remediation support**—not just flagging a library as vulnerable but helping developers update it.

Moreover, the industry is moving toward **unified AST platforms** where SAST, DAST, SCA, and possibly IAST results are correlated for a single view of application risk. This helps eliminate duplicate findings and allows teams to prioritize more effectively (e.g., seeing that a vulnerability flagged by SAST is in a library also flagged by SCA as vulnerable and is exploitable via DAST). **Such correlation and context are improving remediation efficiency**.

The goal is a seamless DevSecOps workflow, where code is continuously scanned (statically and dynamically), components are continuously inventoried for risks, and developers get immediate feedback to fix issues long before applications are in front of users.

These tools are increasingly augmented by **AI/ML** features as well. For example, some SAST vendors introduced AI to better detect patterns or suggest fixes, and some DAST solutions use ML to distinguish normal vs. attack traffic.

Emerging areas in application security

The fast pace of technology means new frontiers in AppSec are constantly emerging. In 2025, several areas gained prominence due to changes in how we build software and evolving external requirements. Each of these is shaping the future of application security:

AI/ML in application security

Artificial intelligence and machine learning are double-edged swords in the realm of application security. On one hand, AI/ML are *being embedded into applications* themselves at an unprecedented rate, and on the other, they are *powerful tools for both attackers and defenders*.

Within applications, **AI/ML components introduce new risks** that AppSec teams must account for. For example, many apps now incorporate AI models (like recommendation engines or LLM-based chatbots) or call out to AI services via APIs. This creates novel attack vectors, such as **model poisoning, prompt injection, and data leakage through AI APIs.** A noteworthy concern is that organizations have rushed to deploy generative AI features without fully understanding the security implications. **Forrester reports that one-third of organizations are using generative AI in production apps**[18] potentially without robust safeguards. An embedded AI model might inadvertently expose sensitive training data or be manipulated by crafted inputs. Furthermore, AI features often rely on extensive API integrations and each of those API calls (to an LLM service or ML microservice) expands the surface for API abuse.

On the development side, AI-assisted coding tools like GitHub Copilot, Amazon CodeWhisperer, have gone mainstream. A survey of 1,700 IT pros found **81% are using GenAI to assist with coding and software development.**[19] While these tools boost productivity, they can also generate insecure code or copy vulnerable code patterns from training data. They might even introduce legal risk by reproducing licensed code. Security leaders are now tasked with mitigating these risks by implementing **policies around AI-assisted code use**, scanning AI-generated code rigorously, and using tools that can detect secrets or known vulnerable snippets potentially inserted by AI. From a defensive standpoint, **AI/ML is becoming a force multiplier for security operations**. In AppSec, this means automated vulnerability discovery, anomaly detection in application behavior, and intelligent triaging. For example, some advanced scanning tools now use ML to reduce false positives or to pattern-match code against known vulnerability fingerprints more effectively. In runtime protection, AI-driven systems analyze traffic to distinguish legitimate users from bots or detect subtle attacks that signature-based systems might miss. An industry survey found **61% of organizations are leveraging AI for threat detection in applications**,[20] for instance, using ML models to detect anomalies in API usage that could indicate an attacker probing for weaknesses. Additionally, AI can help correlate signals faster than a human might.

18 Forrester, *The State Of Application Security, 2025: Yes, AI Just Made It Harder To Do This Right,* May 15, 2025
19 Dark Reading (Jai Vijayan), *6 AI-Related Security Trends to Watch in 2025*, December 31, 2024
20 Fortinet, *2025 Web Application Security Report*

Attackers, however, are equally keen on AI. **Threat actors are weaponizing AI to automate and enhance attacks**. One concern is **AI-generated malware and exploits**: tools that use machine learning to mutate malware to evade defenses, or to automatically find exploitable conditions in software (through AI-driven fuzzing that doesn't require source code). Ransomware gangs are prototyping AI that can identify which data to encrypt for maximum impact or automatically bypass certain security tools. Phishing campaigns are using AI to create highly convincing deepfakes and personalized phishing messages at scale, lowering the barrier for large-scale social engineering.

