AWS Bedrock Attack Surface: Eight Validated Vectors

IAM Abuse, Sandbox Escape, and AI Platform Exploitation in Production Deployments

Unofficial AI-assisted Research

Cloud Security Alliance AI Safety Initiative

2026-03-24

Key Takeaways

- Security researchers at XM Cyber disclosed eight validated IAM-mediated attack paths against Amazon Bedrock in March 2026, spanning Agent manipulation, Knowledge Base exfiltration, Flow hijacking, and Guardrail disabling [1].
- A sandbox breakout in Amazon Bedrock AgentCore's Code Interpreter, discovered in September 2025, allows data exfiltration over DNS even when "no network access" is configured; AWS formally declined to patch it and updated the documentation instead [2].
- LLMjacking attacks against Bedrock—using stolen AWS credentials to invoke foundation models at victim expense—have reached documented costs of up to $46,000 per day for premium model invocations [7].
- Model invocation logging is disabled by default in all Bedrock deployments, creating forensic blind spots that attackers actively exploit before executing model-related operations [4].
- CVE-2026-4269 (CVSS 5.8), patched in March 2026, demonstrated that the AgentCore Starter Toolkit was vulnerable to malicious S3 build artifacts enabling remote code injection into the AgentCore Runtime [5].

Background

Amazon Bedrock is a fully managed AI platform that provides access to foundation models from Anthropic, Meta, Amazon, and third-party providers through a unified API. Launched into general availability in 2023, Bedrock has grown to include agentic orchestration capabilities (Bedrock Agents), vector-backed retrieval pipelines (Knowledge Bases), model-access guardrails, workflow automation (Flows), centralized prompt management (Managed Prompts), and a code execution sandbox (AgentCore Code Interpreter). Bedrock has seen significant enterprise adoption since its 2023 general availability, and the volume of security research disclosed in 2025–2026 reflects growing industry attention to its attack surface.

As organizations have moved from experimental AI deployments to production workloads, the security surface has expanded substantially. Bedrock Agents can invoke external APIs, read from S3 buckets, query vector databases, and execute arbitrary Python through Lambda action groups—all under IAM identities that may carry permissions inherited from broader cloud roles. This combination of elevated

agency, cloud-native integration, and complex permission relationships has attracted serious security research attention in the past twelve months. The findings that emerged between mid-2025 and early 2026 collectively reveal that Bedrock's attack surface is not a hypothetical concern but an active exploitation target.

The research documented in this note draws primarily on three independent bodies of work: a March 2026 disclosure by XM Cyber detailing eight IAM-mediated attack paths [1]; a September 2025 sandbox-escape finding by BeyondTrust Phantom Labs [2]; and a November 2025 real-world LLMjacking incident analyzed by Sysdig Threat Research Team [3]. Together they map a coherent landscape of threats that practitioners must understand to defend production Bedrock deployments.

Security Analysis

Eight IAM-Mediated Attack Vectors

In March 2026, XM Cyber researcher Eli Shparaga published an analysis identifying eight distinct attack paths against Amazon Bedrock infrastructure, each reachable by an identity holding overprivileged IAM permissions [1]. The vectors are notable not because they represent novel exploitation techniques, but because they demonstrate how routine IAM misconfiguration—a persistent problem across cloud environments—gains substantially increased impact when combined with Bedrock's broad integrations and autonomous capabilities. Independent analysis by Software Secured confirms that several of these IAM escalation paths overlap with broader AWS privilege escalation patterns that have been documented across the platform [12].

The first vector targets **model invocation logging**. An attacker with `bedrock:PutModelInvocationLoggingConfiguration` and `s3:DeleteObject` permissions can redirect all prompt and response logs to an attacker-controlled S3 bucket while simultaneously destroying existing forensic evidence. Because invocation logging is disabled by default, this attack is doubly effective: in organizations that have enabled logging, it can covertly neutralize that control; in organizations that have not, the attack simply reinforces an already-blind forensic state.

