Hype Cycle for Application Security, 2025

22 July 2025 - ID G00827963 - 101 min read

By Dionisio Zumerle

Application security innovations continue to emerge in response to new AI challenges, the evolution of DevSecOps and the need for convergence of application security tools. Cybersecurity leaders must adopt innovations to further mature and advance their application security programs.

Strategic Planning Assumptions

By 2027, at least 30% of application security exposures will result from usage of vibe coding practices.

By 2026, at least 40% of organizations will default to their application security testing vendors for AI-based autoremediation of vulnerable code.

Through 2029, over 50% of successful cybersecurity attacks against AI agents will exploit access control issues, using direct or indirect prompt injection as an attack vector.

Analysis

What You Need to Know

Cybersecurity leaders continue to advance their application security programs, but major shifts in application architectures, development styles and attacks are occurring. These changes are disruptive, requiring leaders to reconsider existing best practices and technologies for application security, adding new ones where needed.

The rapid adoption of generative AI (GenAI) induces risks, with AI-enabled development and applications expanding the attack surface. GenAI, however, can also help remediate vulnerabilities: Organizations that struggle with streamlining DevSecOps are turning to AI code security assistants (ACSAs), in addition to application security posture management (ASPM) and techniques such as reachability analysis, to improve the developer experience.

Organizations often have multiple application security tools to secure and protect applications, third-party code, APIs, AI components, cloud infrastructure and/or the software supply chain. However, the overlap among these tools requires an effort to rationalize application security products into broader platforms.

Cybersecurity leaders must take advantage of the underlying trends in the application security space to onboard the most appropriate innovations at the optimal time for their organizational maturity.

The Hype Cycle

This Hype Cycle tracks the maturity and adoption of processes and technologies that can help organizations advance their application security programs. Four main trends are

reshaping the current state of application security through disruptive innovations:

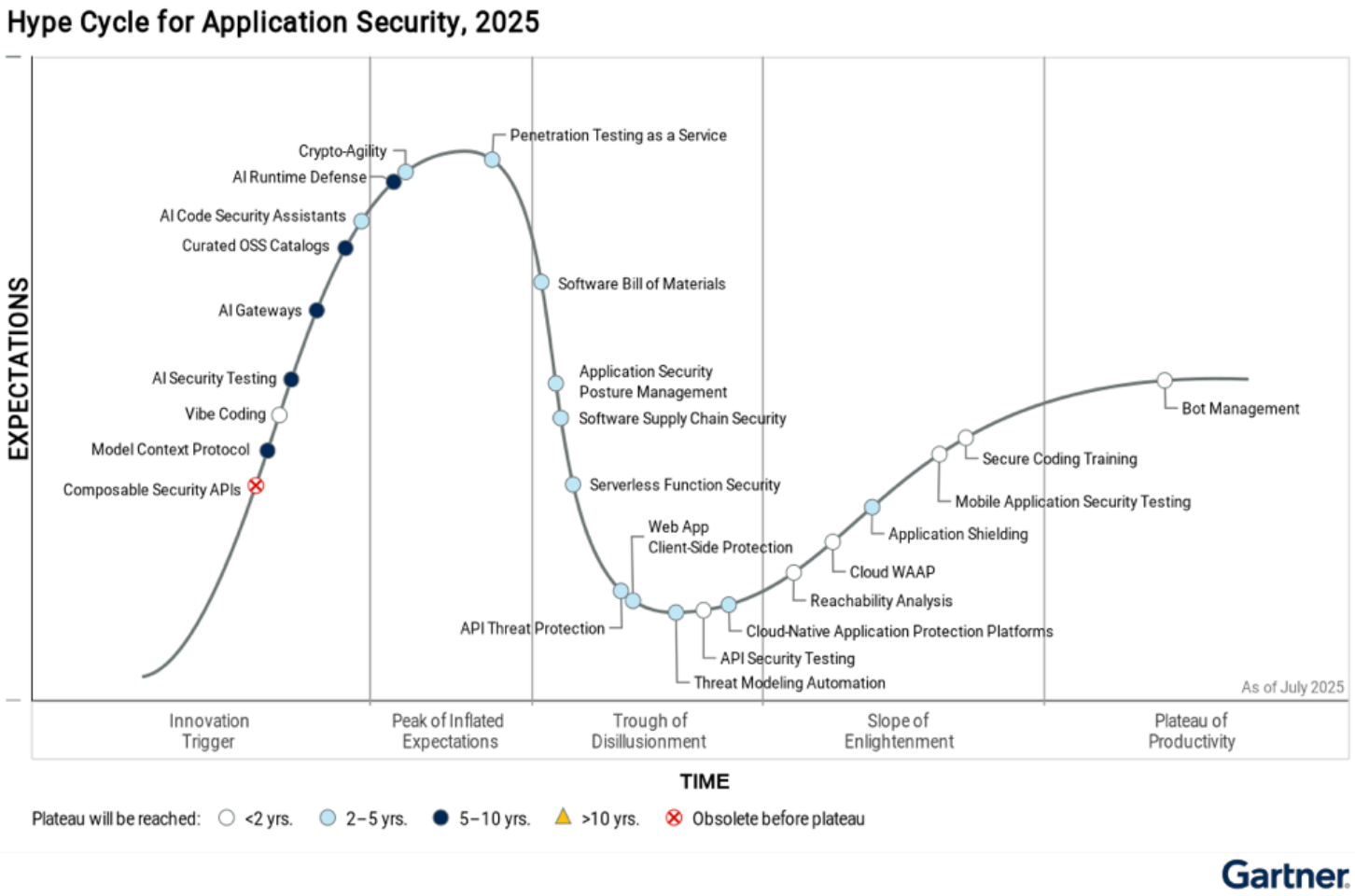
1. **Securing AI-augmented code development:** Application developers are starting to use AI to develop more code at a faster pace. AI coding assistants and vibe coding, while increasing productivity, introduce new security concerns and vulnerabilities that need to be managed. [1] At the same time, however, Gartner is observing adoption of ACSAs. These provide virtual security-champion-like capabilities to help software engineers automate tasks, such as application vulnerability identification and remediation, and to deliver more secure code.

2. **Streamlining DevSecOps:** Organizations are taking a "shift-left" approach to their application security testing. They are entrusting their software engineers to not only conduct the testing, but also remediate any vulnerabilities found. [2] Application security leaders, instead, now define application security policies and monitor their correct enforcement. Innovations such as reachability analysis and ASPM are maturing to serve security leaders and improve developer experience and enablement.

3. **Bridging code and workload security:** Cloud-native and other modern applications merge application code with cloud infrastructure and elevate the speed of application deployment. This trend requires application security that is workload-aware and that interconnects development with runtime. To address the evolving architectures and threats, innovations that follow this pattern, such as API threat protection, are further advancing.

4. **Securing GenAI-enabled applications:** AI, especially generative AI, expands the attack surface with AI agents and GenAI-enabled applications. AI security testing is starting to help cybersecurity leaders identify exposures in these AI-enabled applications. In addition, AI runtime defense is providing visibility into activity to identify and prevent attacks, such as prompt injection.

Figure 1: Hype Cycle for Application Security, 2025



The Priority Matrix

The Priority Matrix maps the benefit rating for each technology or innovation against the amount of time the technology or innovation is likely to take to achieve mainstream adoption. This perspective can help cybersecurity leaders decide how to prioritize their application security investments.

AI-related innovations are maturing faster than others in this Hype Cycle. Innovations such as Model Context Protocol (MCP), used to connect AI agents with tools, and vibe coding introduce new risks. Others, such as AI security testing, AI runtime defense and ACSAs,

enable cybersecurity leaders to mitigate those risks.

Crypto-agility requires cybersecurity leaders to assess their application inventory and modernize encryption to withstand upcoming quantum-enabled cryptanalysis attacks.

ASPM, secure coding training and reachability analysis continue to evolve and mature to help organizations streamline DevSecOps. These innovations help organizations continuously prioritize vulnerabilities based on risk and automate their remediation workflow.

Cloud-native application protection platforms (CNAPPs), cloud web application and API protection (WAAP), API security testing, and API threat protection are all innovations that address the security needs of contemporary, cloud-native and decentralized applications. They overlap with each other, and the broader space is starting to be impacted by the tech product optimization trend, which sees organizations trying to balance stand-alone products and consolidated platforms.

However, modern applications also heavily reuse third-party software. [3] Organizations continue to adopt software supply chain security (SSCS), along with software bills of materials (SBOMs), to better secure development environments and pipelines against growing exposures and attacks. [4]

Priority Matrix for Application Security, 2025

Benefit	*Years to Mainstream Adoption*			
	Less Than 2 Years	*2 to 5 Years*	*5 to 10 Years*	*More Than 10 Years*
Transformational	Vibe Coding	AI Code Security Assistants Application Security Posture Management Software Supply Chain Security		
High	API Security Testing Bot Management Cloud WAAP Reachability	**API Threat Protection Cloud-Native Application Protection Platforms**	**AI Runtime Defense Curated OSS Catalogs**	

Source: Gartner (July 2025)

Off the Hype Cycle

The following innovations have been removed from the Hype Cycle:

- **DevSecOps** is a widely adopted discipline that, despite its streamlining challenges, has reached full maturity. Other innovations in the Hype Cycle, such as ASPM, try to tackle

such challenges.

- **Application monitoring and protection** has materialized through application observability products, but we do not see security programs relying on its functionality to identify vulnerabilities.
- **Policy as code**, **Kubernetes security**, **full life cycle API management** and **DevOps platforms** keep maturing and continue to grow in adoption, but are less central to the focus for cybersecurity security leaders. Thus, we have removed them from this Hype Cycle.

In addition, **AI security and anomaly detection** was renamed to **AI runtime defense** this year.

On the Rise

Composable Security APIs

Analysis By: Mark O'Neill, Manjunath Bhat

Benefit Rating: Moderate

Market Penetration: 1% to 5% of target audience

Maturity: Emerging

Definition:

Composable security APIs are security capabilities such as privacy vaults, authentication services, encryption services and digital signing services that are consumable as APIs. Developers can use composable security APIs to embed security capabilities in their applications.

Why This Is Important

Composable security APIs enable developers to embed security capabilities into applications. Rather than a stand-alone category, these APIs are becoming a capability of larger security platforms.

Business Impact

- Composable security APIs empower developers to select and implement security capabilities within the applications they build. This further drives developers as key technology selectors.
- The use of composable security APIs creates dependencies that bring a business impact if the service becomes unavailable or if the provider suffers a security breach.
- Composable security APIs enable a better relationship between development and security teams, with security teams evaluating and approving services. Platform engineering teams may support these security services as well.
- Misconfigurations in security mechanisms are a common cause of exposure for organizations. The use of composable security APIs, provided by a specialist vendor, may reduce that risk.

Drivers

- APIs are now a core part of software development. Developers expect to have reusable services provided by APIs as part of an “API-first” approach.
- Standardizing software components is a recommended step when maturing an organizational application security program. This drives a “secure by default” approach.
- Developers are typically not security experts. The use of composable security APIs allows them to include security capabilities in applications without the requirement to build the

functionality themselves. Therefore, the resulting application is built faster and more securely than if it was wholly custom-built.

- Developers are becoming more than just influencers in technology or architecture choice and are now often driving decision making and vendor selection (for more, see **Engage the Developer Persona to Fuel Your GTM Strategy**). In turn, vendors increasingly target developers as part of a product-led growth sales motion. This drives the growth of developer-focused products.

Obstacles

- Security APIs are now typically provided by larger security platforms, reducing the market for dedicated vendors in this category.
- APIs may be difficult to swap or replace, which creates difficulty if composable security APIs must be changed for another. This potential lock-in presents a major obstacle for customers, but is advantageous for vendors.
- When security is already built into a platform, or can be built using coding assistants, then the use of separate security services may not be needed. This is a major factor in this category becoming “obsolete before plateau.”

User Recommendations

- Evaluate the use of composable security APIs for use cases including vaults, encryption, file scanning and authentication.
- Build a contingency plan if the API becomes unavailable or the provider suffers a breach.
- Use API-management solutions to monitor the uptime and usage of the APIs your organization depends upon, including composable security APIs. An outage of an authentication service, for example, means that customers or employees cannot sign into

applications.

- Prioritize security APIs provided by larger security platforms.

Sample Vendors

BoxyHQ; HashiCorp; Okta (Auth0); Skyflow

Gartner Recommended Reading

How to Protect Your Cloud-Native Applications in Production

Model Context Protocol

Analysis By: Andrew Humphreys, David Pidsley

Benefit Rating: High

Market Penetration: Less than 1% of target audience

Maturity: Emerging

Definition:

Model context protocol (MCP) is an emerging standard to enable two-way communication between AI models and other applications and data sources. It provides a standardized way for applications to share contextual information with large language models (LLMs) and expose tools and capabilities to AI systems.

Why This Is Important

MCP was published by Anthropic in November 2024 and has been adopted by many vendors, including Amazon Web Services (AWS) and Microsoft. It supports LLM-based AI

application development by simplifying connecting to sources of information and functionality. This has been challenging, as it required writing custom code or using APIs, which have not been designed for AI usage. MCP defines a standard way to access external data and tools in AI workflows, increasing consistency and interoperability.

Business Impact

MCP can lead to more context-aware AI systems and can reduce integration effort by standardizing how to give AI models access to contextually relevant and up-to-date multimodal data and multistructured analytics beyond their initial training data. However, MCP brings additional security concerns, as well as strategic questions about its usage. It also may be eclipsed by future standards that have greater openness and security.