One emerging concept is **"Shadow AI,"** the ungoverned use of AI tools by employees or developers without oversight. This can lead to sensitive code or data being unintentionally shared with external AI services (as happened when some employees pasted proprietary code into public chatbots). CISOs are growing concerned about this and the **data exposure risks from unsanctioned AI use**. There's also anticipation of new regulations (like the EU AI Act) that will place compliance requirements on AI systems which could impact how AI features in applications are secured and audited.

Moving forward, we expect "AI security" to become its own discipline intersecting with AppSec—including practices like adversarial testing of ML models, validation of AI outputs, and continuous monitoring for AI-driven anomalies. For now, security leaders should embrace AI carefully.

Software supply chain security

After several wake-up calls in recent years, **software supply chain security** has solidified as a top-tier priority. This area covers the security of all components that go into software (open-source libraries, third-party services, build tools) and the processes by which software is developed and delivered.

As noted earlier, open-source components are ubiquitous— **97% of applications contain open-source software**,[21] and these components often come with known vulnerabilities or can be compromised at the source. Attackers have various techniques to exploit this:

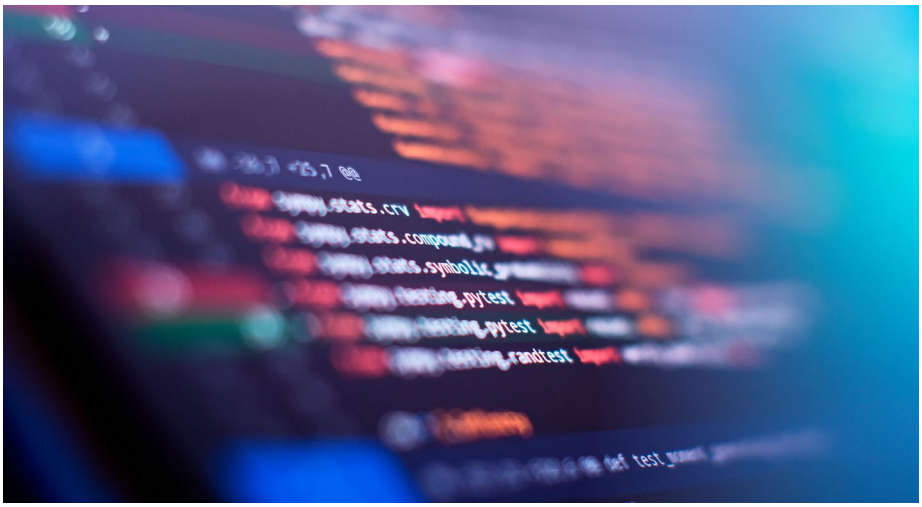
- **Typosquatting and repository attacks:** Planting malicious packages in public repositories with names similar to popular ones, hoping developers install them by mistake. This has happened multiple times, leading to malware inside corporate networks.
- **Hijacking maintainer accounts or build infrastructure:** We've seen instances where attackers took over the account of an open-source project maintainer or CI system, then pushed a tainted update (as in the SolarWinds 2020 incident). Organizations consuming these updates unknowingly brought in malware.
- **Exploiting vulnerabilities in dependencies:** Often companies don't know they are using a vulnerable library until a major CVE (like Log4Shell in Log4j) surfaces. The scramble to find and fix it across dozens of applications highlights the need for better inventory and patching mechanisms.

21 Black Duck, *Six takeaways from the 2025 "Open Source Security and Risk Analysis" report,* February 25, 2025

Gartner predicts that by the end of 2025, 60% of organizations will use cybersecurity risk as a key determinant in third-party business dealings.

Governments and industry groups are enforcing more rigorous supply chain security measures. This push for transparency means that in procurements and partnerships, companies increasingly ask for evidence of secure software practices. In fact, **Gartner predicts that by the end of 2025, 60% of organizations will use cybersecurity risk as a key determinant in third-party business dealings**.[22] Practically, this means enterprises must vet the security of software vendors, demand vulnerability disclosure practices, and include security criteria in contracts Key developments and practices in software supply chain security include:

- **Comprehensive SCA and SBOMs:** Companies are leveraging SCA tools to generate SBOMs for their applications. An SBOM is essentially an ingredient list of software. By having SBOMs, organizations can quickly answer "are we affected by the latest Struts/Log4j/XYZ vulnerability?" and also communicate to customers what's inside their software. Many are sharing SBOMs with customers or using them internally for risk assessments.
- **Vulnerability management and patching processes:** Given that **91% of applications have outdated components**, a big focus is establishing processes to update dependencies continuously. The idea is to shrink that window of exposure for known flaws. DevSecOps pipelines are being configured to fail builds if a critical vulnerability is present in a component (policy-based controls).