The second vector exploits **Knowledge Base data sources**. Bedrock Knowledge Bases can ingest content from SharePoint, Salesforce, Confluence, and S3 through native connectors. An attacker with access to the stored connector credentials—accessible via `bedrock:GetKnowledgeBase` in some configurations—can bypass the AI model entirely and retrieve raw proprietary documents from

connected SaaS platforms. This also creates a lateral-movement path: the same credentials that authenticate Bedrock to SharePoint authenticate an adversary to the same SharePoint environment, potentially providing access to Active Directory, Exchange, and other enterprise systems.

Knowledge Base data stores represent the third vector. Bedrock can persist embeddings in external vector databases including Pinecone, Redis Enterprise Cloud, Amazon Aurora, and Amazon Redshift. Credentials stored for these integrations can be extracted to gain full administrative access to the vector store, enabling data theft, embedding manipulation, or service disruption independent of any Bedrock API interaction.

The fourth and fifth vectors target **Bedrock Agents** through direct and indirect means, respectively. In the direct variant, an identity holding `bedrock:UpdateAgent` or `bedrock:CreateAgentActionGroup` can rewrite the agent's base system prompt or attach a malicious action group. Because agents execute with their own IAM role, a rewritten prompt can instruct the agent to exfiltrate data through its legitimate API-calling capabilities without the end user's knowledge. The indirect variant is subtler: an attacker with `lambda:UpdateFunctionCode` replaces the Lambda function that implements an agent's action group with malicious code conforming to Bedrock's expected response schema. The agent continues to function normally from the perspective of downstream consumers while executing attacker-controlled logic on every invocation. A variant using `lambda:PublishLayer` is harder to detect because the function's code hash remains unchanged while a malicious dependency is loaded at runtime.

Flow hijacking constitutes the sixth vector. Bedrock Flows enable visual orchestration of multi-step AI pipelines that can include S3 reads, Lambda invocations, and KMS operations. An attacker with `bedrock:UpdateFlow` can inject additional nodes into production pipelines, routing sensitive inputs and outputs to attacker-controlled endpoints or swapping encryption keys. Because flows execute asynchronously and may process high volumes of data, such modifications can persist undetected for extended periods.

The seventh vector targets **Guardrails**—Bedrock's content filtering and safety layer. An identity holding `bedrock:UpdateGuardrail` or `bedrock:DeleteGuardrail` can weaken topic blocks, disable sensitive information filters, or remove prompt-attack detection entirely. The practical consequence is that every subsequent model invocation through the affected endpoint operates without safety controls, dramatically increasing susceptibility to both direct and indirect prompt injection. Trend Micro documented an additional sub-variant in which an attacker injects malicious URLs into Guardrail block messages through `bedrock:UpdateGuardrail`, turning the safety infrastructure into a social engineering delivery mechanism [6].

The eighth vector attacks **Managed Prompts**—centralized prompt templates that may be shared across multiple agents and applications. An identity with `bedrock:UpdatePrompt` can modify these templates to embed persistent instruction injections, triggering data exfiltration or harmful content generation at scale across every application that references the shared template.

AgentCore Sandbox Escape via DNS Tunneling

In September 2025, BeyondTrust Phantom Labs disclosed a sandbox breakout in Amazon Bedrock AgentCore's Code Interpreter, a feature launched in August 2025 that allows agents to execute Python code in an isolated environment [2]. The finding was significant because AWS's documentation characterized the sandbox as having "complete isolation with no external network access." In practice, outbound DNS queries for A and AAAA records remained permitted even in no-network-access mode.

The attack chain proceeds through five stages. An attacker with the ability to influence the code executed inside the sandbox can encode exfiltration payloads into DNS subdomain labels, establishing a bidirectional command-and-control channel over DNS. From there, the execution context provides access to the agent's IAM role credentials, enabling lateral movement to S3, DynamoDB, Secrets Manager, and other AWS services accessible under that role. Researchers rated the vulnerability at CVSS 7.5 (High) [2].