Drivers

- MCP extends the reach and simplifies implementing and maintaining LLM-based AI applications and agents. They can accomplish this because they can more easily be given access to new data sources or tools without retraining the model or significantly altering the core system, increasing the flexibility and agility of the AI solutions.
- Productivity and effectiveness for AI agent developers of LLM-based AI applications and agents is increased because connecting to data sources and tools is simplified. Developers can use MCP to integrate with multiple, diverse systems without needing to use system-specific approaches and tools, reducing the cost and time to integrate.
- AI service providers and multiple application, analytics, decision intelligence and middleware vendors are starting to offer prebuilt MCP-defined integration interfaces in their applications and MCP support in tools. This further helps grow the MCP community adoption and ecosystem and simplifies adoption. This can lead to increased value and innovation as teams can reuse predefined MCP interfaces for new use cases without needing to build new MCP interfaces or custom integration points.

Obstacles

- MCP adds new security risks, particularly related to authorization, to data and tools and raises concerns about data privacy and security. Ensuring that classified information is protected and that access is appropriately controlled is crucial.
- Many organizations already register tools and resources as part of their AI platform architectures. AI development tools such as AIXplain, Boomi AI Studio, Crew.AI, Langchain and SnapLogic already provide mechanisms to register resources to be accessible by LLM-based AI applications and agents.
- Implementing MCP servers can be complex, especially for legacy systems or diverse technology stacks and may lead to suboptimal architectures and governance. Integrating MCP into existing infrastructure may require significant effort and expertise.
- MCP is relatively new and is evolving rapidly, with limited backward compatibility. Early adopters may face challenges with keeping implementations up to date with the latest version of the standard.

User Recommendations

- Investigate MCP clients to understand the benefits and risks, and compare these with the capabilities offered by your AI development platforms.
- Prototype MCP servers for your internal systems and data sources to facilitate streamlined access for AI applications developers.
- Avoid the highest risks associated with the current MCP standard by prohibiting remote MCP server implementations that are externally accessible. Any use of external MCP servers should be strictly controlled within a test environment to assess MCP benefits, rather than deploying them in production.
- Budget to address technical debt, ensuring your teams have the resources and time to

monitor the standard's evolution and adapt their implementations accordingly, with the flexibility to abandon the standard if MCP loses its initial advantage.

- Strengthen security measures by requiring MCP servers to use HTTPS endpoints and implementing OAuth, while fortifying configurations to protect MCP client and server description files.

Sample Vendors

Anthropic; Axiom; Cloudflare; Composio; Google; Microsoft; OpenTools; Stripe; Zapier

Gartner Recommended Reading

Emerging Patterns for Building LLM-Based AI Agents

Market Guide for AI Application Development Platforms

Innovation Insight for the AI Agent Platform Landscape

Innovation Insight: Model Context Protocol

Vibe Coding

Analysis By: Bill Blosen, Peter Hyde

Benefit Rating: Transformational

Market Penetration: 1% to 5% of target audience

Maturity: Emerging

Definition:

"Vibe coding" is a term coined in February 2025 by computer scientist Andrej Karpathy. It transcends AI-augmented development tools to envision a new state of human-computer interaction. Developers become composers, using voice recognition or light keyboarding to rapidly prototype complex yet throwaway, not-for-production software. Vibe coding ignores the generated code, focuses on results and has AI solve all bugs. Thus, the developer stays in a high productivity state known as "flow."

Why This Is Important

Generative AI (GenAI)-augmented coding is a paradigm shift in how software is created and managed. Vibe coding builds on that and is a harbinger of the future of software programming. Much hype is being generated, and it is important to understand it clearly, as there are profound implications; the advancement of low-code, AI coding assistants and vibe coding may alter the need for traditional professional coding. In addition, the risks of implementing this improperly are very high.

Business Impact

Vibe coding must be considered by enterprises that prioritize innovation and rapid learning cycles with focus group customers. As tools develop and are trained on reliable and secure codebases, vibe coding will become viable and is poised to give new and experienced engineers a massive gain in innovation, creativity and productivity.

Drivers

- **GenAI technology advances:** State-of-the-art, frontier GenAI models from Anthropic and OpenAI have enabled developers to implement solutions in a fraction of the time they previously spent looking things up on Stack Overflow or Google. Many AI coding assistants now support vibe coding through implementing new agentic capabilities that allow the developer to easily compose complex software through a flow-based conversation.

- **Desire for rapid customer feedback:** Enterprises are increasingly focused on accelerating the cycle of delivering software to generate learning from customers. Vibe coding and fast cycles like lean startup are a natural fit for early-stage venture-capital-funded startups.
- **Repository indexing:** As AI tools become more refined, the quality of source code used to prompt or tune them will improve. Fine-tuning with your own codebases allows the tools to integrate your unique architecture, security and reliability patterns into the software produced by vibe coding.
- **The search for flow:** Software developers value having time to focus on their core work, but often do not have enough of it. Flow state is a high-productivity mental state in which a person is completely focused on their work. Vibe coding keeps a developer in this state by continually focusing on creating functionality without any need to fix bugs, create tests or review the code.
- **Voice technology advances:** Software coding involves significant amounts of typing and mousing to write and manage code. Voice recognition and text-to-speech technologies have now advanced to provide an even further detachment from coding distractions and rote tasks, thus allowing easier access to flow states.

Obstacles

- **Code quality:** The code produced by vibe coding is in no way intended for production use today. Tools supporting vibe coding use GenAI models trained on internet code and focus only on creating functionality without attention to quality principles, like testing and code review. Over time, this will fade.
- **Risks:** Unlike production code, little is known about the contents of the vibe code. Until a track record is established, Gartner recommends using this code only for prototyping and creative innovation.

- **Resistance:** With the rapid pace of change, many workers today are experiencing change fatigue. Although developers enjoy flow state and nice tools, their resistance to more change may overpower it. Developers also take pride in their work and may resist the risks of vibed code.
- **Limited gain:** The resulting code is useful only for innovation experiments. Also as coding is a small portion of the full software development life cycle, the risk may outweigh the benefits.

User Recommendations

- Do not use vibe-coded software in production. Limit it to a controlled, safe sandbox for execution. Use platform engineering principles to provide a safe paved road for engineers to use vibe coding tools and techniques.
- Prioritize AI code assistant tools that incorporate agentic flows with excellent context awareness through RAG or fine-tuning capabilities.
- Enable pilot teams to explore by supporting an innovation culture. Find engineers searching for flow. Lean startup and dedicated innovation teams will easily adopt these techniques. Consider product areas that have rapid feedback opportunities, like customer-facing interactions where you can find willing beta audiences.
- Build a strong integration strategy, libraries of modular components and an API catalog to ease composing programs with vibe coding.

Sample Vendors

Anthropic; Anysphere; Lovable; Microsoft; OpenAI; Replit; StackBlitz; Windsurf

Gartner Recommended Reading

Magic Quadrant for AI Code Assistants

Innovation Insight: Open Generative AI Models for Coding

How to Improve Developer Effectiveness Using AI Coding Tools

AI Security Testing

Analysis By: Jeremy D'Hoinne, Dionisio Zumerle, Dennis Xu

Benefit Rating: Moderate

Market Penetration: 1% to 5% of target audience

Maturity: Emerging

Definition:

AI security testing (AI-ST) uncovers vulnerabilities and exposures in AI-enabled systems and applications.It offers specialized assessments for AI applications, including offensive tactics like automated generation and execution of adversarial prompts and AI component scanning — model repositories, libraries, frameworks and notebooks. AI-ST can function independently, or as part of an AI security platform also including runtime controls.

Why This Is Important

Many organizations have built custom AI applications, including chatbots, and are now considering AI agents. These AI applications include models, which extend the attack surfaces, especially probabilistic models like large language models that carry new risks, such as prompt injection. Further, these applications are developed at a fast pace leveraging emerging and low maturity tools and environments, increasing the risks of vulnerabilities. AI-ST enables the identification of these vulnerabilities early in the development cycle.

Business Impact

Business, IT and cybersecurity stakeholders require a way to estimate the resilience of AI systems against attacks, preferably before pushing them to production. This is especially true for public-facing AI applications, including agents. Existing application security tools have not yet fully evolved to include AI-specific tests, including adversarial prompts or model inversion techniques.

Drivers

- As organizations transition AI application pilots to production, they face a distinct set of risks due to the lack of a novel approach to testing adapted to natural language inputs and outputs. This highlights the need for comprehensive AI risk assessment and enhanced testing procedures.
- GenAI applications are driving a lot of innovation, but carry specific risks due to the probabilistic processing of inputs and generation of outputs. Risks of data leakage, prompt injection and compromised outputs are higher for these applications.
- Organization stakeholders require an assessment of the resistance to attacks and misuse of their chatbots, and of goal-driven AI agents before authorizing external-facing deployments.
- Security teams look for ways to automate the testing part because the fast-paced development of AI applications require frequent retests.
- Many AI applications leverage commercial or open-weight AI models that are stored in specialized repositories, with various file formats and metadata. Configuration mistakes, model and metadata tampering can significantly alter application behavior.
- Global regulations and popular frameworks such as NIST AI RMF are driving the need for specific AI testing practices.

Obstacles

- Organizations might not have full visibility and management of their AI models, especially locally downloaded models, limiting the ability to apply AI-ST at scale.
- AI-ST responsibilities are often unclear due to the involvement of multiple teams, including some outside of the traditional application development cohort. This can lead to AI applications being created, and reaching preproduction without a proper approach to application security testing.
- The nondeterministic nature of both AI applications and attacks against them complicate the evaluation and benchmark of AI-ST products. Consequently, security teams in charge of AI application security face challenges in defining what constitutes a successful test result.
- Most AI-ST are startups without the visibility and proven track record to inspire confidence about the quality and comprehensiveness of the provided tests.
- Not all providers offer a comprehensive set of AI-ST features. Many providers are limited to offensive security testing.
- Testing is not a one-time activity. AI-ST technologies rarely offer mature retesting or result analysis guidance.

User Recommendations

- Implement AI-ST as part of a structured AI trust, risk and security management (AI TRiSM) program.
- Ensure that deployment options (on-premises vs. privately hosted, SaaS) comply with your requirements.
- Review your inventory process for all AI assets, including local and remote models, and

resource-constrained environments (e.g., edge AI), as the AI-ST program depends on this visibility.

- Evaluate offensive security testing for AI cybersecurity risks (e.g., prompt injection, data leakage) and also consider their AI risks (e.g., harmful, toxic). Prioritize tools supporting continuous testing and flexible retesting.
- Assess your needs for offensive security multimodal chatbots that may accept visual or audio input in addition to text, or coverage for other types of AI models.
- Evaluate scanning capabilities for scope (model, metadata, notebooks), supported model repositories and file types.

Sample Vendors

Aim Security; CalypsoAI; HiddenLayer; KELA; Mend.io; Mindgard; Pillar Security; Prompt Security; Protect AI; TrojAI

Gartner Recommended Reading

Use an AI Security Platform to Launch Your AI Security Strategy

AI Gateways

Analysis By: Andrew Humphreys, Mark O'Neill

Benefit Rating: Moderate

Market Penetration: 1% to 5% of target audience

Maturity: Embryonic

Definition:

AI gateways, also known as LLM routers or multi-LLM gateways, manage and secure connections to AI providers. Some AI gateways are based on established API gateway products, since REST APIs are the primary mechanism for connecting to AI providers, while others are purpose-built. AI gateways can be used to apply security, such as protection of API keys issued by AI providers, multi-LLM routing, cost visibility, and data privacy scanning to an organization's AI usage.

Why This Is Important

AI gateways manage interactions between applications and AI models by providing security, governance, observability and cost management. This reduces the risk of unexpected costs incurred from AI providers, and prevents private data in API traffic from being compromised or misused. An AI gateway may also enable an organization to manage usage of multiple AI models, from different providers, rather than focusing the organization on one provider.

Business Impact

As the volume and scale of generative AI and other types of AI projects have increased, organizations require greater control over their use of AI providers. AI gateways provide runtime traffic management between the organization and their AI providers. The gateways can help implement and manage prompt-based policy controls, track AI service use and costs, route across multiple LLMs, and manage access to AI subscriptions, including protecting API keys issued by AI providers.

Drivers

- **Control access costs:** Pricing models for AI services tend to be token-based, which presents a risk for businesses as the costs associated with use of these services can accrue rapidly, and requires special metering ability beyond an API gateway's typical

transaction-based metering. Organizations are looking to control access costs, and AI gateways can help by caching responses to limit duplicate calls and tracking and controlling access to the services.

- **Enable visibility of use of AI services:** AI gateways provide a way to get greater visibility into the use of AI APIs across the organization by leveraging API management observability and analytics features for API usage.
- **Optimize access to AI engines:** AI gateways can be configured to provide a single API in front of multiple different AI providers, such as multiple LLMs. This means that developers can access multiple AI services using the same API.
- **Enforce security on AI interactions:** AI trust, risk and security management (AI TRiSM) ensures AI model governance, trustworthiness, fairness, reliability, robustness, efficacy and data protection. AI gateways address aspects of AI TRiSM, including model governance and reliability. In particular, protection of API keys issued by AI providers is vital. If an attacker gets access to an organization's API keys for an AI provider, they may access private data and run up large usage bills. AI gateways can be used to protect API keys issued by AI providers.

Obstacles

- AI gateways are new and largely unproven and in the rapidly evolving AI environment new requirements emerge frequently; for example, to be able to manage Model Context Protocol endpoints. Conducting a thorough evaluation before committing to a vendor is a requirement.
- Latency is a concern for any intermediary, and especially in AI where end users may notice any delay in response time. Unlike traditional API latency, the important metrics here are time-to-first token and token per seconds (or token throughput). An AI gateway may help users optimize these two metrics.