- **Security of build and deployment pipelines:** The supply chain extends to CI/CD. Attacks on CI pipelines can insert malicious code during the build (. More organizations are hardening these environments: using principles of zero trust, signing artifacts (so that any tampering with binaries would be detectable), and implementing least privilege for pipeline tools. The concept of **"pipeline integrity"** is on the rise, with frameworks like SLSA (supply-chain levels for software artifacts) providing guidelines to achieve tamper-resistant build processes. **Third-party risk assessments:** There is now often

22 TechRepublic, *Gartner reveals 8 cybersecurity predictions for the next 4 years,* June 22, 2022

The mantra for software supply chain security in 2025 is *"Know your code"*—what's in it and where it comes from, and keep it updated and secure. Still, many companies are early in building these capabilities, and attackers will look for the weakest links. By 2026 and beyond, supply chain security will mature into standardized practice and possibly even see more *regulatory requirements imposing liability on software providers* for insecure components.

a dedicated function or team to evaluate the security of third-party software and services. This can involve questionnaires, reviewing pen-test reports or ISO certifications, and requiring certain practices (e.g. annual code audits or secure development training). For open-source projects that are critical (think OpenSSL, etc.), some companies even contribute back resources or funding to ensure those components stay secure and maintained.

- **Runtime protections for supply chain attacks:** Because not all supply chain attacks can be prevented (zero-day vulnerabilities or a trusted vendor unknowingly shipping a backdoor), detection is vital. Organizations are deploying monitoring at runtime to catch unusual behavior, for example, if a library suddenly starts making outbound network calls it never did before, or if an application tries to modify files it shouldn't. Endpoint and workload protection platforms can sometimes detect these anomalies.
- **Collaborative industry initiatives:** Industry and government collaborations like the Open-Source Security Foundation (OpenSSF) are working on securing the ecosystem (initiatives like publishing best practices and tools for open-source maintainers). The US and EU are also funding efforts to audit widely used OSS projects.

API security

API security has emerged from the shadow of "traditional" web application security to become a specialization of its own. As businesses expose more APIs (for mobile apps, partner integrations, microservice communication, etc.), adversaries have followed the data—attacking APIs to exfiltrate information, abuse functionality, or compromise systems. APIs are *both* a critical enabler of digital business and a significant source of security risk that must be addressed head-on.

As we've stated earlier, insecure APIs are not a hypothetical threat but a commonplace reality. Several factors contribute to the challenge:

- **Rapid proliferation:** Development teams deploy APIs at a rapid pace, sometimes without the knowledge of central IT/security. With microservices, one application might comprise dozens of internal and external APIs. Keeping an up-to-date inventory is difficult.
- **Complex access patterns:** Unlike a web app with a clear user interface, an API might be consumed by many clients (mobile apps, IoT devices, third parties). Ensuring proper auth and access control for each use case is complex. Many API breaches involve mistakes in authentication/authorization logic, for instance, not verifying the caller's permissions on a resource (leading to broken object level authorization issues, OWASP API Top 10 API1:2023).
- **Data exposure:** APIs often expose raw data (in JSON/XML) and rely on the client to filter or display it appropriately. This can lead to excessive data exposure where APIs return more data than necessary (maybe including sensitive fields), trusting that the client will hide it—a trust that attackers can exploit by directly calling the API.

- **Lack of visibility and testing:** Traditional testing tools were web-page centric and often missed APIs, especially if those APIs weren't documented or linked. Many security teams found that their DAST scanners did not automatically test all API endpoints, and developers weren't including APIs in their regular testing either. As noted, only about **7.5% of organizations have dedicated API testing and modeling programs**,[23] meaning the majority rely on ad-hoc or manual efforts, which leave gaps.

