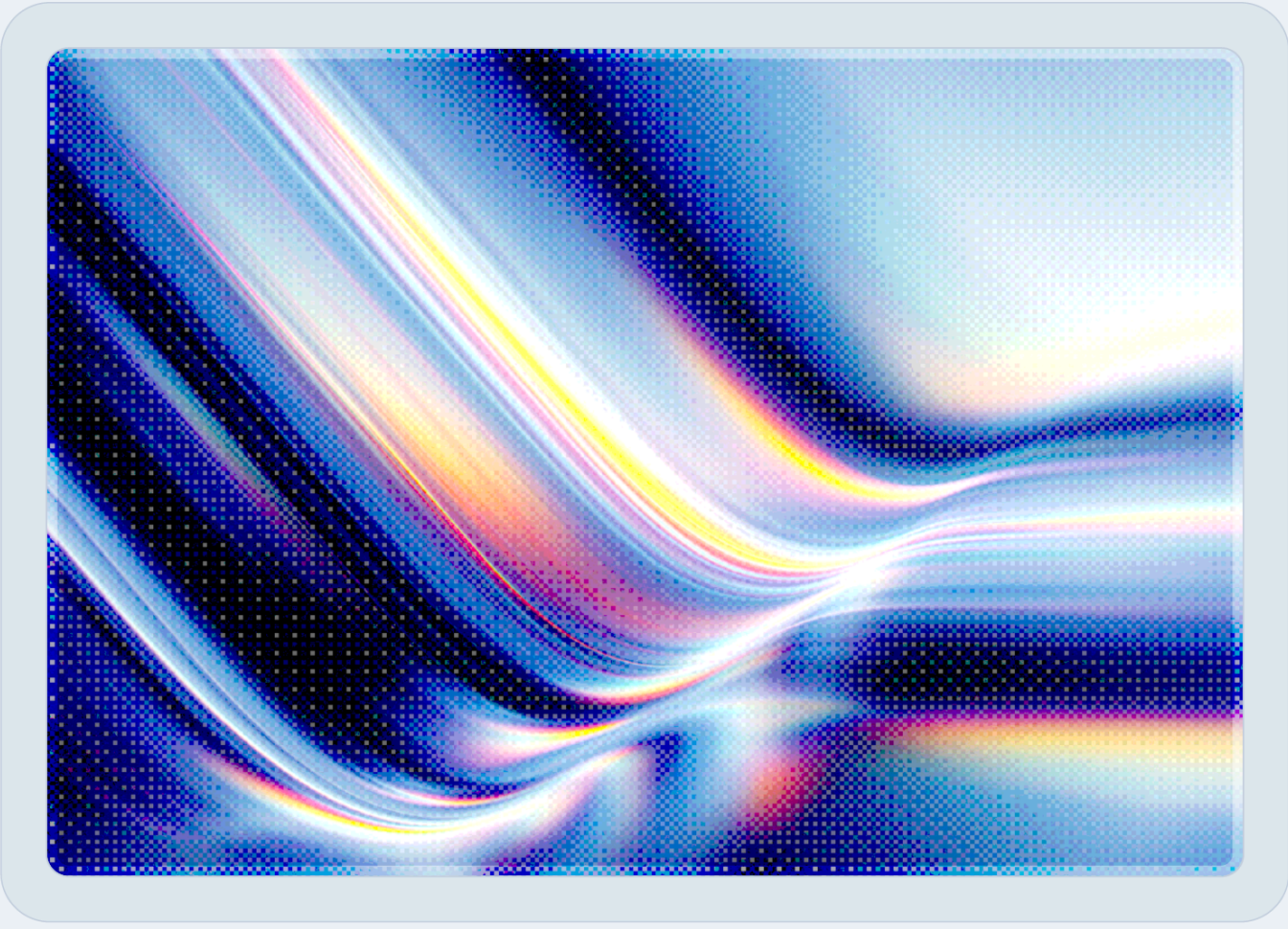
CSAI Foundation | Cloud Security Alliance

Canary Evidence of Prompt Theft at AI Inference Providers

Months Inside the Inference Layer and the Zero Data Retention Verification Gap

2026-11-23

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

© 2026 Cloud Security Alliance. Some rights reserved.

You may download, store, display, view, print, redistribute, and link to this document in its original, unmodified form, provided that attribution to the Cloud Security Alliance is maintained and all trademark and copyright notices remain intact.

This document may not be modified or altered. You may quote portions of the document as permitted by the Fair Use provisions of the United States Copyright Act, provided that attribution is given to the Cloud Security Alliance.

This document may be shared on professional and social media platforms in its original form with attribution.

This document was generated with AI assistance and has not undergone official CSA review and approval processes.