- A single point of failure is also a concern if an AI gateway is deployed without redundancy in place.
- Caching of AI API responses is ineffective and complex, since different AI models may use different prompts. Determining whether natural language requests are requesting the same information is difficult, particularly when dealing with a large context window.

User Recommendations

- Consider using an AI gateway if you are looking to implement multiple LLM use cases or running AI applications at scale, as they provide centralized control, security, and cost management that is focused on AI-specific requirements that are not supported by standard API gateways.
- AI gateways are a relatively new category, and the features that are supported differ between vendor offerings. Conduct a thorough evaluation to evaluate the specific features and capabilities of the AI gateway you're considering to ensure that it meets your requirements.
- For simpler scenarios, such as rate-limiting of AI APIs, evaluate your existing API gateway's capabilities to act as an AI gateway, to see if they already or can be upgraded, to support your requirements.
- Ask your AI providers whether they can provide similar functionality, which would remove the requirement for a separate AI gateway.

Sample Vendors

Aguru; Cloudflare; lunar.dev; Martian; Portkey; Radiant

Gartner Recommended Reading

AI Pushes API Management Providers to Reimagine Their Products

Market Guide for API Gateways

How to Calculate Business Value and Cost for Generative AI Use Cases

10 Best Practices for Optimizing Generative AI Costs

Innovation Insight: AI Gateways

Curated OSS Catalogs

Analysis By: Manjunath Bhat

Benefit Rating: High

Market Penetration: 1% to 5% of target audience

Maturity: Emerging

Definition:

Curated open-source software (OSS) catalogs are trusted repositories of vetted OSS dependencies that enable policy-based curation. The repositories can either be curated externally by a provider or internally by the organization based on security, compliance and operational policy checks. These checks can include open vulnerabilities, license compliance and project health. Their goal is to prevent introducing unmaintained, vulnerable, noncompliant and malicious code into the organization.

Why This Is Important

Organizations have significantly increased their use of OSS. Yet, risks to the software supply

chain from malicious and vulnerable OSS dependencies continue unabated. Providing developers with a curated catalog of trusted open-source packages improves governance without impeding developer experience. It ensures centralized visibility, streamlines patching and enables versioning.

Business Impact

Given the ubiquity of open-source usage in software development, curated catalogs can help every software engineering organization reduce exposure to security risks. By centralizing the responsibility for security scanning, dependency management, and compliance monitoring, curated catalogs allow organizations to benefit from open-source innovation while mitigating associated risks. This makes it easier to ensure regulatory compliance, particularly when creating software bills of materials (SBOMs) or monitoring Common Vulnerabilities and Exposures (CVE) reports.

Drivers

- The onslaught of software supply chain attacks on upstream OSS is making security and risk teams concerned about unfettered use of OSS within the organization. These concerns include security risks such as typosquatting, backdoors, dependency confusion and zero-day vulnerabilities, as well as operational risks from using out-of-date and unmaintained packages, which are easy targets for software supply chain attacks.
- Government regulations are mandating the use of SBOMs, necessitating greater visibility and auditability of OSS being used internally within organizations. For organizations using open-source components in their technology stacks, these curated catalogs offer a path toward standardized, secure, compliant and maintainable software ecosystems.
- Platform engineering teams provide curated development experiences via secure “paved roads” for a broad range of software development life cycle (SDLC) use cases. Therefore, providing a curated OSS catalog aligns well with platform engineering goals and helps

meet the need for both developer autonomy and governance.

- A few vendors now provide trusted OSS packages "as a service," with the assurance that the curated catalog contains prevetted OSS packages that are safer to use. This delivery model helps drive adoption by making it easier to create a catalog. The vendors assume responsibility for finding and fixing vulnerabilities, as well as for blocking packages that pose software supply chain security risks.

Obstacles

- Curated OSS catalogs deliver maximum benefit when they are part of internal developer platforms consumed by multiple product teams. Therefore, organizations that lack a platform engineering approach may find it difficult to scale the use of curated catalogs.
- Developers may perceive a curated catalog as a threat to their freedom and autonomy, especially if it is positioned as an "enforcer" of OSS governance rather than an "enabler."
- The lack of integration with continuous integration/continuous delivery (CI/CD) pipelines can make it difficult to systematically block packages from deploying to production using automated verification. This shortcoming dilutes the purpose of the catalog and undermines its utility.
- Organizations that operate in silos of application security, software engineering and infrastructure and operations will find it difficult to create catalogs that provide centralized visibility into OSS usage across all teams.

User Recommendations

- Establish and maintain a "trusted repository" of vetted and approved dependencies aligned with your governance policy, including supporting processes (evaluating requests for exceptions, updates, maintenance, etc.).

- Prevent vulnerable, unapproved, malicious and noncompliant packages from entering the SDLC by continuously scanning and assessing packages in the curated catalog for security, compliance and operational risks. This effectively blocks access to unapproved third-party components with minimal developer friction and requires developers to seek exemptions for using unsanctioned OSS.
- Treat the curated OSS catalog as a dynamic repository rather than a static list. Establish trust in the OSS based on continuous policy checks, rather than a single point-in-time scan.
- Ensure that the curated catalog integrates with developer workflows and existing CI/CD pipelines to automate policy enforcement checks.

Sample Vendors

ActiveState; Chainguard; Docker; JFrog; Lineaje; Red Hat; Seal Security; Sonar; Sonatype

Gartner Recommended Reading

Mitigate Enterprise Software Supply Chain Security Risks

3 Steps for Assessing an Open-Source Software Project

How Software Engineering Leaders Can Mitigate Software Supply Chain Security Risks

Innovation Insight for SBOMs

Application Security Guide for Software Engineering Leaders

AI Code Security Assistants

Analysis By: Mark Horvath

Benefit Rating: Transformational

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

Artificial intelligence (AI) code security assistants (ACSAs) are technologies that help developers identify and remediate security vulnerabilities in code. To do so, they offer autoremediation suggestions, and direct code assistance or chatbots, using forms of AI, such as generative AI (GenAI). ACSAs are primarily delivered as features of application security testing (AST) products and use cloud services to analyze code and make recommendations.

Why This Is Important

Developers are not well-trained to identify and resolve security issues in their code. Secure code training for developers is recommended for application security programs, and AI code assistants can be a multiplier for learning. When properly trained, the large language models (LLMs) offered by AST vendors deliver more secure code. Developers write better secure code when they get effective answers in a security context, LLMs and ACSAs offer more-scalable, security-centric code advice.

Business Impact

- ACSAs go beyond more-general AI coding assistants by offering just-in-time (JIT) solutions to the specific security problems found by AST tools.
- Unlike most tools, ACSAs may be in a unique position to optimize several variables (e.g., security and code quality) at the same time.

- ACSAs are next-generation tools for closing one of the remaining gaps — JIT training and security remediation — in shifting security left in the modern, secure software development life cycle (SDLC).

Drivers

- AI assistants, such as GitHub Copilot or Google Vertex, are becoming increasingly popular with developers. AI assistants are delivering useful results in multiple areas of their code including secure coding and in the infrastructure and operations of the production environment.
- Shift-left initiatives — moving work from the runtime environment (the right) to the developer (left) — are often slow to be adopted or fail completely. This is because the channels needed to supply developers with solid security guidance are limited and fragile. Initiatives often rely on experts to guide the organization. ACSAs offer an opportunity to add virtual experts at many layers of the organization in a way that is easily consumed by developers.
- AST vendors, which have traditionally supplied tools for AppSec, have offered security code assistance for more than a decade, and have access to the core security coding data their tools generate. Using this data as a training corpus, and combining them with foundational models into ACSAs, was a logical next step.
- Developers believe that ACSAs will provide reliable security advice. The 2025 Gartner Software Engineering Survey (n = 300) found that a surprising 69% of developers expressed high or medium confidence in the tools.

Obstacles

- Most organizations worry about exfiltrating private data or intellectual property (IP) through AI assistants. There are ways to protect against this, but trust is low.

- GenAI occasionally hallucinates, identifying problems that are not real or are wrong, or giving explanations that sound plausible, but are insecure or unwise to use. This can be hard to detect, especially for junior developers.
- Code ownership issues are a concern, especially if an ACSA has contributed a significant amount of code. Given that the assistants are generally using code they were trained on, rather than creating original code, accidentally acquiring another organization's IP is a concern.
- ACSAs are an example of optimizers, which have failed in the past by overcompensating for security while ignoring issues such as performance, reliability and code quality.

User Recommendations

- Continue to use traditional AST tools — static AST (SAST) or SCA — to measure security and code quality issues when adding an ACSA. Although it's common to see an immediate increase in security metrics (e.g., code quality), watch the other indicators of code quality.
- Protect your privacy and IP by avoiding vendors that do not handle privacy and confidentiality issues with your code in a manner consistent with your organization's policies and regulatory requirements.
- Retain and adjust existing methods of improving code security, as ACSAs become a bigger part of the developer experience.
- Continue programs such as security coaches and code reviews so they act as a balance to ensure that the advice developers receive is correct and helpful.

Sample Vendors

Black Duck; Checkmarx; GitHub; GitLab; HCLTech; Qwiet AI; Snyk; Symbiotic; Veracode

Gartner Recommended Reading

Innovation Insight for Generative AI

Magic Quadrant for Application Security Testing

Conversational AI Agents and Assistant Applications Need Guardrails

Magic Quadrant for AI Code Assistants

At the Peak

AI Runtime Defense

Analysis By: Jeremy D'Hoinne, Avivah Litan, Dionisio Zumerle

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Emerging

Definition:

AI runtime defense implements intent-based policy enforcement and anomaly detection for AI applications and models. It offers content inspection for application abuse and attacks (e.g., prompt injection) and aims to detect intent or content anomalies (toxicity, hallucination). Some AI runtime defense tools are application-specific (e.g., chatbots). AI runtime defense is a common feature of AI governance platforms, but can also be a stand-alone product when focused on cybersecurity.

Why This Is Important

Generative AI (GenAI) applications require specialized controls to mitigate cybersecurity risks. AI runtime defense often specializes in application security, typically on prompt abuses in the case of large language model (LLM) chatbots, but extends to other types of attacks, such as model query attacks. It can also help mitigate content anomalies by monitoring the model for jailbreaks, data leakage, performance drift, and biased or toxic model input or output.

Business Impact

AI runtime defense offers real-time visibility of potential AI anomalies or attacks. In organizations with a large number of AI applications, centralized visibility and alert management help identify issues shared by multiple applications using the same AI model. LLM-based applications represent a growing share of new applications that require controls on inputs to avoid application abuses and on outputs to reduce the risk of brand damage and decision making based on faulty outputs.

Drivers

- Security teams are still catching up with ongoing AI application initiatives and seeking AI runtime controls that they could easily deploy.
- GenAI is driving significant security investments, but anomaly detection and security controls can apply more broadly to AI applications and models.
- AI applications require monitoring to ensure compliant implementation, avoid application abuses and prevent malicious activity.
- The emergence of AI agents reinforces the need for runtime protection and anomaly detection. Moreover, it adds low latency and even lower false positive requirements.
- Open-source offerings, such as Meta's Llama Guard, are available for commercial use with

certain limitations.

- Early-stage startups face increased and intense competition from large, established incumbent providers providing similar features, such as security service edge (SSE). But today, specialized providers offer differentiators, such as getting access to system prompts and raw model outputs, or applying machine learning techniques to analyze the intent of these inputs and outputs.
- Insurance requirements, upcoming regulations (such as the EU AI Act and several U.S. laws) and existing enterprise policies potentially mandate the adoption of application security practices. These practices include preventative controls and monitoring to support risk mitigation for high-risk AI systems.

Obstacles

- New AI control tools are difficult to integrate with existing security and other monitoring systems. They often require building new playbooks and triage capabilities, especially for nonsecurity events (legal, compliance, acceptable use).
- AI runtime defense tools are often integrated into broader AI governance platforms, combining testing and risk assessments with runtime monitoring. This makes it more difficult for application security leaders to implement and tune for their specific purpose.
- Real-time security alerts on text-based inputs can be noisy and prone to a high false-positive rate, for example, when detecting SQL injection and XSS.
- Many providers are small and growing, and the effectiveness of their product remains unproven.
- The deployment form factor may influence what the controls can see; deployed in front of the applications, they see only user inputs and application outputs. Multiple entry points in the application stack may be necessary, especially in multimodal AI applications.

- Organizations may have privacy concerns about inspecting application input and output, especially when it implies sharing sensitive content with the provider's SaaS platform.

User Recommendations

- Develop and share an understanding of AI and security risks by leveraging Gartner's research on AI trust, risk and security management (AI TRiSM) and industry frameworks, such as the OWASP Top 10 for Large Language Model Applications. For more details, see **Market Guide for AI Trust, Risk and Security Management**.
- Request benchmarks, test data and test results to evaluate the provider's commitment to quality and see how much due diligence work will be necessary to estimate false-positive and false-negative rates.
- Develop or acquire AI model testing capability to complement runtime controls and independently evaluate their efficacy.
- Develop checklists and test plans to continuously evaluate the ability of these security controls to identify new attacks.
- Evaluate providers for their ability to serve multiple AI security use cases, in addition to specialized LLM input and output monitoring.