In response, **API security tooling and practices have boomed:**

- **API discovery:** One of the foundational steps is discovering all APIs in use. New tools can sniff network traffic or integrate with API gateways to automatically catalog APIs, endpoints, parameters, and usage patterns. This helps identify "rogue" or undocumented APIs. Some enterprises also mandate that all new APIs be registered in a central catalog and have an OpenAPI/Swagger spec, which can be used by security tools.
- **Security testing for APIs:** Penetration testing is being extended to focus on APIs. Automated API testing tools can take an API spec and generate security tests (fuzzing inputs, testing auth bypasses, etc.). We've also seen uptick in **schema analysis**—reviewing API schemas for risky patterns. OWASP updated its API Security Top 10 in 2023, which gives teams an up-to-date blueprint of issues to test for.
- **Runtime protection and monitoring:** Many organizations deployed API gateways or WAFs with API-specific features. These act as chokepoints to enforce authentication, rate limiting, and to detect common attacks. API monitoring is equally crucial: analyzing logs to spot abnormal behavior, such as one user suddenly requesting thousands of records (potential data exfiltration) or a spike in 401/403 errors (which might indicate an attacker probing for valid endpoints). Advanced solutions using AI help profile normal API usage and flag anomalies, which is essential for catching logic abuses that signature-based tools might miss.
- **Shift-left API security:** Development teams are being encouraged to include security earlier when building APIs. This includes code-level controls (using secure frameworks that handle auth for you), performing threat modeling specifically for APIs (thinking through how each endpoint could be misused), and using linters/static analysis tuned for API issues. Some organizations have also started using **"API security checklists"** during code reviews, which align with things like OWASP API Top 10 to ensure common mistakes are caught.
- **C-level awareness and strategy:** Importantly, **API security has reached the boardroom**. With 46% of companies saying API security is discussed at the C-level now,[24] we see formal API security programs being chartered. This might entail appointing an API security lead, building cross-functional teams to govern APIs, and setting KPIs such as reducing the number of incidents or time to remediate API vulnerabilities. Executive awareness also means budget: Spending on API security solutions (testing tools, management platforms) has increased.

23 Exclusive Networks (Salt Security), *Salt Security – API Security Report 2024,* July 9, 2024
24 Ibid

The threat landscape for APIs includes notable incidents like the compromise of APIs at financial institutions and critical vulnerabilities discovered in popular API frameworks. Also noteworthy is the interaction of API security with **OAuth/OIDC and third-party integrations**—configuration errors in how tokens are validated or how scopes are enforced have caused some security lapses. Attackers also exploit development/test APIs that are inadvertently left exposed (for example, a "debug" API that was deployed and forgotten).

API security will only grow in importance, as the number of APIs is expected to keep rising.Organizations will likely integrate API security checks into every stage: design (using secure design patterns), development (linting and SAST for API issues), testing (specialized DAST/fuzzing for APIs), and operations (continuous monitoring and incident response playbooks for API abuse). The guiding principle must be **zero trust for APIs**. Never assume an API is safe by obscurity or that internal APIs won't be targeted. Every API call should be authenticated, authorized, validated, and logged.

Regulatory impacts and compliance landscape

Regulation has become a driving force in application security practices globally. This includes data protection laws, industry-specific regulations, and new cybersecurity-focused rules from governments. The "compliance landscape" is now a key consideration for AppSec leaders, as non-compliance can result in heavy fines, legal liability, and reputational damage.

Global data protection and privacy laws

Since GDPR in 2018, many regions—including CCPA/CPRA (California), LGPD (Brazil), POPIA (South Africa), and PDPA (Singapore)—have enacted similar privacy laws requiring data protection by design and default. For AppSec, this means strong access controls, encryption in transit and at rest, and timely breach notification. GDPR fines of up to 4% of global turnover have driven investment in prevention, with more applications adopting client-side encryption and tokenization. Privacy laws also reinforce secure development practices through DPIAs, embedding privacy and security reviews into the SDLC.

The regulatory and compliance landscape drives improved AppSec and poses an additional challenge to manage. It compels organizations to:

Build security in (to meet "secure by design" expectations).