AWS deployed an initial fix in November 2025 but rolled it back within two weeks. By December 2025, AWS formally declined to patch the issue and updated its documentation to reflect "limited external network access" rather than "complete isolation." Customers seeking true network isolation must deploy AgentCore Code Interpreter within a Virtual Private Cloud (VPC) configuration—an option that adds operational complexity and is not the default [2]. This episode suggests a broader tension in managed AI platforms: isolation guarantees in vendor documentation may not always reflect underlying technical behavior, particularly for newly launched sandbox features.

LLMjacking: Credential Theft for Unauthorized Model Invocation

LLMjacking is a financially motivated attack pattern in which adversaries steal AWS credentials and use them to invoke foundation models at the victim's expense. The attack has been documented in Bedrock-specific variants since at least mid-2025 [3]. A detailed incident from November 28, 2025, analyzed by Sysdig Threat Research Team, traced an attacker who obtained credentials from a publicly accessible source, validated their permissions using the IAM Policy Simulator, then systematically enumerated available models via `ListFoundationModels` and `ListAgents`. Before invoking any models, the attacker called `DeleteModelInvocationLoggingConfiguration` to suppress forensic evidence—a step that required only seconds and was undetected in standard CloudWatch alerting [3].

In that incident, the attacker invoked nine foundation models across thirteen calls, including Claude Opus 4, DeepSeek R1, Llama 4 Scout, Amazon Nova Premier, and Amazon Titan Image Generator [3]. Cross-region inference was used to distribute requests across geographic regions, fragmenting the invocation footprint across CloudTrail logs. Mitigant's independent analysis of LLMjacking attacks against Bedrock documented financial exposure of up to $46,000 per day for premium model tiers [7]; extrapolating from per-model pricing schedules, sustained premium-tier abuse could exceed $100,000 per day. Mitigant's analysis of Bedrock-specific attacks additionally found that many commercial XDR and CDR products generated no alerts for AI-platform credential abuse, attributing the gap to absent detection logic for these attack patterns [8].

Supply Chain Vulnerability: CVE-2026-4269

CVE-2026-4269, disclosed by AWS on March 16, 2026, affected all versions of the `bedrock-agentcore-starter-toolkit` Python library prior to v0.1.13 built after September 24, 2025 [5]. The vulnerability (CWE-283: Unverified Ownership) allowed an attacker to inject malicious build artifacts via S3 without proper ownership verification, leading to remote code execution within the AgentCore Runtime with complete loss of confidentiality, integrity, and availability. The CVSS v4.0 score of 5.8 (Moderate) reflects the elevated attack complexity required for exploitation; if those conditions are met, the impact is scored as complete loss of confidentiality, integrity, and availability within the affected component [5]. Remediation requires upgrading to v0.1.13 or later.

This CVE exemplifies an emerging class of AI-platform supply chain risks. As organizations adopt managed AI SDKs and starter toolkits, the dependency graph of AI platform components becomes a vector for supply chain compromise analogous to the risks documented in traditional open-source software supply chains.

RAG Data Source Poisoning and Direct Exfiltration

Bedrock Knowledge Bases built on S3-backed Retrieval-Augmented Generation (RAG) pipelines are vulnerable to three distinct attack patterns that operate entirely at the storage layer, without any Bedrock API interaction. First, an adversary with S3 write access can inject poisoned documents into the embedding corpus, influencing model responses for every user who triggers retrieval of the affected content—an instance of MITRE ATLAS technique AML.T0020 (Training Data Poisoning) applied to inference-time retrieval [8]. Second, an attacker with `s3:GetObject` permissions on a Knowledge Base data source bucket can retrieve raw proprietary data—contracts, source code, customer records—

bypassing the AI model and its guardrails entirely. Third, deletion or encryption of S3 objects backing a Knowledge Base destroys the vector embeddings and disrupts AI service availability, constituting a ransomware-style attack against AI infrastructure.