Sample Vendors

Aim Security; CalypsoAI; Cranium; HiddenLayer; Lasso; Noma Security; Pillar Security; Prompt Security; Protect AI; TrojAI

Gartner Recommended Reading

Use TRiSM to Manage AI Governance, Trust, Risk and Security

Market Guide for AI Trust, Risk and Security Management

How to Secure Custom-Built AI Agents

Crypto-Agility

Analysis By: Mark Horvath

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

Crypto-agility is the capability to transparently swap out encryption algorithms and related artifacts in an application, replacing them with different and, presumably, safer algorithms.

Why This Is Important

- Mandates and regulations (such as PCI DSS 4.0.1, DORA Commission Delegated Regulation [EU] 2024/1774 Section 6.4 and the proposals to increase the amount of certificate renewals) often require changes to existing cryptographic systems and policies. This increasing scale and complexity demands more scalable and dynamic cryptographic management.
- The National Institute of Standards and Technology (NIST) has standardized its first suite of quantum-safe alternatives, CRYSTALS-Kyber, CRYSTALS-Dilithium, SPHINCS+, and FALCON (for ML-KEM, ML-DSA, SLH-DSA, and FN-DSA standards respectively), for widespread use. HQC, an alternative to lattice, is scheduled to be approved in 2027. These algorithms will form the core of crypto-agile alternatives for vendors and developers going forward.

Business Impact

Given the increased demands for classical cryptography to achieve operational efficiencies and cost reductions (e.g., agility through policy and automation), and the coming threat of quantum computing, Gartner expects crypto-agility to be a significant differentiator for technology vendors. Businesses impacted by the changes to cryptography will need to evaluate and execute the following activities:

- Applications using encryption or public-key infrastructure (PKI) need visibility and the ability to automatically update or change by policy. Data about algorithms, key sizes, expiration dates and uses must be tracked with a metadata database.
- Suitable replacement implementations must be identified for applications and use cases that match performance and security expectations.
- New algorithms are not drop-in replacements for existing cryptography, so alternatives must be tested and replacement policies must be generated.
- Vendor cryptography products must be identified, and vendors will need to provide some level of crypto-agility consistent with the business objectives.

Drivers

- Cryptography in vendor products will need to be identified as the regulatory and security needs of the business evolve, and vendors will need to provide a schedule for updates and replacement.
- Most organizations that have inventoried their cryptographic metadata have found they already have considerable PKI technical debt (e.g., expired certificates, deprecated algorithms, short keys). Cleaning that up will substantially lower the organization’s risk profile.
- The development of quantum computers capable of damaging cryptography through

Shor's or Grover's algorithms is estimated to be five to seven years away. This is much shorter than the typical life span of sensitive information found in most organizations, leading to data exposure of sensitive data. Keybreaking with Shor's algorithm goes linearly with key size, but is limited by the number of available qubits; once the algorithm is running, larger key sizes will fall quickly, leading to a short runway to implement changes.

- Early adopters of new algorithms have found that performance can vary significantly with the implementation. Most have reported race conditions, slower performance and other issues related to timing. Moving applications to a crypto-agile posture is a good way to test implementations to find the right mix of performance and security.
- Government organizations are beginning to require a quantum-safe encryption strategy for vendors selling to them (e.g., U.S. NSM-10, EO-14028). This includes all products or code in the final product, including open-source software (OSS) and other vendor products. As with a software bill of materials, this significantly expands the scope of these orders to include many vendors not otherwise selling directly to the government.
- New algorithms have additional uses that can foster novel business cases (e.g., stateful signatures, homomorphic encryption).

Obstacles

- Cryptographic technical debt (e.g., use of hard coded secrets and algorithms, deprecated algorithms, etc.) often have very low visibility in IT, and thereby liable to be undervalued.
- Many organizations don't know how to inventory their cryptographic usage, relying on vendors or struggling themselves.
- Crypto-agility is fundamentally a developer-driven effort, but developers often lack architectures, patterns, libraries and other artifacts needed to successfully implement crypto-agile applications.

- Many organizations lack the expertise needed to lead this kind of cryptographic hygiene, delaying action and becoming blind to risks.
- Most cryptographic systems have source/destination dependencies and stakeholders have difficulty orchestrating change across all dependent parties in a coordinated and mutually beneficial way.

User Recommendations

- Start building a cryptographic inventory sooner rather than later. This will help identify key systems and data, allowing a controlled rollover to newer encryption and help to prioritize technical debt.
- Experiment with various libraries and implementations of the NIST algorithms to determine the effects on your infrastructure.
- Create standard patterns, policies, architectures and other developer artifacts to make it easier to integrate crypto-agility into existing development plans.
- Ask vendors what their plans are for replacing algorithms, and when quantum-safe versions of their products will be available.
- Evaluate crypto dependencies in applications and favor crypto-agile implementations in both vendor purchases and in creating custom-based applications.
- Automate key and certificate management and favor a policy-based approach for key applications so they can be secure even as the market evolves.

Sample Vendors

Crypto4A; Cyberark; DigiCert; Entrust; IBM; InfoSec Global; ISARA; Keyfactor; Sectigo; Thales

Gartner Recommended Reading

Top Strategic Technology Trends for 2025: Postquantum Cryptography

Postquantum Cryptography: The Time to Prepare Is Now!

Secure Your Network Against "Harvest Now, Decrypt Later" Quantum-Enabled Attacks

Leaders' Guide to Quantum Computing

Penetration Testing as a Service

Analysis By: Mitchell Schneider, William Dupre, Carlos De Sola Caraballo

Benefit Rating: Moderate

Market Penetration: 20% to 50% of target audience

Maturity: Adolescent

Definition:

Penetration testing as a service (PTaaS) provides technology-led, point-in-time and continuous application and infrastructure testing aligned with penetration testing (pentesting) standards, which have traditionally relied heavily on human pentesters using commercial/proprietary tools. The service is delivered via a SaaS platform, leveraging a hybrid approach of automation and human pentesters (crowdsourced or vendors' in-house team) to increase the efficiency and effectiveness of the results.

Why This Is Important

Pentesting is foundational in a security program and mandated by various compliance

standards and regulations. PTaaS delivers continuous security testing via a platform that enables faster scheduling and execution of pentests, and real-time communications with testers and visibility of test results. It provides API access to enable integration with existing DevOps and ticketing solutions for workflow automation. It also provides the ability to document and track pentesting results to demonstrate progress over time to leadership/auditors.

Business Impact

PTaaS complements exposure assessments and traditional application security testing. It also provides cost optimization and quality improvement of pentesting output along with validation of exposure status. PTaaS enables organizations to elevate their security posture with continuous assessments that integrate validation earlier in the software development life cycle as compared with traditional pentesting efforts. It gives access to real-time findings delivered through a platform, which accelerates remediation and improves collaboration efforts.

Drivers

- Organizations are turning to PTaaS to deal with the increase in attack surfaces due to the accelerating use of public cloud and expansion of public-facing digital assets. PTaaS allows developers to talk to and receive guidance from pentesters instead of relying completely on scanners, such as dynamic application security testing/static application security testing (DAST/SAST) scanners.
- Organizations with limited in-house security expertise must meet their compliance and risk management objectives, in addition to improving their security posture, and therefore look to PTaaS offerings to meet these initiatives.
- In order to meet fast production deadlines, security-aware organizations must integrate a more agile way of conducting pentesting into their continuous integration/continuous

delivery (CI/CD) pipelines for their DevSecOps practices.

- Gartner clients express an appetite to test on a more frequent basis to support their continuous threat exposure management (CTEM) initiatives. However, manual pentesting is cost-prohibitive in modern infrastructure (for example, infrastructure as a service [IaaS], SaaS and third-party subscriptions).
- Most organizations have a pentesting budget and often seek better ways to use their annual budget. Highly automated, technology-led pentesting has the potential to offer higher-quality deliverables for the price, or at least more frequent deliverables for the price.
- Engaging external security researchers for penetration testing can present legal challenges, such as ensuring proper authorization and compliance with computer crime laws. Additionally, manually managing payments for discovered vulnerabilities can be complex and time-consuming,

Obstacles

- Selecting a suitable PTaaS vendor in the market may be difficult, as their capabilities vary. Vendors use one or a combination of automation and human testers, which are in-house or community-led — typically vetted freelancers — to perform penetration testing for the client organization.
- Most PTaaS vendors in the market focus on internet-facing digital assets, like web and mobile applications, and APIs that may only partially fulfill client requirements.
- PTaaS vendors may not be able to support very complex environments where extensive domain expertise is needed.
- The depth and extensibility of a PTaaS is not as flexible as a statement of work (SOW)-led engagement. Therefore, if you have some special requests, and/or are seeking extensive

testing, you are not going to get it with PTaaS.

- PTaaS overlaps with adversarial exposure validation (AEV), which is an adjacent market, yet they are different in terms of adoption and operation. AEV focuses on continuous, real-world attack simulations, while PTaaS emphasizes human expertise and integration with development processes for on-demand or continuous testing.

User Recommendations

- Determine which option/mix of penetration testing programs is best for your organization: compliance-driven service engagement; PTaaS; in-house red team leveraging an automated pentesting tool; or bug bounty.
- Identify and evaluate the pentesting scope and requirements that PTaaS vendors will be able to fulfill before engaging with vendors. PTaaS is well-aligned to both application testing and external infrastructure testing. Not all vendors will be able to replace internal infrastructure pentests, wireless, social engineering and physical assessments.
- Favor hybrid scanning models that combine human analysis and automation to increase both effectiveness and efficiency.
- Select a PTaaS vendor that aligns with relevant compliance requirements, and not focused only on internet-facing infrastructure and applications.
- Seek PTaaS vendors that provide customized and tailored guidance throughout the life cycle of their service to alleviate the security skills gap.

Sample Vendors

Bishop Fox; BreachLock; Bugcrowd; Cobalt; HackerOne; NetSPI; Praetorian; Siemba; Synack; TrollEye Security

Gartner Recommended Reading

Innovation Insight: Penetration Testing as a Service

How to Select the Right Penetration Testing Provider for Your Organization

Strengthen Critical Applications With an Effective Penetration Testing Program

Sliding into the Trough

Software Bill of Materials

Analysis By: Jason Gross

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

A software bill of materials (SBOM) is a structured, machine-readable document that provides a nested inventory of all software components (open-source or proprietary) used to build a software package. SBOMs offer visibility into potential risks within the software supply chain by providing detailed metadata about each component they contain. They enable transparency, auditability and traceability, facilitating better risk management and collaboration between systems and organizations.

Why This Is Important

Gartner estimates that 60% to 80% of the code in new software projects originates from

third parties, with most coming from open-source software (OSS) projects. This trend is driven by the productivity gains of code reuse; however, it comes with notable trade-offs. In recent years, the number of OSS vulnerabilities and supply chain attacks has risen significantly, with yearly increases reaching double- and triple-digit levels. SBOMs provide the visibility organizations need to manage these risks.

Business Impact

Organizations that have developed and integrated SBOM capabilities into their workflows improve their risk posture through efficient vulnerability mitigation, streamlined regulatory compliance, and improved transparency during internal development efforts and while performing M&A due diligence. SBOM also empowers procurement teams with software supply chain risk insights to inform purchasing decisions.

Drivers

- **Increasing regulatory requirements:** Regulatory authorities worldwide are introducing requirements aimed at providing transparency into, and reducing risk within, the software supply chain. In the U.S., agencies like the FDA and CISA require or strongly encourage SBOMs as prerequisites for transacting with them. Outside the U.S., similar initiatives are being adopted by the European Union (e.g., EU Cyber Resilience Act [CRA]), Japan's Pharmaceuticals and Medical Devices Agency (PMDA), and Australia's Therapeutic Goods Administration (TGA), to name a few. These global efforts highlight the importance of SBOMs in understanding and managing software supply chain risks.

- **Investor and board-level focus on software risk:** Increased regulation and the rise of high-profile breaches have made cyber risk a priority at the board and investor level. Security leaders are increasingly expected to provide clear visibility into software risk and are being held accountable for managing exposures associated with the software supply chain. SBOM adoption is seen as a reflection of cyber maturity, demonstrating a commitment to transparency, effective risk mitigation capabilities, and readiness to

address regulatory and stakeholder expectations.

- **Customer and market expectations:** Customers increasingly demand transparency into the supply chain risks associated with the software they buy or integrate with. This pressure is reinforced by an industrywide movement toward secure-by-design practices, driven by standards bodies and open-source initiatives, such as OpenSSF, OWASP and CISA. Many enterprise buyers are beginning to require SBOMs during software renewal and procurement processes, making transparency a competitive differentiator. As adoption continues to grow, organizations unable or unwilling to provide SBOMs risk losing customer trust and market share.

Obstacles

- **Technical complexity:** Each released artifact version needs an SBOM, as OSS components can change between builds. Cloud-native, loosely coupled architectures lead to SBOM sprawl, forcing SBOM artifact consolidation into composite, app-level SBOMs. This demands tight workflow integration and scalable methods for SBOM distribution, consumption and trust validation.
- **Organizational impediments:** Many organizations struggle to integrate SBOM use cases into existing processes, which hinders ROI demonstration and leads to deprioritization of SBOM initiatives. Limited expertise and legal concerns over IP and licensing further restrict SBOM sharing.
- **Tooling ecosystem challenges:** Despite growing adoption, the SBOM tooling landscape remains fragmented and immature. Many tools cannot merge SBOMs from multiple artifacts or integrate with processes that support SBOM use cases. Inconsistent standards, weak trust mechanisms (e.g., signing and verification) and poor distribution further limit SBOM utility.

User Recommendations

- Integrate SBOM generation into CI/CD workflows to ensure that every build artifact has an up-to-date SBOM.
- Request SBOMs from software vendors and third-party suppliers as part of procurement, renewal or security review processes.
- Define and prioritize SBOM use cases and update supporting processes to develop ROI projections.
- Engage vendors to address SBOM capability gaps by advocating for roadmap enhancements or evaluating alternative solutions, where needed.