Prove it with documentation and reports (SBOMs, compliance reports, etc.).

Respond swiftly and transparently to incidents (due to breach notification rules).

Keep up with an ever-evolving patchwork of laws across jurisdictions.

Cybersecurity disclosure and governance (SEC rules in the US)

A landmark development is the US SEC's cybersecurity disclosure rules adopted in 2023 (effective 2024-2025). Public companies are now required to:

- **Report material cybersecurity incidents within 4 business days** via an 8-K filing (with some allowances for law enforcement delays).
- **Annually disclose their cybersecurity risk management and governance** in 10-K/20-F, including the board's oversight of cyber risk and management's processes.

This raises AppSec stakes—boards demand assurance of strong controls, and CISOs work closely with legal teams to define "material" incidents, ensure rapid escalation, and prepare accurate filings. Even amid debate, transparency and potential executive liability drive stronger AppSec programs, from secure coding to continuous testing, to reduce the risk of reportable breaches.

Sector-specific regulations and standards

Different industries face their own rules that affect AppSec:

- **Financial services:** In the EU, the **Digital Operational Resilience Act (DORA)** came into force in 2025, requiring banks and financial entities to ensure the security of network and information systems, including rigorous testing. In the US, financial regulators also expect strong application controls. Many banks adhere to PCI DSS if they handle payments—PCI DSS v4.0 has enhanced requirements for software security, such as more frequent code reviews and vulnerability scans.
- **Healthcare:** In the US, HIPAA and the HITECH Act mandate protection of electronic health data. For app dev, that means audit logs, user authentication, and encryption must be built in. There is also increasing oversight on medical device software.
- **Critical infrastructure:** Governments have expanded the definition of critical infrastructure to include IT and software providers. For example, the EU's NIS2 Directive (effective 2024) imposes security and incident reporting requirements on a wide range of sectors, including digital services and software companies. Many countries (Australia, India, etc.) require timely breach reporting for critical service providers as well. This pushes AppSec teams to implement **continuous monitoring and incident detection** so they can meet tight reporting timelines.

Product security and liability

The EU is leading with legislation like the **Cyber Resilience Act (CRA)**, which, as mentioned, mandates that products with digital elements (software, IoT devices, etc.) are built securely and come with vulnerability disclosure policies and SBOMs. Similarly, there's discussion in the US about imposing liability on software makers for egregious security flaws (shifting some responsibility from users to producers). This signals a future where insecure software could mean legal consequences, not just technical ones. Forward-looking companies are already adopting secure development frameworks (like NIST SSDF), documenting their security controls, and preparing to furnish this documentation to regulators or customers.

International and cross-border challenges

With so many regional laws, global enterprises face the challenge of complying everywhere. Data residency and sovereignty laws might affect how applications are designed (e.g., using regional shards so personal data doesn't leave a country). Encryption mandates or bans in certain jurisdictions could complicate things. A practical example: China's Cybersecurity Law and ensuing regulations require companies to undergo security assessments if they export data abroad, pushing firms to localize data storage and double down on security to pass audits. Similarly, Russia's data laws and Middle Eastern countries' regulations all impose various security expectations.

Compliance as a baseline, not a goal

One common understanding is that compliance does not equal security, but it's the minimum bar. Many organizations use frameworks like **ISO/IEC 27001, NIST Cybersecurity Framework, SOC2**, or **OWASP SAMM** to structure their security programs and demonstrate due diligence. These frameworks indirectly push better AppSec. Being compliant often forces organizations to adopt certain AppSec practices (regular scans, training developers, access controls, etc.). The trick is doing so in a way that actually reduces risk, not just checks a box. For instance, the better organizations have transformed requirements (like "do an annual app pen-test") into meaningful practices (like a continuous testing program combined with bug bounty, far exceeding the bare minimum).