Mitigant's research noted that the S3 data source attack paths represent the primary threat vector for organizations that have secured their Bedrock API surface but neglected the underlying storage layer [8]. The research also found that detection tooling had not yet implemented signatures for these AI-specific storage attacks at the time of publication, leaving them as effective detection gaps in many environments.

Indirect Prompt Injection via Retrieved Content

Indirect prompt injection occurs when adversarial instructions are embedded in external content that a Bedrock Agent retrieves at runtime—web pages, documents, emails, database records, or API responses. Because agents treat retrieved context as implicitly trusted, a single poisoned document can hijack the agent's behavior for every subsequent user who triggers retrieval of that content. In agentic workflows where agents take real-world actions—sending emails, creating calendar entries, modifying database records—an indirect injection can produce harm that propagates far beyond the AI system itself [9].

AWS has published mitigation guidance centered on enabling Bedrock Guardrails' `PROMPT_ATTACK` filter and explicitly tagging dynamically retrieved content as user input for evaluation [9]. However, the `PROMPT_ATTACK` filter is available only in the Standard pricing tier and requires a non-trivial integration step that many deployments omit. Additionally, academic research published in April 2025 demonstrated that character-level injection techniques—Unicode substitutions, homoglyphs, and zero-width characters—can evade token-pattern-based detection in guardrail systems, achieving evasion rates up to 100% against tested guardrail systems in some cases [10]. These findings indicate that Guardrails should be treated as a defense-in-depth layer rather than a primary protection against sophisticated injection attacks.

Monitoring and Logging Blind Spots

The forensic posture of many Bedrock deployments is weaker than it may appear. Because model invocation logging—the only mechanism that records the content of prompts and model responses—is disabled by default [4], organizations that have not explicitly enabled it lack visibility into prompt and response content. Even when administrators enable invocation logging, CloudTrail's coverage of Bedrock data-plane operations is incomplete: streaming inference (`InvokeModelWithBidirectionalStream`), asynchronous invocation (`StartAsyncInvoke`, `GetAsyncInvoke`), and other runtime operations are classified as

CloudTrail data events, which are not logged by default and require explicit activation [4]. Throttled requests are excluded from both invocation counts and error metrics in CloudWatch, meaning rate-limiting-based abuse is invisible to error-rate alerting.

Attackers deliberately exploit this default gap. In the November 2025 incident analyzed by Sysdig, the attacker verified that logging was disabled and deleted it if present before invoking any models [3]. Organizations should not assume default configurations reflect secure-by-default posture. Splunk Security Content has published a detection rule for `DeleteModelInvocationLoggingConfiguration` API calls, which serves as a high-fidelity indicator of attacker cleanup activity [11], but this rule only fires if CloudTrail management events are being collected and forwarded to a SIEM.

Recommendations

Immediate Actions

Organizations running production Bedrock workloads should treat the following as urgent. All accounts should enable Bedrock model invocation logging immediately, directed to a dedicated S3 bucket with object-lock enabled to prevent post-compromise deletion. CloudTrail data events for Bedrock should be enabled in all regions where cross-region inference is permitted. Alert rules should be deployed for `DeleteModelInvocationLoggingConfiguration`, `UpdateGuardrail`, `DeleteGuardrail`, `UpdateAgent`, and `UpdatePrompt` API calls. The `bedrock-agentcore-starter-toolkit` package should be audited and upgraded to v0.1.13 or later, and any AgentCore Code Interpreter deployments that rely on the sandbox's documented isolation guarantees should be evaluated against the actual DNS-escape behavior and migrated to VPC-isolated configurations if true network isolation is required.

Short-Term Mitigations

IAM posture for Bedrock-related identities should be reviewed and tightened using least-privilege principles. The permissions enabling the eight IAM-mediated attack vectors—`bedrock:UpdateAgent`, `bedrock:UpdateGuardrail`, `bedrock:UpdateFlow`, `bedrock:UpdatePrompt`, `bedrock:PutModelInvocationLoggingConfiguration`, `lambda:UpdateFunctionCode`, `lambda:PublishLayer`, and `s3:DeleteObject` on invocation-log buckets—should be audited and removed from roles that do not require them for

legitimate operations. S3 buckets backing Bedrock Knowledge Bases should be reviewed for public-access settings and over-broad write permissions. SCP policies should be considered to prevent any role from disabling model invocation logging across the organization.