Sample Vendors

Apiiro; Codenotary; Cybeats Technologies; Cycode; Eracent; Finite State; Revenera; ReversingLabs; Sonatype; SOOS

Gartner Recommended Reading

Emerging Tech: A Software Bill of Materials Is Critical to Software Supply Chain Management

Innovation Insight for SBOMs

How Software Engineering Leaders Can Mitigate Software Supply Chain Security Risks

Mitigate Enterprise Software Supply Chain Security Risks

Application Security Posture Management

Analysis By: Giles Williams, Dionisio Zumerle

Benefit Rating: Transformational

Market Penetration: 20% to 50% of target audience

Maturity: Adolescent

Definition:

Application security posture management (ASPM) tools continuously manage application risk through collection, analysis and prioritization of security issues from across the software life cycle. They ingest data from multiple sources, correlate and analyze findings for easier interpretation and triage. They enable the enforcement of security policies and facilitate the remediation of security issues while offering a comprehensive view of risk across an application.

Why This Is Important

Complex, modern applications create siloed security data across development and operations, making visibility and control exceptionally difficult. ASPM supports integration of security tools and data across the entire DevOps environment, enabling organizations to implement security policies and processes. Key benefits include prioritization and triage of findings, ensuring a focus on the most critically impactful issues while assessing risk in terms meaningful to all stakeholders.

Business Impact

ASPM aids security and software engineering teams by integrating application security tools and controls, improving visibility and control, and enabling the measurement and management of risk. Triage of application security data brings increased productivity by prioritizing resources to focus on the most critical issues. ASPM delivers clarity and improved insights, both from an operational and risk-oriented view, into an application’s security

status.

Drivers

- AppSec and software delivery teams struggle with prioritizing remediation and mitigation efforts throughout the software development life cycle (SDLC) due to increased application complexity and the rapidly growing volume of data provided by application security tools. ASPM solves this challenge by ingesting information from multiple sources across the SDLC, correlating results and centralizing visibility.
- Prioritization capabilities aid in gaining acceptance and support for security efforts among engineering teams. Too often, these teams are inundated with siloed data from multiple tools. Consolidating information, evaluating its validity and prioritizing remediation is a cumbersome process that doesn't scale well manually to address the amount of data and the speed of engineering processes.
- Security and engineering teams have difficulty in reporting the risk postures of applications, without meaningful business metrics and threat intelligence. ASPM products can assist in translating raw vulnerability data into a form more relevant to executives and application owners.
- In organizations with diverse development and deployment processes, establishing automated controls for policy enforcement is complex, leading to a tendency to establish and enforce a "one size fits all" approach to policy definition. ASPM's integration and intermediation capabilities between application development and deployment processes and security controls offer the ability to centralize management of those controls and allow for more granular approaches to enforcement.

Obstacles

- Effective automation of security testing presumes an understanding of the overall risk of

an application and the types of testing needed. Without this understanding, efforts to articulate policies on which prioritization and triage efforts will rely are complicated and lead to reduced benefits from ASPM tools.

- Some vendors offer integration exclusively with either development or operations security tools. This presents a barrier to delivering a “full stack” view of an application’s security risks.
- ASPM tools work by aggregating and analyzing data, and some level of abstraction is inherent. That poses a risk that critical information may be inadvertently overlooked in processing, yielding the potential for false positives or a false sense of security.
- While users are well aware of the problems ASPM can potentially address, they may not be aware of the existence of tools. This leads to delayed evaluation and adoption.

User Recommendations

- Prioritize ASPM in organizations with diverse development teams and a wide assortment of security tooling.
- Evaluate vendor viability and direction, given merger and acquisition activity within the market.
- Identify key stakeholders who will use an ASPM solution. Because underlying processes and outputs will affect many parts of the organization, stakeholder support is crucial to success in implementation and use.
- Evaluate the ability of tools to scale to process the amount of data generated across the application life cycle.
- Examine native ASPM capabilities (for example, as a feature of application security testing or other security tools) as an alternative to an investment in a dedicated solution. Such offerings may be effective in more homogeneous environments.

Sample Vendors

Apiiro; ArmorCode; Cycode; DefectDojo; Kondukto; Legit Security; OX Security; Phoenix Security; PointGuard AppSOC; Snyk

Gartner Recommended Reading

Innovation Insight: Application Security Posture Management

Guide to Application Security Concepts

Market Guide for DevOps Continuous Compliance Automation Tools

Adapting Cybersecurity for the Agile Enterprise

Software Supply Chain Security

Analysis By: Manjunath Bhat

Benefit Rating: Transformational

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

Software supply chain security (SSCS) is about building secure software by protecting against compromised code, tooling, identities and pipelines during development, delivery and postdeployment. SSCS reduces third-party risks through policy-based curation of dependencies, software composition analysis (SCA) and software bill of materials (SBOM) inspection. They ensure artifact provenance and traceability with signing and verification as

they pass through development and delivery pipelines.

Why This Is Important

Software supply chains transcend organizational boundaries and consist of external entities in addition to internal systems. Internal systems include software delivery pipelines, software dependencies and software development environments. External entities include partners, open-source software (OSS) and vendors. Organizations have greater control over internal systems and little to no control over external entities. SSCS protects organizations both from insider threats as well as external entities that are potentially exploitable.

Business Impact

- Identifies and mitigates security and compliance risks arising from the widespread use of open-source software.
- Reduces developer friction and lost productivity due to attacks on tools, environments, pipelines and infrastructure used for software development, delivery and runtimes.
- Satisfies governance and regulatory requirements by making the software delivery infrastructure auditable by automating enforcement of application security policies.

Drivers

- **State-sponsored attacks:** Much of open-source software is vulnerable to infiltration by nation-state bad actors. State-sponsored software supply chain attacks have, therefore, become increasingly sophisticated and prevalent, targeting various sectors globally. These attacks have either used vulnerabilities in the software itself (as in the case of Kaseya) or compromised the software development life cycle (SDLC) (as in the cases of SolarWinds and NotPetya attacks). SSCS becomes critical to reduce these risks.
- **Regulatory compliance and government mandates:** Governments, policymakers and

regulatory bodies around the world are mandating third-party supplier assessments, continuous vulnerability scanning and software bills of materials to build a trusted software supply chain. Examples of mandatory regulations include the **United States Government Cybersecurity Executive Order 14028**, **EU Cyber Resilience Act**, **EU NIS2 Directive**, and the **Federal Food, Drug, and Cosmetic (FD&C) Act**.

- **Pervasive use of open source and reliance on third-party software:** Every software application uses third-party code by way of open-source dependencies. Based on hundreds of analyst-client interactions, Gartner believes that more than 95% of organizations are using OSS, even if they are not aware of it. According to the 2025 BlackDuck Open Source Security and Risk Analysis Report, 97% of the codebases contained open source. The same report finds 86% of audited applications contained open-source vulnerabilities, with 81% of the applications containing high- or critical-risk vulnerabilities.

- **Use of open-weight AI models:** The ease of access to open-weight large language models (LLMs) and the low barrier to integrating them in applications add a new dimension to software supply chain risks. The Gartner Software Engineering Survey for 2025 shows 47% of respondents integrating LLMs into existing applications. SSCS helps to discover use of LLMs, scan models for risks and enforce policies to prevent the use of unapproved LLMs. Examples of risks include weak model provenance, unsupported models and geopolitical barriers preventing model use.

Obstacles

- The majority of organizations lack a complete understanding of SSCS, and thus have yet to adopt a comprehensive approach to manage software supply chain risks. For example, many organizations are focused on acquiring SBOMs but have yet to establish how those artifacts will be evaluated, stored and used.

- Efforts to secure the integrity and provenance of software artifacts throughout the

software supply chain are emerging but are uneven in scope, execution and adoption. Implementing policies that determine what dependencies are allowed can introduce friction and require negotiation of processes with affected stakeholders (software engineering, application security and platform engineering).

- The adoption of capabilities to harden DevOps pipelines through artifact integrity validation and automated policy enforcement are comparatively lower. This is because of the historically poor developer experience associated with signing and verification workflows in continuous integration/continuous delivery (CI/CD) pipelines. The heterogeneity of tools in the DevOps pipeline exacerbates the challenge of creating artifact attestations and ensuring pipeline integrity.

User Recommendations

- Enhance trust in your software supply chains by improving visibility, protecting integrity and enhancing security posture throughout the SDLC.
- Reduce visibility gaps in the software supply chain by using software composition analysis and software bills of materials to manage third-party risks as well as ensuring auditability and traceability across pipeline activities and interactions in the SDLC.
- Protect software integrity throughout the software delivery process by signing and verifying build artifacts, ensuring provenance data is in place and preventing use of noncompliant artifacts.
- Improve the security posture of the software delivery process by automating policy enforcement in the SDLC as well as detecting and resolving misconfiguration errors in DevOps tooling.

Sample Vendors

ActiveState; Anchore; Apiiro; Aqua Security Software; Arnica; BlueFlag Security; Boost

Security; Cycode; Endor Labs; GitHub

Gartner Recommended Reading

Market Guide for Software Supply Chain Security

Mitigate Enterprise Software Supply Chain Security Risks

How Software Engineering Leaders Can Mitigate Software Supply Chain Security Risks

Innovation Insight for SBOMs

Serverless Function Security

Analysis By: Feng Gao

Benefit Rating: Moderate

Market Penetration: 1% to 5% of target audience

Maturity: Emerging

Definition:

Serverless function security provides unique and differentiated controls to ensure serverless functions (also known as function platform as a service, or fPaaS) are developed, deployed and operated securely. Comprehensive solutions start with proactive vulnerability and configuration scanning in development, entitlement and access checking — typically combined with lightweight runtime protection and behavioral analysis.

Why This Is Important

Serverless functions are available from all hyperscale cloud providers, and may be included in cloud-native applications as a simple and scalable way to enable an event-driven microservices architecture. These services appeal to developers, and require proper protection in the development phase due to their ephemeral nature. Serverless function security provides security capabilities where traditional controls are hard to secure.

Business Impact

By ensuring the security and compliance of serverless functions, starting from code development, information security organizations can securely enable the developer-driven adoption of these technologies while minimizing any business impact or operational slowdown. In the longer term, serverless functions have the potential to improve overall enterprise security profiles by migrating the responsibility for significant elements of the attack surface to the hyperscaler, rather than the enterprise.

Drivers

- The adoption of serverless functions as elements of cloud-native applications is increasing.
- Serverless functions have a differentiated attack surface driving the need for security capabilities, such as software composition analysis, vulnerability scanning, API security, runtime protections, and correct and compliant serverless platform as a service (PaaS) configuration.
- Serverless function security is uniquely positioned to help organizations prevent, detect and respond to any attacks that exploit issues related to serverless functions, as it provides visibility and security controls into the environment.
- Cloud permissions are extremely complex. Serverless function security allows for detection and remediation of overly permissioned functions that increase risk.

Obstacles

- Information security is often blind to the use of serverless functions and unaware of the risks they pose. Thus, they deprioritize security for these functions. Additionally, few attacks on serverless code have been clearly documented, meaning that any perceived risk is low for the effort and money invested.
- Serverless function security must have minimal friction for developers to avoid its adoption being disrupted by the developer community.
- Serverless function consumes input from various event sources, like APIs, IoT device connections and message queues. This requires protection against threats from untrusted or malformed event triggers.
- Very few runtime protection options are available, which fall short of injecting or wrapping serverless functions with runtime protection code.
- Serverless security tools are still maturing. As such, standards for secure deployment across multiple platforms are yet to be defined.

User Recommendations

- Engage with development teams to understand the scope of serverless function's current and planned usage. Run a discovery project to see if serverless code is in use that you aren't aware of.
- Scan proactively for vulnerabilities, vulnerable open-source code and misconfiguration during development, as you would for any static application code.
- Provide risk visibility and configuration/permissions monitoring of the entire hyperscale cloud provider configuration, including serverless, through your cloud security posture management and cloud-native application protection platform tools.

- Adopt a least-privilege security posture, including serverless function permissions and network connectivity. Automate the discovery of overprivileged use of serverless functions and reduce it to the least possible. Isolate functions and strongly manage serverless function resource access.
- Deploy an API gateway, proxy or event broker for invocation, providing a visibility and control point.

Sample Vendors

Aqua Security; Check Point Software Technologies; Contrast Security; Data Theorem; Palo Alto Networks; Rapid7; Snyk; Sysdig; Trend Micro

Gartner Recommended Reading

Market Guide for Cloud-Native Application Protection Platforms

How to Make Integrated IaaS and PaaS More Secure Than Your Own Data Center

API Threat Protection

Analysis By: Mark O'Neill, Dionisio Zumerle

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

API threat protection is the defense of web APIs from exploits, abuse, access violations and

denial of service (DoS) attacks. Specialist API threat protection tools, API gateways, and web application and API protection (WAAP) protect APIs by providing visibility, a combination of content inspection of API parameters and payloads, traffic management and anomaly detection.

Why This Is Important

Organizations use APIs to provide access to data and application functionality. APIs are easy to expose but difficult to defend. This creates a large and growing attack surface, leading to a growing number of API attacks and breaches. Successful attacks on APIs exploiting common vulnerabilities like broken authorization may result in data breaches, leading to loss of sensitive data as well as reputational harm. A successful DoS attack on an API will result in a lack of availability, which, in turn, brings reputational loss.