For AppSec leaders, staying ahead means collaborating with legal/compliance teams, tracking emerging laws (like AI regulations, IoT security laws), and baking compliance requirements into the development lifecycle early (so that being compliant is a natural outcome of the way software is built). The trend is clear: **security is becoming a legal requirement, not just a technical nice-to-have.**

Forecast: Looking ahead to 2026 and beyond

Based on current trends, expert predictions, and emerging technologies, the following developments are expected soon:

1 AI everywhere: Smarter apps and smarter attacks

AI will be deeply embedded in development and security workflows. Developers will use AI for code generation, testing, and remediation, supported by emerging "AI governance" within DevSecOps to ensure secure, compliant output. Security tools like SAST and DAST will leverage AI to suggest fixes, while attackers will use AI to automate zero-day discovery and adaptive malware—driving an arms race of defensive vs. offensive AI. Organizations that pair skilled professionals with AI-driven tools will be best positioned to manage growing threats. Fully autonomous security remains unlikely, but human-AI collaboration will dominate. Regulatory guidance on AI use and transparency will also shape security practices.

2 Further shift-left with "continuous everything"

As Agile, continuous deployment, and platform engineering accelerate development, AppSec must match pace. By 2026, high-performing teams will adopt continuous security—integrating checks at each commit, automated threat modeling, and real-time scanning into daily workflows. SAST feedback in IDEs, background DAST on each deployment, and security testing embedded in QA will blur the line between development and security. Emerging concepts like "continuous authorization" will let apps adapt defenses in real time, extending Zero Trust principles to the application layer.

3 Unified platforms and DevX focus

Tool consolidation will drive end-to-end AppSec platforms from major vendors and cloud providers, covering code, dependencies, infrastructure, and runtime with centralized analytics. These platforms will correlate issues across layers, leverage cloud-scale analytics, and prioritize developer experience by integrating into toolchains and providing actionable insights. Success will hinge on minimal friction for developers, with security "as code" and "as data" enabling policies, tests, and security telemetry to be defined, queried, and visualized seamlessly.

4 Rise of software liability and secure software as a market differentiator

With the EU Cyber Resilience Act as precedent, by 2026 software makers may face legal accountability for preventable flaws. Certification programs are likely, similar to US IoT labeling efforts. Buyers will demand proof of security—continuous vulnerability disclosure, SBOMs, and adherence to frameworks like NIST SSDF or ISO 27034. Vendors investing in security could gain trust and cyber insurance advantages, while others risk higher premiums, fines, or lost business. Gartner predicts that by 2026, 50% of C-level executives will have cybersecurity risk performance tied to their contracts, making "secure by design" an enforceable priority.[25]

25 TechRepublic, *Gartner reveals 8 cybersecurity predictions for the next 4 years,* June 22, 2022

5 New frontiers: Quantum prep, IoT/OT convergence, and beyond

While these are slightly further out, forward-looking AppSec teams are starting to consider them:

- **Post-quantum cryptography:** By 2026, the threat of quantum computers breaking current encryption might not be immediate, but proactive organizations (especially in government, finance) will begin transitioning to quantum-resistant algorithms for applications. This is a huge undertaking (as it involves updating cryptographic libraries, protocols, possibly hardware accelerators) and needs long lead times. So, 2026 might see the first wave of mainstream applications advertising "post-quantum" security features, especially as NIST has already standardized some post-quantum algorithms.
- **Secure Software Supply Chain ecosystems:** We may see **more industry-wide collaborations** to secure supply chains—for example, package repositories implementing mandatory 2FA for maintainers (some already have), or automated scans for malicious packages. By 2026, concepts like **dependency signing and verification** (e.g., using Sigstore and Sigstore's Cosign to sign container images and packages) could become standard. Ideally, an end-to-end verified chain: from a developer's commit (signed) to the build (reproducible builds and signed artifacts) to the deployment (infrastructure verifying those signatures). This could drastically reduce certain attack vectors, but requires broad adoption.
- **Application security for OT/embedded systems:** As operational technology and IT converge, the line between enterprise app security and product security blurs. AppSec teams might find themselves responsible for the security of software running in cars, factories, medical devices, etc., especially as those become connected. Techniques from traditional AppSec will be applied in these domains (e.g., threat modeling a power plant's monitoring app similarly to a banking app). The convergence also means more responsibility—an app vulnerability in an OT environment could lead to physical consequences. So by 2026, expect stronger regulation and guidelines specifically targeting software in critical infrastructure, and cross-pollination of AppSec best practices into those areas.