For agentic deployments, the principle of minimal agent scope should be applied rigorously: agents should operate with the minimum IAM permissions necessary to perform their designated functions, and action groups should be scoped to the narrowest possible Lambda function permissions. Bedrock Guardrails' `PROMPT_ATTACK` filter should be enabled on all production endpoints and configured to evaluate dynamically retrieved content as well as user input. AWS prescriptive guidance for OWASP LLM Top 10 mitigations in Bedrock deployments provides additional implementation detail for these controls [13].

Strategic Considerations

The depth of Bedrock's cloud-native integration—spanning S3, Lambda, vector databases, SaaS connectors, and KMS—means that securing Bedrock requires a holistic cloud security posture review, not just AI-specific controls. Organizations should treat Bedrock IAM roles with the same rigor applied to privileged cloud infrastructure roles. Continuous automated permission analysis tools should be configured to detect and alert on any grant of the high-risk Bedrock permissions documented above.

AWS Billing Alerts should be configured for anomalous model invocation costs, as LLMjacking attacks may be detectable as billing anomalies before they appear in security logs. Cross-region inference usage should be baselined and monitored, since adversaries use it specifically to fragment their CloudTrail footprint across regions.

Dependency scanning should be integrated into CI/CD pipelines for any code that imports Bedrock-related libraries, particularly `bedrock-agentcore-starter-toolkit` and associated SDKs. SBOM (Software Bill of Materials) generation should be mandated for AI platform components.

CSA Resource Alignment

This research note connects directly to several active CSA frameworks and programs.

The **MAESTRO Agentic AI Threat Modeling Framework** provides the conceptual architecture for categorizing the eight IAM-mediated attack vectors as threats against the Orchestration Layer (Flow and Agent manipulation), the Tool/Action Layer (Lambda code injection), the Knowledge Layer (Knowledge Base poisoning and data source attacks), and the Safety Layer (Guardrail disabling).

MAESTRO's emphasis on trust boundary analysis between agent components reflects the same insight that drives the XM Cyber findings: that IAM permissions within a Bedrock deployment constitute implicit trust relationships, and that over-permissioned identities flatten these boundaries in dangerous ways.

The **CSA AI Controls Matrix (AICM)**, a superset of the Cloud Controls Matrix, provides a compliance mapping surface for the controls that would mitigate these vectors. Relevant control domains include AI-specific access control, audit logging, incident response, and supply chain security. Organizations seeking to demonstrate compliance posture against the attack surface documented here should map their Bedrock hardening controls to AICM domains and submit self-assessments through the STAR for AI Level 1 program.

The **CSA Agentic AI Red Teaming Guide** (2025), authored by the AI Organizational Responsibilities Working Group, provides testing methodologies directly applicable to the Bedrock agentic attack surface [14]. Its coverage of Agent Authorization and Control Hijacking, Knowledge Base Poisoning, Supply Chain and Dependency Attacks, and Agent Untraceability maps closely onto the attack vectors documented here. Organizations should incorporate these testing procedures into pre-production security assessments of Bedrock Agents and Flows.

The **CSA Zero Trust guidance** underlies the strategic recommendation to never treat Bedrock's internal communication paths as inherently trusted. The fact that agents, flows, Lambda action groups, and vector databases exchange data through IAM-authenticated channels does not confer trust; each integration point requires explicit authorization scoping, continuous verification, and audit trail maintenance consistent with Zero Trust principles.