Business Impact

When APIs are typically used for accessing data or business capabilities, often linked to systems of record, the impact of an API breach can be substantial. Privacy regulations typically require reporting if private data is breached through an unsecured API. APIs are easily consumable, so a vulnerability can leak large volumes of data. The challenge of distinguishing malicious access from valid access further complicates the task of securing APIs.

Drivers

- A persistent stream of publicized API breaches continues to drive awareness of the need for API threat protection.
- Certain industries, including financial services and e-commerce, involve APIs that are high-value by nature and, therefore, vulnerable to abuse and fraud.
- Some established vendors have started using machine learning to detect potentially

harmful API usage patterns and thus protect APIs from threats. These include WAAP vendors.

- Generative AI (GenAI) application access to APIs, including via the new Model Context Protocol (MCP), drives the need to secure access to APIs. Because many APIs provide access to data, a successful API breach is particularly harmful, which highlights the need to secure APIs — including those using new standards such as MCP.

Obstacles

- Organizational ownership of API threat protection is a challenge as the security team, under a chief information security officer, typically manages a web application firewall (WAF), while API gateways are typically managed by API teams. This can lead to API threat protection being neglected because of a lack of expertise.
- Many API security issues are related to business logic. Protection against business logic threats is difficult to automate completely because the protection tool needs to understand the logic and identify unusual usage.
- Behavioral anomaly detection is, by nature, a generator of false positives. Despite the growing use of GenAI, most API threat detection tools currently require human intervention to process false positives.

User Recommendations

- Discover and categorize your APIs first, since you can’t protect what you don’t know.
- Implement a layered API security model. At the outer (typically cloud-based) layer, use volumetric distributed DoS protection and bot detection. Behind this layer, apply API-specific authentication, authorization and protection mechanisms.
- Assess the API protection provided by your current WAF vendor. If it provides immature or

insufficient API protection, then investigate API threat protection specialist vendors.

- Ensure that the API protection rules in your chosen API threat protection solution are adaptable, based on the nature of the API itself. Static rate limits or Internet Protocol allow/block lists are rarely useful in production environments or at scale.

Sample Vendors

42Crunch; Akamai; AppSentinels; Cequence Security; Cloudflare; F5; Harness (Traceable); Imperva; NSFOCUS; Salt Security

Gartner Recommended Reading

Market Guide for API Protection

API Security Maturity Model

Innovation Insight for API Protection

Implement Effective Application and API Security Controls

Web App Client-Side Protection

Analysis By: Esraa ElTahawy

Benefit Rating: Moderate

Market Penetration: 1% to 5% of target audience

Maturity: Adolescent

Definition:

Web app client-side protection defends against application-level attacks that aim at exploiting client-side vulnerabilities. Innovations in this space focus on securing interactions between the user's browser and the web application, protecting against attacks such as cross-site scripting (XSS) and cross-site request forgery (CSRF) and ensuring the integrity of scripts loaded in the browser.

Why This Is Important

Client-side attacks have proven to be particularly impactful as they exploit the increasingly decentralized design of modern applications. In particular, single-page applications migrate the control and software logic to the client side, where it is exposed to attacks. For example, by injecting malicious scripts into JavaScript applications, attackers have lured thousands of visitors to banking and online commerce websites to hand over their credit card information.

Business Impact

Companies in industries such as financial services, ticketing and online retail have experienced a proliferation of fraudulent script injections into their web applications. These attacks steal customer payment information and expose organizations to significant fines. Organizations have turned to client-side security recently to protect their applications from these types of attacks.

Drivers

- Modern web applications rely heavily on third-party code executing directly on a client's web browser. This opens up clients' web browsers — and, in turn, the end-user device itself — to a wide range of attacks such as java script injection and browser-focused attacks.
- Client-side attacks, such as those carried out by the Magecart groups, have impacted online retail, airline and ticketing websites. Attackers inject malicious scripts that lure

website visitors to hand over their credentials or payment details.

- New PCI DSS v4.0.1 requirements now mandate the inclusion of client-side protection capabilities
- Cloud web application and API protection (WAAP) providers have been introducing more advanced client-side protection capabilities to fulfill client and regulatory requirements.

Obstacles

- The lack of clear ownership of this technology slows down adoption. Three different entities in enterprises — development, security and line of business — are involved in the buying process.
- Many organizations still lack awareness of threats related to the client side. There's also a lack of awareness around protection mechanisms that can help secure the client side.
- Concerns about performance impacts, possible customer privacy issues and false positives that block business traffic can discourage enterprises from enabling client-side protection features.

User Recommendations

- Identify applications that might benefit from client-side security, such as public-facing JavaScript web applications that contain software logic on the client side. More broadly, single-page web applications and mobile web apps will also be good candidates to receive this sort of protection.
- Start utilizing your current cloud WAAP vendor's client-side protection capabilities. Conduct a gap analysis to identify any missing mandatory capabilities against the new PCI DSS v4.0.1 mandates and use the results of that analysis to decide if a stand-alone client-side protection tool should be assessed.

- Focus on tools that can help you discover existing and newly added scripts, automate blocking of suspicious scripts and data exfiltration requests, as well as real-time detection of scripts' modifications.

Sample Vendors

Akamai; CHEQ (Ensighten); Cloudflare; c/side; Feroot Security; HUMAN; Jscrambler; Microsoft; Source Defense; Thales (Imperva)

Gartner Recommended Reading

Market Guide for Cloud Web Application and API Protection

Invest Implications: Market Guide for Cloud-Native Application Protection Platforms

Threat Modeling Automation

Analysis By: William Dupre

Benefit Rating: High

Market Penetration: 20% to 50% of target audience

Maturity: Early mainstream

Definition:

Threat modeling automation tools automate the creation of security requirements and threat models. Such tools highlight potential security ramifications of application architectures and recommend secure coding practices or architectural mitigations. They also can manage threat libraries, track mitigations and integrate security analysis into the software development life cycle (SDLC).

Why This Is Important

Threat modeling to surface security requirements is key to efforts to create applications that are secure by design. Automated tools help facilitate this process by more easily incorporating security into the design phase of application development. Although they do not secure applications, automated tools help in the creation of secure application architectures and ensure identification of appropriate and specific security requirements.

Business Impact

Threat modeling automation tools significantly decrease the effort required to create and maintain threat models, security requirements and risk assessments. This ensures early definition of security specifications that are specific to individual projects — while costs and risks are low, rather than later in the development process. This approach offers significant benefits to multiple groups within an organization, including architects, developers, security teams and even business stakeholders.

Drivers

- Organizations of all kinds continue to struggle to create secure applications. Issues include inadequate security capabilities, such as authentication, access control and data protection, and fundamental security design flaws, all of which leave applications vulnerable to attack.
- Secure-by-design initiatives and secure software compliance mandates make threat modeling an essential best practice. Automated threat modeling tools make the practice faster and more scalable. They also help ensure that specific requirements associated with mandates and regulations are addressed.
- Modern applications — incorporating distributed cloud-native technologies, increased use of internal and third-party APIs, and forms of AI — are more complex and, as a result, prove difficult to manually and accurately model for threats. Threat modeling automation

speeds these processes and helps modelers identify all threats and relevant countermeasures.

- The rapid pace of development limits the time and resources available for threat modeling. Iterative approaches to development, such as agile practices, mean that threat models must be updated more frequently. Both factors strain manual approaches and increase the likelihood that threat modeling will be limited in scope or simply skipped entirely.

Obstacles

- Ability to accurately represent a rapidly changing application remains a weak spot of threat modeling automation tools. Most of today's tools require user intervention to update models as applications change, which leads to abandonment. This is improving as vendors begin to link systems directly to cloud platforms or infrastructure-as-code files, ensuring that changes are reflected automatically in the model, which will then automatically produce updated guidance.
- Capabilities vary. Free and open-source tools enable easy adoption but fall short when modeling more complex systems. This prompts consideration of commercial tools, though limitations constrain suitability for the most advanced users.
- Most organizations still focus on application security testing as they establish an application security program. These tools are essential to identify vulnerabilities in code during the development stage but fail to identify design flaws during planning.

User Recommendations

- Treat threat modeling and security requirements generation as primary practices within a secure SDLC. Threat modeling automation tools can help accelerate these tasks while incorporating threat and security requirement knowledge from tool vendors.

- Use these tools to automate manual or overlooked efforts. Doing this ensures that threat modeling and security requirements generation activities are incorporated into the development workflow. Consider integrating test cases to confirm that requirements are met.
- Evaluate emerging AI-based capabilities that support threat modeling. Although still emerging, AI offers the potential for improved analysis and threat identification and for simplified interaction with the tools.
- Train development, engineering, operations and architectural staff in the use and value of these tools. Encourage their use early and continuously in the development process and after deployment to validate application threat protection efforts.

Sample Vendors

Aristiun; IriusRisk; SecureFlag; Security Compass; ThreatModeler; TrustOnCloud; Tutamantic Sec

Gartner Recommended Reading

An Introduction to Threat Modeling Best Practices

The Art of Threat Modeling

12 Things to Get Right for Successful DevSecOps

API Security Testing

Analysis By: Dionisio Zumerle, Giles Williams

Benefit Rating: High

Market Penetration: More than 50% of target audience

Maturity: Adolescent

Definition:

API security testing is a specialized type of application security testing (AST) that identifies vulnerabilities in APIs. Checks should include traditional application vulnerabilities (such as injection attacks) and API-specific issues (e.g., broken-object-level authorization). Discovery capabilities are sometimes supported to ensure that unknown APIs are identified and tested for vulnerabilities.

Why This Is Important

APIs represent a major attack surface for web-enabled applications. Attacks on and abuse of APIs result in serious adverse consequences, including data breaches and other security incidents. DevSecOps teams focus on the need for API security testing in development to prevent this. Vulnerability assessment teams may test the security of production APIs.

Business Impact

APIs are a foundational element of many organizations' digital transformation efforts. Hence, securing APIs from attack and misuse is an ongoing concern for many cybersecurity leaders. Pre- and post-deployment API-specific testing builds a solid foundation for an overall API security strategy. Organizations need automated API discovery because many struggle to maintain an inventory of APIs and need help locating them so they can be tested and managed.

Drivers

- APIs, in support of digital transformation efforts, have become common in application architectures to enable information flow and support transactions between processes,

applications and systems. However, that growth has attracted more attacker attention, and APIs have become the primary attack surface for many systems.

- API attacks have resulted in data breaches and other security incidents, yielding significant damage to organizations and individuals. As a consequence, security leaders — along with the business leaders whose applications are supported by APIs — are increasingly interested in API security testing.
- Creation, development and deployment of APIs may be loosely managed, so security teams often contend with undocumented or shadow APIs that exist outside normal processes and controls. Such APIs may be deficient in secure design and testing, posing an even greater risk. Hence, the discovery of APIs deployed to production is another important need.
- API security testing helps avert the tangible and intangible costs associated with breaches and other types of security incidents.
- Traditional AST tools — static, dynamic and interactive — were not originally designed to test for some of the unique vulnerabilities associated with typical API attacks. Nor were they aimed at newer types of APIs such as GraphQL or gRPC. API-specific vulnerabilities and modern API formats prompt security and development teams to implement specialized API security tools focused on testing, discovery of shadow APIs and protection from threats, or a combination of the three.

Obstacles

- Specialized tools or penetration testing may be needed to reliably detect security flaws. APIs are susceptible to most traditional application attacks, but other attacks are common. For example, improper authorization checks have been cited in a number of breaches, and business logic tests are difficult to automate. In addition, the effort to move past traditional tools that offer inconsistent support for detecting API-specific

vulnerabilities may hinder adoption.

- Tools for testing may not support all API protocols. SOAP remains in widespread use, although REST APIs are supplanting it. GraphQL-based and gRPC-based APIs are increasingly common and require additional support from tool vendors for effective testing.
- Evaluation and selection efforts can be complicated because various types of companies, including traditional AST vendors and API security testing and protection vendors, offer products.
- The number of API endpoints typically determines licensing and pricing, which can be difficult to manage when discovery is lacking.

User Recommendations

- Begin tool evaluation by focusing on the criticality of APIs, their security and business risks and the technical requirements they pose. This will help identify needed functions and the level of testing necessary.
- Assess the suitability of the existing testing and discovery capabilities that tools in your application security portfolio provide. Many testing tools now include support for APIs, but tests may focus primarily on traditional application vulnerabilities, not those specific to APIs. API security tool capabilities — including discovery, testing and threat protection — often overlap.
- Evaluate dedicated API security testing tools in your incumbent provider offerings. The tools may also offer additional options, including audit of design-stage API specifications, discovery, vulnerability scans and threat protection.
- Complement automated testing tools with penetration testing services for comprehensive coverage, because traditional AST is often unable to detect some

common API-specific vulnerabilities.

Sample Vendors

42Crunch; Akamai Technologies; Akto; APIsec; AppSentinels; Bright Security; Equixly; Escape; Levo; StackHawk

Gartner Recommended Reading

Leaders' Guide to API Security

API Security Maturity Model

Critical Capabilities for Application Security Testing

Market Guide for API Protection

Cloud-Native Application Protection Platforms

Analysis By: Neil MacDonald, Dale Koeppen

Benefit Rating: High

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

Cloud-native application protection platforms (CNAPPs) are an integrated set of security and compliance capabilities designed to help secure and protect cloud-native infrastructure and applications across development and production. CNAPPs consolidate a large number of

previously siloed capabilities, including container scanning, cloud security posture management, infrastructure-as-code scanning and runtime workload visibility and protection.