6 Human factor and skills

Despite automation, the need for skilled AppSec professionals will remain acute. If anything, by 2026 the skill gap might widen because securing complex AI-driven, cloud-native systems is even more challenging. Organizations will invest more in talent development. We might also see more outsourcing or "AppSec as a Service" models to cope with skill shortages, where specialized firms manage certain testing or monitoring tasks for companies that can't hire enough in-house expertise. However, long-term, a new generation of developers educated with security mindset from the start could begin to alleviate this. Ideally, by 2026 security starts to become a standard part of all software engineering curricula, which will gradually infuse industry with more security-aware professionals.

7 Threat landscape 2026+

We anticipate attackers will continue to do what works (social engineering, exploiting unpatched flaws) but will also pivot to new targets. APIs, as discussed, will remain a hot target; supply chain attacks are likely to increase before they decrease (because many companies are still catching up on defenses). Ransomware may evolve tactics if payments become harder (e.g., more data theft for extortion). The possibility of more nation-state cyber activity is high, potentially aiming at software critical to economies. One can imagine attackers trying to compromise widely used software at the source (similar to SolarWinds) for espionage or disruption purposes, which reinforces everything we said about supply chain security being vital. Attacks on CI/CD or software updates could become more frequent if they continue to yield success. The **geopolitical dimension of AppSec** will thus be more pronounced; companies might need to consider threats from advanced persistent threat (APT) groups and align their defenses accordingly, not just worry about cybercriminals.

In conclusion, we will likely see **application security become even more ingrained in the fabric of how we build software**—out of necessity. Organizations that embrace a forward-looking approach will be in the best position to protect their applications and users. The journey will require adaptability; AppSec leaders should foster a culture of continuous improvement and learning, as the threat and technology landscape can shift rapidly. Yet, the progress so far gives reason for optimism: **security is now a board-level issue, developers are more engaged in security, and powerful new tools are emerging to tip the balance in favor of the defender**. We expect application security to be not just a reactive cost center, but a **strategic enabler of trust** in the digital enterprise—a world where organizations can confidently innovate through software, knowing they have the resilience to withstand the cyberthreats of the future.

How OpenText addresses current and future application security challenges

OpenText Application Security is uniquely positioned to help enterprises meet these challenges head-on with a modern, integrated, and AI-augmented platform.

1 Tackling the AI explosion in development

Challenge: More than 33% of enterprises use GenAI in production, expanding the attack surface and introducing new classes of vulnerabilities.

OpenText response:

- **OpenText™ SAST Aviator™**, powered by LLMs, automates vulnerability triage and remediation guidance, significantly reducing false positives and boosting developer productivity.
- OpenText platform detects risks from insecure AI model integration (e.g., Python AutoGen, OpenAI libraries) via updated content from Software Security Research.
- Planned features include macro generation for authenticated DAST scans in GenAI-driven apps.

2 Securing the Software Supply Chain

Challenge: Supply chain attacks are increasingly sophisticated, targeting dependencies and pipelines.

OpenText response:

- OpenText embeds **Software Composition Analysis (SCA)** across the SDLC to identify and manage vulnerabilities in open-source components. **Open-Source Select** empowers developers to make better, compliant package choices at intake.
- Through our integration with **Debricked**, we enhance visibility into open-source health and ecosystem risk by leveraging community and contextual intelligence—helping teams evaluate popularity, maintenance, and security posture before adoption.
- Additional controls include integration with Sonatype Nexus Firewall and the Advanced Legal Pack for license governance and real-time vulnerability filtering. OpenText's approach aligns with NIST 800-53 SR controls and mandates a designated supply chain advocate for each product line—ensuring accountability and continuous risk oversight.

3 API security as a core priority

Challenge: API-driven architectures dominate, making APIs a top attack vector.

OpenText response:

- **SAST**, **DAST**, and **IAST** engines support comprehensive API scanning (REST, gRPC, GraphQL), aligned with the 2023 OWASP API Top 10.
- OpenText's **DAST** solution includes workflow-driven and macro-enabled scans to handle complex API scenarios.

4 DevSecOps at scale with developer-first focus

Challenge: Dev teams now influence tool selection and expect seamless integration into CI/CD.