References

[1] Eli Shparaga, XM Cyber, "We Found Eight Attack Vectors Inside AWS Bedrock," The Hacker News, March 2026. https://thehackernews.com/2026/03/we-found-eight-attack-vectors-inside.html

[2] BeyondTrust Phantom Labs / Aviatrix Threat Research, "Amazon Bedrock AgentCore Code Interpreter DNS Exfiltration Vulnerability," CSO Online / Aviatrix, September 2025 – March 2026. https://www.csoonline.com/article/4146202/aws-bedrocks-isolated-sandbox-comes-with-a-dns-escape-hatch.html; https://aviatrix.ai/threat-research-center/amazon-bedrock-agentcore-2026-dns-exfiltration-vulnerability/

[3] Sysdig Threat Research Team, "AI-Assisted Cloud Intrusion Achieves Admin Access in 8 Minutes," Sysdig Blog, February 2026. https://www.sysdig.com/blog/ai-assisted-cloud-intrusion-achieves-admin-access-in-8-minutes

[4] Amazon Web Services, "Model Invocation Logging," AWS Bedrock Documentation, 2026. https://docs.aws.amazon.com/bedrock/latest/userguide/model-invocation-logging.html; https://docs.aws.amazon.com/bedrock/latest/userguide/logging-using-cloudtrail.html

[5] Amazon Web Services, "AWS Security Bulletin 2026-008-AWS: CVE-2026-4269 – Improper S3 Ownership Verification in bedrock-agentcore-starter-toolkit," AWS Security, March 16, 2026. https://aws.amazon.com/security/security-bulletins/rss/2026-008-aws/; https://github.com/aws/bedrock-agentcore-starter-toolkit/security/advisories/GHSA-xfhr-q72q-jcrj

[6] Trend Micro, "Detecting Attacks on AWS AI Services with Trend Vision One," Trend Micro Research, 2025. https://www.trendmicro.com/vinfo/us/security/news/virtualization-and-cloud/detecting-attacks-on-aws-ai-services-with-trend-vision-one

[7] Mitigant, "Demystifying Amazon Bedrock LLMjacking Attacks," Mitigant Blog, 2024. https://mitigant.io/en/blog/demystifying-amazon-bedrock-llmjacking-attacks

[8] Mitigant, "Bedrock or Bedsand: Attacking Amazon Bedrock's Achilles Heel," Mitigant Blog, 2025. https://mitigant.io/en/blog/bedrock-or-bedsand-attacking-amazon-bedrocks-achilles-heel

[9] Amazon Web Services, "Securing Amazon Bedrock Agents: A Guide to Safeguarding Against Indirect Prompt Injections," AWS Machine Learning Blog, 2025. https://aws.amazon.com/blogs/machine-learning/securing-amazon-bedrock-agents-a-guide-to-safeguarding-against-indirect-prompt-injections/

[10] William Hackett, Lewis Birch, Stefan Trawicki, Neeraj Suri, Peter Garraghan, "Bypassing LLM Guardrails: Evasion Attacks via Character-Level Injection," arXiv:2504.11168, April 2025. https://arxiv.org/abs/2504.11168

[11] Splunk Security Research, "AWS Bedrock Delete Model Invocation Logging Configuration," Splunk Security Content, 2024. https://research.splunk.com/cloud/9c5e3d62-f743-11ee-9f6e-acde48001124/

[12] Software Secured, "AWS Privilege Escalation: IAM Risks, Service-Based Attacks and New AI-Driven Bedrock AgentCore Vectors," Software Secured Blog, 2026. https://www.softwaresecured.com/post/aws-privilege-escalation-iam-risks-service-based-attacks-and-new-ai-driven-bedrock-agentcore-vectors

[13] Amazon Web Services, "Prescriptive Guidance: OWASP LLM Top 10 for Amazon Bedrock," AWS Prescriptive Guidance, 2025. https://docs.aws.amazon.com/prescriptive-guidance/latest/agentic-ai-security/owasp-top-ten.html

[14] Cloud Security Alliance, "Agentic AI Red Teaming Guide," CSA AI Organizational Responsibilities Working Group, 2025. https://cloudsecurityalliance.org/research/working-groups/ai-organizational-responsibilities