Why This Is Important

Securing cloud environments usually requires the use of disparate tools from multiple vendors that are rarely well-integrated. This lack of integration and automation slows down developers, creates fragmented risk visibility and cross-team communication gaps, and overwhelms security professionals and developers with alerts. CNAPP offerings allow an organization to use a single integrated solution to protect the entire life cycle of a cloud-native application and to obtain an integrated and prioritized view of overall cloud risk.

Business Impact

Cloud-native application protection platforms consolidate disparate, fragmented security scanning and protection tools that increase cost and complexity for IT. Using a CNAPP offering will improve developer and security professional efficacy. It will also reduce complexity and provide improved cloud risk exposure visibility while maintaining development agility and improving the developer's experience.

Drivers

Cloud-native application protection platforms provide an integrated set of cloud exposure management capabilities that:

- Reduce the chance of misconfiguration, mistake or mismanagement as cloud-native applications are rapidly developed, released into production and iterated.
- Converge and reduce the number of tools and vendors involved in the continuous integration/continuous delivery (CI/CD) pipeline.

- Reduce the complexity and costs associated with creating secure and compliant cloud-native applications.
- Facilitate automated and continuous reporting and auditing of cloud security posture and status.
- Improve developer acceptance with security-scanning capabilities that seamlessly integrate into their development pipelines and tooling.
- Emphasize scanning proactively in development and rely less on runtime protection, which is well-suited for container as a service and serverless function environments.
- Address security posture management of emerging AI services offered by cloud providers.

Obstacles

- Cloud workload protection platform (CWPP) vendors that are good at runtime protection aren’t necessarily good at integrating into development and vice versa.
- Cloud-native workloads in the form of containers and serverless functions don’t require heavyweight runtime protection capabilities.
- No single CNAPP offering does everything. Convergence of capabilities will occur but will take place over several years.
- Organizations may have siloed purchases of application security testing tooling chosen by a different team that manages the runtime protection of workloads. At runtime, a separate team may be responsible for web application protection.
- Buying centers and influencers are shifting to newer roles such as DevOps architects, cloud security engineering and platform engineering, requiring information security teams to coordinate with these users.

User Recommendations

- Prioritize vendors that fulfill core CNAPP requirements by scanning containers in development and adding cloud security posture management (CSPM) capabilities, including snapshots and infrastructure-as-code scanning.
- Select integrated offerings with flexible licensing models that allow you to pay only for the capabilities your organization is prepared to use.
- Evaluate the CSPM vendor's ability to report on Kubernetes security posture management and provide runtime Kubernetes protection capabilities.
- Consolidate open-source software vulnerability scanning and software composition analysis through integrations or replacement within a CNAPP offering.
- Proactively scan containers in development for all types of vulnerabilities, not just vulnerable components, including hard-coded secrets, malware and Kubernetes misconfiguration.
- Look for explicitly integrated partnerships where CNAPP providers have gaps, for example, workload runtime agents, static analysis or software composition analysis.

Sample Vendors

Aqua Security; CrowdStrike; Fortinet (Lacework); Microsoft; Palo Alto Networks; Rapid7; SentinelOne; Sysdig; Upwind; Wiz

Gartner Recommended Reading

Market Guide for Cloud-Native Application Protection Platforms

How to Select DevSecOps Tools for Secure Software Delivery

5 Ways CNAPP Will Improve Your Cloud Security

How to Protect Your Cloud-Native Applications in Production

Climbing the Slope

Reachability Analysis

Analysis By: Aaron Lord

Benefit Rating: High

Market Penetration: 20% to 50% of target audience

Maturity: Early mainstream

Definition:

Reachability analysis in security testing is a method that focuses on identifying and prioritizing vulnerabilities based on whether they are reachable or exploitable within a given application's code. Used in conjunction with static application security testing (SAST) and software composition analysis (SCA), reachability analysis will contextualize vulnerability management by revealing what vulnerabilities realistically affect application workloads.

Why This Is Important

Organizations are overburdened with security vulnerabilities in their backlogs and require solutions to reduce their "mountains" into "molehills." Open-source and third-party components may contain a long list of vulnerabilities, but not all of them are directly accessible from your codebase. Reachability analysis helps in triaging the vulnerabilities based on their exploitability.

Business Impact

Besides the risk of security issues becoming security incidents, developers are reserving more hours for remediation, taking time away from new revenue-driving features. For cybersecurity personnel, this leads to more frustration and lower morale when dealing with the security posture of software.

Drivers

- Strong security vulnerability management now requires dynamic prioritization of issues using organizational context, as opposed to using third-party standards alone, like the Common Vulnerability Scoring System.
- Increased usage of open-source and third-party software with known vulnerabilities is contributing to security findings that further bloat security and development backlogs.
- A big portion of third-party code is onboarded but never actually invoked during runtime. Therefore, most of the vulnerabilities present in third-party components are not practically exploitable. By postponing the remediation of such vulnerabilities, developers can prioritize resolving the most immediate risks with minimizing impact on the current sprint.
- Reachability analysis has become a requirement for security tooling that scans for vulnerabilities, putting security tools that do not have reachability analysis out of the market.

Obstacles

- Reachability analysis can add an additional layer of tests to currently existing security tooling that needs management and configuration.
- When utilized at runtime, reachability analysis instruments the workload and not the application directly. This adds a minor amount of performance overhead to the execution.

- Reachability analysis can produce a false sense of security if it discovers that no exploitable vulnerabilities are remaining in a codebase, allowing nonexploitable vulnerabilities to accumulate if not prioritized by the organization.

User Recommendations

- Use reachability analysis in conjunction with static security scanning methods (i.e., SAST and SCA) to achieve more accurate findings from these methods.
- If an already-deployed static security scanning tool does not have reachability analysis, consult with the vendor providing the solution for their roadmap to include it.
- Use the reachability analysis context to improve software bill of materials with Vulnerability Exploitability eXchange data.
- Ensure that engineering or security teams managing SAST or SCA tooling are capable of taking on minor additional overhead of configuring reachability analysis tests.
- Organizations must still prioritize resolving nonexploitable vulnerabilities through strategic technical debt management to continue mitigating risk.

Sample Vendors

Backslash; Cisco (Deepfactor); Endor Labs; Insignary; Kodem; Lineaje; Mend.io; Oligo Security; Semgrep

Gartner Recommended Reading

Mitigate Enterprise Software Supply Chain Security Risks

Market Guide for Software Supply Chain Security

Cloud WAAP

Analysis By: Esraa ElTahawy, Dale Koeppen, Adam Hils, Aaron McQuaid

Benefit Rating: High

Market Penetration: 20% to 50% of target audience

Maturity: Mature mainstream

Definition:

Cloud web application and API protection (WAAP) is a security solution designed to protect web applications and APIs from different types of attacks. Typically delivered as a service, cloud WAAP consolidates multiple capabilities into a series of security modules. Core capabilities are web application firewalls (WAFs), application-level DDoS mitigation, API protection and automated traffic management. A cloud WAAP solution must incorporate all four core capabilities in the same offering.

Why This Is Important

Organizations that host web applications and APIs on-premises or in the cloud are increasingly buying cloud WAAP solutions because they can be easier to deploy and more flexible to deliver and manage, compared with traditional appliances. Additionally, constant increase in API traffic (driven by continuous integration/continuous delivery practices, increased agentic AI use and proliferation of microservices architectures) and development/use of applications powered by generative AI (GenAI) also increase the need for cloud WAAP.

Business Impact

Modern applications are mostly web-based, microservices-architected and accessible using APIs. Those applications and APIs are prime targets for cyberattacks, including sophisticated bot attacks, client-side attacks and zero-day vulnerabilities. A cloud WAAP solution addresses these challenges by integrating a mix of detective and preventive controls and consolidating them under a single as-a-service offering. Cloud WAAP solutions are easier for organizations to deploy than their appliance counterparts. They're also highly scalable and provide centralized management, centralized visibility and stronger integration with hyperscalers.

Drivers

- **Increase in API traffic:** Modern software development relies heavily on microservices architectures. A microservices-based design decomposes a single monolithic application into several loosely coupled, independent software instances (typically containerized). These instances communicate with each other via APIs and, therefore, must be secured.
- **Continuous increase in application layer attacks:** Especially with the introduction of GenAI-powered applications, robust application layer security is required to mitigate traditional and newer types of attacks.
- **Constant advancements in automated attacks:** Bot attacks are increasingly being leveraged to target multiple application components. This development is a critical concern for retail and banking sectors.
- **Cloud migrations:** Buyer strategy is shifting from traditional appliance models to cloud-delivered services. Adopting a cloud-delivered service that can cover both cloud and on-premises environments with a common management interface mitigates the burden of deploying and managing a discrete set of appliances at every physical location. Cloud WAAP solutions, based on a content delivery network (CDN), that offer caching services and allow for lower latency, in addition to offering WAAP security services, appear to be

more appealing to organizations with a distributed client base.

- **Support for integrations:** Many cloud WAAP vendors support integrations with different types of cloud-based, third-party security tools. Examples are application security testing and application security posture management tools, as well as cloud-native application protection platforms.

Obstacles

- **Privacy and compliance:** The fact that cloud WAAP solutions have access to decrypted traffic continues to pose privacy and compliance concerns to organizations, especially those located in heavily regulated industries or regions.
- **Disconnect:** Full visibility into application security posture and related risks is lacking, as cloud WAAP's main focus is on application runtime protection. In some instances, integration options offered by some cloud WAAP vendors dilute the impact of this disconnect.
- **Cost:** Cloud WAAP offerings can be more expensive compared with on-premises WAAP offerings, especially those offered on top of a CDN underlay. There's also the added cost of migration, especially for organizations that require a redesign of their security architecture.
- **Skills gap:** Managing a WAAP solution that comprises many application security capabilities demands highly skilled — and thus highly paid — application security experts. In addition, the expanding feature sets for cloud WAAP solutions lead many organizations to use a managed security service, which is an additional cost.

User Recommendations

- Align the choice of cloud web application protection platforms with their security program mandates, risk mitigation requirements and architectural initiatives.

- Establish the security requirements and priorities for each critical application's core and common capabilities and features. Set desired evaluation criteria and develop measurable metrics to compare vendor offerings.
- Prefer WAAP solutions that provide the ability to get inside an application, such as a container-based WAF that can leverage an extended Berkeley Packet Filter (eBPF) to provide kernel- and network-level visibility and control.
- Continue to improve your stance against bots and automated attacks by measuring the efficacy of existing controls and adding new techniques when needed.
- Choose cloud WAAP products that include automated API discovery and anomaly detection. Some cloud WAAP solutions do not yet offer best-of-breed API security capabilities; compare those with offerings from pure-play API security vendors and map the offered capabilities against your API security requirements.

Sample Vendors

Akamai; AWS; Cloudflare; F5; Fastly; Fortinet; Microsoft; Palo Alto Networks; Radware; Thales (Imperva)

Gartner Recommended Reading

Market Guide for Cloud Web Application and API Protection

Invest Implications: Market Guide for Cloud Web Application and API Protection

Application Shielding

Analysis By: Dionisio Zumerle

Benefit Rating: Moderate

Market Penetration: 5% to 20% of target audience

Maturity: Adolescent

Definition:

Application shielding prevents and detects attacks such as tampering and reverse engineering. It is an in-app protection technology, meaning its capabilities are implemented directly within the application, rather than in-line or on the hosting system. Application shielding can be used for any type of application, but there is currently a particular focus on mobile apps.

Why This Is Important

Mobile applications move software logic and place sensitive data on the user devices. These applications expose functionality that, unless shielded, can lead to attacks such as data exfiltration from the app or its back end, and fraud against the user or the application. Application shielding is an important security measure when applications convey or store sensitive data or enable payments..

Business Impact

Application shielding protects an enterprise’s software and applications when running on untrusted devices from cloning, information leakage, fraud, intellectual property (IP) theft and other forms of abuse. Besides helping to achieve regulatory compliance, it can be used to enhance UX in industries such as financial services and online retail by hardening the application, which enables the organization to minimize restrictions for its customers.

Drivers

- Innovation for application shielding revolves around two broad families of functionality to address current mobile application security threats: hardening and antitampering.

  - **Hardening** hinders the attacker from stealing information (such as IP or credentials) or cloning the application by making reverse-engineering harder. Hardening includes techniques such as code obfuscation and white-box cryptography.
  - **Antitampering** performs reconnaissance of the environment the application runs in to identify potential risks. Antitampering includes techniques such as emulation, rooting and debugging detection.
- Adoption is growing for consumer-facing mobile applications in industry verticals such as financial services, online retail, healthcare, insurance, gaming, entertainment and automotive. Application shielding is suitable for applications that run on untrusted environments.
- Current attacks often use functionality such as accessibility services or remote assistance to perform fraudulent actions against customer victims on mobile devices. Application shielding combats these attacks by enriching antitampering controls with anti-malware-specific techniques.
- Evolving native application protection capabilities, especially from Android's Play Integrity API, are making it possible for organizations to add antitampering application shielding functionality, albeit less specialized, without requiring them to acquire specialized products.

Obstacles

- To be effective, application shielding techniques must constantly evolve and keep up with the latest attacks, making application shielding a research-intensive discipline with products perceived as costly by mainstream organizations.
- Many technology-savvy security and development practitioners are not familiar with application shielding. Even if the risks they address may be clear, the techniques and efficacy to mitigate these risks are not apparent to end users. Most application shielding

measures function as deterrence measures; with enough time and persistence, obfuscation and other protections are possible to defeat.