OpenText response:

- OpenText supports deep integrations across IDEs, Git repositories, CI/CD platforms (e.g., Jenkins, GitHub Actions), and ticketing tools (e.g., Jira).
- Solutions are built for automation—ScanCentral for SAST/DAST enables parallelized scanning across cloud and on-premises environments.
- Secure Code Warrior integration delivers contextual training tied to real findings, reinforcing secure coding early in the lifecycle.

5 Managing tool sprawl and operational complexity

Challenge: Nearly half of enterprises are consolidating AppSec tools due to cost and management concerns.

OpenText response:

- Offers a unified **application security platform** with centralized policy, reporting, and compliance capabilities through Software Security Center and Application Security Insight.
- Modular architecture supports SAST, DAST, SCA, and ASPM under one umbrella, reducing overlap and streamlining operations.

6 Responding to regulatory and compliance pressure

Challenge: Regulations like the SEC's cybersecurity disclosure rules and SBOM mandates are raising the bar on software assurance.

OpenText response:

- The platform provides built-in support for SBOM generation, regulatory compliance mappings (e.g., GDPR, CCPA), and auditable workflows.
- **FedRAMP-authorized SaaS** and **Iron Bank-compliant on-premises** options support public sector and regulated industry needs.

7 Elevating detection accuracy and reducing false positives

Challenge: False positives reduce trust in security tools and slow developer adoption.

OpenText response:

- Proprietary **AI/ML auditing (e.g., SAST Aviator, Audit Assistant)** dramatically cuts review overhead, with human-grade classification accuracy.
- Additional capabilities like scan policy tuning, filtersets, and rule customization further reduce noise.

8 Preparing for the post-quantum era

Challenge: With NIST's post-quantum cryptography (PQC) standards emerging, organizations must assess and future-proof systems against quantum-enabled cryptographic breaks—without disrupting current operations.

OpenText response:

- Detection rules for weak or deprecated encryption algorithms and insecure key management practices are continually updated through Software Security Research (SSR) to track evolving PQC recommendations from NIST and ETSI.
- Integration with CI/CD pipelines automates cryptographic inventory—identifying libraries, APIs, and code paths requiring PQC-ready migration.
- Along with SBOM capabilities, security teams get a roadmap for phased crypto-agility, ensuring compliance readiness and reduced long-term exposure.

9 Defending against AI-powered and AI-targeted attacks

Challenge: AI is being weaponized both as an attack tool and an attack target, enabling automated vulnerability discovery, adaptive phishing, and malicious model manipulation.

OpenText response:

- AI/ML detection capabilities identify insecure AI model integrations, prompt injection vulnerabilities, and unsafe handling of LLM output—covering frameworks such as OpenAI, AutoGen, and LangChain.
- DAST solutions can simulate adversarial API interactions to test resilience against AI-driven fuzzing and model exploitation attempts.
- Application Security Aviator, powered by LLMs accelerates triage and flags code patterns susceptible to AI-specific exploits.
- Continuous SSR content updates ensure that AI-related vulnerability categories are tracked and expanded as new research emerges.

Ready to go deeper? Download the extended companion guide, *How OpenText addresses current and future application security challenges*, for implementation detail you can put to work.

10 Securing smart contracts and blockchain applications

Challenge: As blockchain adoption expands, smart contracts have become high-value targets due to their immutability and direct control over assets. Vulnerabilities can lead to irreversible financial and reputational loss.

OpenText response:

- Supports static and dynamic analysis for Solidity and other smart contract languages, detecting common vulnerabilities such as reentrancy, integer overflows/underflows, unchecked calls, and access control flaws.
- Rulesets draw from both OWASP and blockchain-specific security research, providing coverage for decentralized finance (DeFi) risks and NFT marketplace code.
- Integration with OpenText SSC enables centralized policy enforcement for blockchain projects alongside.

Summary

OpenText Application Security offers a future-ready, deeply integrated, and developer-centric platform designed to meet the evolving challenges of modern software development. Whether securing GenAI applications, defending APIs, enabling DevSecOps at scale, or ensuring regulatory compliance, OpenText empowers security and engineering teams to build resilient, secure-by-design software—without compromising velocity or innovation.

opentext™