User Recommendations

- Identify critical apps that store or access sensitive information (payment data or IP) and run in untrusted environments (for example, on customer devices).
- Use shielding measures natively offered by iOS and Android where possible. Complement them with specialized products in high-security scenarios, such as products offering obfuscation, device binding and anti-emulation.
- Prioritize the adoption of commercial application shielding if your organization is in the financial services, online retail, gaming, insurance or healthcare industry verticals.
- Prefer postcoding implementation when you do not have access to the source code or do not want to impact the development life cycle. Favor implementation during development when you require complex functionality such as white-box cryptography and have access to the source code.

Sample Vendors

Appdome; Build38; Digital.ai; DoveRunner; Guardsquare; Licel; PreEmptive; Promon; Verimatrix; Zimperium

Gartner Recommended Reading

How to Avoid Common Cybersecurity Pitfalls in Mobile App Development

Mobile Application Security Testing

Analysis By: Dionisio Zumerle

Benefit Rating: Moderate

Market Penetration: 20% to 50% of target audience

Maturity: Early mainstream

Definition:

Mobile application security testing (AST) identifies and helps remediate vulnerabilities within mobile applications for iOS and Android devices. Mobile AST analyzes source, byte or binary code, and observes or attacks mobile applications to identify coding, design, packaging, deployment and runtime conditions that introduce security vulnerabilities.

Why This Is Important

Mobile applications can be exploited by attackers to steal enterprise data and defraud their customers. Mobile AST largely uses similar techniques to traditional AST that are adapted for the mobile device environment, and the more agile development processes that come with it.

Business Impact

Depending on an organization's structure, mobile AST may be used by security and application development teams, and sometimes directly by the line-of-business departments. Even though every organization that delivers mobile applications should perform security testing, regulated and other high-security industries, such as financial services, healthcare and online retail, have a higher urgency to adopt mobile AST.

Drivers

- The techniques used in mobile AST build on well-known static AST (SAST), dynamic AST (DAST), software composition analysis (SCA) and interactive AST (IAST) techniques that

have been used for years to test web-based applications. However, when applied to mobile applications, testing must be adapted to support mobile-specific programming languages and frameworks, and identify, in addition to traditional application vulnerabilities, mobile-specific ones, such as unsecured storage and hard-coded credentials.

- The OWASP Mobile Application Security Verification Standard (MASVS) provides a granular set of requirements for mobile application security testing, helping both mobile AST vendors and practitioners better frame mobile AST needs.
- Conducting mobile AST can contribute to streamlining the approval and publication in commercial application stores easier. For example, through the App Defense Alliance Mobile Application Security Assessment, Google recognizes mobile applications that have undergone independent security testing against MASVS Level 1 requirements.
- Mobile AST is needed to scan open-source software (OSS) components and software development kits (SDKs), which are both frequently used with mobile applications. OSS components and SDKs are often vulnerable or require excessive permissions. With the emergence of software bills of materials, this need becomes even more pressing.
- While mobile AST products are mainly used with homegrown applications, some enterprises are using them for application vetting. This allows organizations to identify leaky or malicious applications. This use case is also often addressed by mobile threat defense product offerings.

Obstacles

- Many organizations have less-advanced application security programs and are not yet testing mobile application code. The ones that do often have high-security requirements, and their applications are heavily protected, making their testing difficult and time-consuming.

- Organizations typically see back ends as being the highest risk and, therefore, put less focus on mobile AST.

User Recommendations

- Perform mobile AST, whether your mobile applications are workforce- or consumer-facing, starting with the most critical ones. These include applications that run in untrusted environments, with software logic on the client side, and applications that have transactional or intellectual-property value.
- Investigate whether the mobile AST capabilities your incumbent AST vendor provides, directly or via a partnership, can suffice. Typical AST selection considerations still apply, but for mobile specifically, the OWASP MASVS can be a practical reference guide to assess a solution's technical capabilities and coverage.
- Look into dedicated mobile AST vendor offerings if you do not have an incumbent AST solution, or need a specialized or lightweight and speedier mobile AST product.

Sample Vendors

Appknox; Data Theorem; Guardsquare; ImmuniWeb; Mobisec; NowSecure; Ostorlab; Quixxi; Quokka; Zimperium

Gartner Recommended Reading

Critical Capabilities for Application Security Testing

Secure Coding Training

Analysis By: Aaron Lord, Manjunath Bhat

Benefit Rating: High

Market Penetration: 20% to 50% of target audience

Maturity: Early mainstream

Definition:

Secure coding training raises developer awareness of the impact — and enables the prevention of — vulnerabilities in source code. Developers get secure coding training for specific languages and frameworks using different methods like just-in-time training, gamified lessons, workshops and challenges.

Why This Is Important

According to the 2025 Gartner Software Engineering Survey, 64% of software engineering leaders stated that application security was highly important to deliver software that meets business needs. Secure coding skills are often missing from general software engineering education, yet they are essential as secure code evolves over time. Ensuring that software engineers are up-to-date with the latest secure coding skills will reduce the risk of introducing vulnerabilities.

Business Impact

Any organization that develops software needs to be concerned about security, and out-of-date security training is just as bad as no training. Security issues in software will lead to undesirable outcomes, such as longer lead times to deployment of new features, increase in the risk exposure of vulnerabilities that are left unaddressed and lower morale for software engineers.

Drivers

- Application security is consistently a major concern for software organizations.

- According to the 2023 Gartner Security in Software Engineering Survey (n = 300), software engineers who are more directly involved with security activities see improvements to security outcomes.
- Developers lacking security skills and knowledge produce unsecured applications by introducing vulnerable code, third-party components and infrastructure configurations.
- Staying up-to-date with secure coding practices will help reduce the risk of software vulnerability exposures by shortening the time to remediate issues.
- Organizations leverage generative AI to enhance the creation of training content, but it also drives the need for more training for using these AI coding assistants.

Obstacles

- Just like any approach to teaching, different types of secure coding training (like presentation, tests and workshops) work better or worse for certain individuals.
- According to the 2023 Gartner Security in Software Engineering Survey, software engineers do not have enough time for learning, nor do they get incentives for its completion.
- When training is assigned annually, dropping knowledge retention between training sessions may lead to more vulnerable applications.
- Offerings and pricing models for secure coding training can vary widely, making it difficult to understand what is needed.

User Recommendations

- Collaborate with security leadership to form strategies that build security skills within software engineering teams.

- Choose a secure coding training vendor that offers training that matches the organization's technology stacks, languages and frameworks.
- Create dedicated time for software engineers for training activities outside of their day-to-day work, and measure the success to get buy-in from the leadership.
- Use a training platform that offers workshops for improved engagement and emphasizes peer-to-peer knowledge sharing.
- Ensure that secure coding training offerings come in multiple languages that can support engineers across multiple regions.
- Implement learning for software engineers organically by integrating microtraining into the software development life cycle.
- Utilize secure coding training platforms as a tool to foster and recruit security champions from software engineering teams.

Sample Vendors

Avatao; Black Duck; Checkmarx (Codebashing); Immersive Labs; Secure Code Warrior; SecureFlag; Security Compass; Security Journey; Snyk; Veracode

Gartner Recommended Reading

Develop Software Engineering Skills Using Online Learning Platforms

DevSecOps Maturity Model for Secure Software Development

2025 Technology Adoption Roadmap for Software Engineering

Entering the Plateau

Bot Management

Analysis By: Akif Khan

Benefit Rating: High

Market Penetration: More than 50% of target audience

Maturity: Mature mainstream

Definition:

Bot management solutions can detect and respond to automated applications and bots that interact with websites, chatbots, mobile apps and APIs. They can block unwanted automated activity while ensuring that human users and legitimate bots have access as intended by the business. Assessing the humanity of interactions is typically achieved through a layered approach of examining network signals, devices and session attributes, and potentially forcing the user to solve a CAPTCHA.

Why This Is Important

Bot traffic continues to rise along with increasing bot sophistication, including the recent introduction of AI agents for consumer use. Attackers use bots to automate business logic abuse attacks on online assets, including scraping of competitive data, inventory grabbing and credential stuffing, and full or partial denial of service — intentional or not. Attackers' approach of combining specialized bots with human-operated fraud-farm requires ever-improving detection and response solutions.

Business Impact

Although used initially to protect large B2C web applications, bot management has become increasingly relevant to any B2C or B2B public-facing endpoint where automated traffic can

directly impact business outcomes. Malicious bots can negatively impact user experience (UX) by causing slower page load times, hoarding inventory and facilitating account takeover via credential stuffing or credential cracking. The use of AI agents by both good and bad actors adds further complexity.

Drivers

- There is a need to prevent large-scale, low-sophistication automated attacks from basic bots or scripts created using generative AI (GenAI) applications.
- Leaders' increasing recognition that bot management crosses multiple use cases and business units makes these capabilities more sought after.
- Legacy CAPTCHA solutions can be solved by bots or hu-bots that delegate to CAPTCHA solver services. Also, poorly conceived CAPTCHA solutions can degrade UX. This has driven innovation in bot management solutions that have minimal impact on UX, such as device intelligence, behavioral biometrics and invisible challenges such as cryptographic proof-of-work.
- Bot management solutions can identify malicious bots and preserve the UX of legitimate application users and authorized bots. Enterprise-approved bots can include search engine crawlers, automated testing and web application monitoring software, robotic process automation (RPA) or other machine-to-machine (M2M) communication. M2M communication includes bots that run in environments such as Microsoft Teams, Slack, and some bot marketplaces.
- Bot mitigation is being commonly bundled into content delivery network (CDN) solutions, thus lowering the barriers for organizations to explore the benefits of bot management.
- There is renewed focus on bot management solutions as a means to identify and quantify the use of emerging AI agents such as OpenAI's Operator or Anthropic's Claude on consumer services; for instance, as e-commerce or banking. Organizations have yet to

develop strategies on how to engage with AI agent use, given that they may be used by both good and bad actors. The first step is to identify when they are being used.

Obstacles

- The fear of blocking a single legitimate user is often higher than the perception of the damage being caused by malicious bots.
- Concerns persist that bot management vendors use CAPTCHA too frequently, often to test for false positives. This is usually <1% of traffic, and CAPTCHA solutions are proprietary and relatively optimized. However, organizations still fear the friction inherent in CAPTCHA challenges.
- The nature of the threat posed by bots will change rapidly in the next several years as AI agents become commonplace. While such agents can be used by good actors for legitimate time saving and automation tasks, they will also be used by attackers. Traditionally used malicious bots based on scripts and headless browsers are likely to be replaced by AI agents as the attackers' tools of choice. Bot management vendors will need to adapt and demonstrate that they can help organizations not only identify when AI agents are being used, but also whether they have malicious intent.

User Recommendations

- Assess the threat level to business assets caused by bots, starting with the most business-critical and most exposed web applications and APIs.
- Evaluate the capabilities of bot management solutions that come with your web application and API protection (WAAP) or CDN platform. If these are sufficient, they represent a quick win in terms of implementation and tool rationalization. Evaluate specialized solutions if these tools do not meet your needs.
- Gauge how dependent a solution is on CAPTCHA challenges to assess false positives and

what that CAPTCHA experience looks like. Ensure that internal stakeholders focused on UX are comfortable with the approach. Reject any approach that relies on mandatory CAPTCHA solving.

- Engage with vendors who have forward-looking roadmaps with respect to the emerging threat and opportunity posed by AI agents. Develop your own strategy regarding AI agents engaging with your services given that not all AI agent use will be malicious.

Sample Vendors

Akamai Technologies; Arkose Labs; Cequence Security; Cloudflare; DataDome; F5; hCaptcha; HUMAN Security; Netacea; Radware

Gartner Recommended Reading

Innovation Insight: Bot Management for the IAM Leader

Market Guide for Cloud Web Application and API Protection

Appendixes

See the previous Hype Cycle: **Hype Cycle for Application Security, 2024**

Hype Cycle Phases, Benefit Ratings and Maturity Levels

Hype Cycle Phases

Phase	Definition
Innovation Trigger	A breakthrough, public demonstration, product launch or other event generates significant media and industry interest.
Peak of Inflated Expectations	During this phase of overenthusiasm and unrealistic projections, a flurry of well-publicized activity by technology leaders results in some successes, but more failures, as the innovation is pushed to its limits. The only enterprises making money are conference organizers and content publishers.
Trough of Disillusionment	Because the innovation does not live up to its overinflated expectations, it rapidly becomes unfashionable. Media interest wanes, except for a few cautionary tales.
Slope of Enlightenment	Focused experimentation and solid hard work by an increasingly diverse range of organizations lead to a true understanding of the innovation's applicability, risks and benefits. Commercial off-the-shelf methodologies and tools ease the development process

Source: Gartner (July 2025)

Benefit Ratings

Benefit Rating	Definition
Transformational	Enables new ways of doing business across industries that will result in major shifts in industry dynamics
High	Enables new ways of performing horizontal or vertical processes that will result in significantly increased revenue or cost savings for an enterprise
Moderate	Provides incremental improvements to established processes that will result in increased revenue or cost savings for an enterprise
Low	Slightly improves processes (for example, improved user experience) that will be difficult to translate into increased revenue or cost savings

Source: Gartner (July 2025)

Maturity Levels

Maturity Levels	Status	Products/Vendors
Embryonic	In labs	None
Emerging	Commercialization by vendors Pilots and deployments by industry leaders	First generation High price Much customization
Adolescent	Maturing technology capabilities and process understanding Uptake beyond early adopters	Second generation Less customization
Early mainstream	Proven technology Vendors, technology and adoption rapidly evolving	Third generation More out-of-box methodologies
Mature mainstream	Robust technology Not much evolution in vendors or	Several dominant vendors

Source: Gartner (July 2025)

⊕ Evidence