Key Takeaways

Between July and November 2026, Dragos researchers seeded roughly two million prompts with unique canary URLs and credentials. They found evidence that attackers had been copying prompt traffic from three separate AI hosting providers. The earliest of these compromises began in mid-July, and none of the three providers knew about it until Dragos notified them [1]. In two of the breaches, some of the stolen requests had been sent with the Zero Data Retention (ZDR) flag set [1]. A separate pattern involved no breach at all. An unnamed Chinese AI lab fetched secret URLs from more than half of the prompts sent to its models, including about two-thirds of requests marked "no data collection" or ZDR [1].

The disclosures from individual providers describe distinct failure modes. Phala reported an attacker who took control of an inference VM [4]. Together AI reported that stored prompt samples were exfiltrated after an AI coding agent committed a credential [5]. Novita reported a routing misconfiguration that forwarded some MiniMax requests to an upstream API for more than five months [6]. On November 22, 2026, OpenRouter off-boarded three providers, removed 125 endpoints from ZDR eligibility, and committed to continuous technical testing [2]. OpenRouter's documentation describes ZDR eligibility as based on provider data policies and agreements [3], and its new commitment to ongoing technical testing [2] indicates that ZDR status had not previously been independently verified.

Organizations should treat any secret that has appeared in a prompt sent through third-party inference providers as potentially exposed and rotate it. They should also begin seeding their own canaries. As currently implemented, ZDR is a contractual claim that requires independent verification, not a technical control.

Background

A growing share of AI traffic no longer travels from a customer directly to a single model developer, particularly for open-weight models. Instead, requests pass through a layered supply chain. Application code calls an AI router or gateway, and the router chooses among dozens of hosting providers serving the same model, often on price or latency. Some of those providers resell capacity from other providers, and inference marketplaces let individual sellers list endpoints at competitive prices. Every hop sees the

full plaintext of the prompt. That includes system instructions, retrieved documents, tool outputs, and any credentials an agent has read from its environment. Yet the customer often has a contract with only the first hop.

Zero Data Retention is one prominent industry response to the confidentiality problem this architecture creates. OpenRouter, a widely used independent router, defines ZDR as "no routine storage of prompts or responses after processing." Customers can require ZDR routing globally, per model family, or per request through a `zdr` provider parameter [3]. The documentation also explains how providers qualify. OpenRouter "works with providers to understand each of their data policies," sometimes through special agreements, and it treats an endpoint without a clear policy as one that retains and trains on data [3]. Several qualifications follow. In-memory prompt caching does not count as retention. ZDR applies to inference routing but not to plugins such as web search. For private endpoints, the customer's declaration is "your attestation," which OpenRouter "does not independently verify" [3]. Because this documentation page is updated over time, its current ZDR definition also carves out an exception for CSAM screening [3]. ZDR is therefore a classification derived from stated policy, and until recently neither the router nor the customer had a practical way to test it.

On November 19, 2026, HD Moore, Chief Research Officer at Dragos, published the results of a five-month experiment that tested this classification empirically [1]. The work started with Thinkst Canarytokens [7] and later moved to a purpose-built service. The service derives unique hostnames, URLs, and credentials from a per-prompt seed and runs the matching DNS, web, git, SSH, database, and mail services. The AWS keys were real keys attached to empty, fully logged accounts. As a result, any later use of a token could be traced to the exact prompt, model, router, and privacy setting that carried it [1]. Dragos also built a technique it calls "something like traceroute for AI inference." The technique embeds externally hosted images to reveal which systems fetch them. It also sends deliberately malformed requests so that each hop wraps the next hop's error in its own envelope [1].

Security Analysis

The Dragos findings fall into four categories of exposure. Each has a different cause and calls for a different response. The subsections below cover them in turn: direct compromise of providers, retention by marketplace sellers and undisclosed hops, retention that ignores privacy settings, and breach of data retained outside the request path. The section closes by assessing what these cases together mean for how ZDR can be assured.

Three Provider Compromises

The most serious finding is that attackers were "actively stealing data from three different hosting providers," and none of the three knew about the compromise until Dragos contacted them [1]. In the largest incident, attackers stole "a couple hundred" AWS keys. Each key had been sent in exactly one request to one provider during a few short windows in late October, and every request sent during those windows was taken. Some of the stolen keys came from requests that the provider had rejected before any model ran. Dragos reads this as evidence that the attacker was copying traffic "at the provider's front door" [1]. Dragos also reports that, in the other two breaches, some of the stolen requests had been marked as ZDR [1]. The attackers validated keys, sometimes within hours. They then enumerated S3 buckets, searched Secrets Manager and Parameter Store, and tried to enable Bedrock models, create Bedrock API keys, and add IAM users. In total they generated "tens of thousands of AWS API calls" against the bait accounts [1].

According to Dragos, one adversary appears to have compromised two providers at once. Secrets from both were used together from a single rented server. Over three weeks, the actor's tooling progressed from a script impersonating Chrome to Boto3 and then to the AWS CLI on Kali Linux [1]. On November 1, the same actor began using ARTEX, described as an LLM-driven penetration testing framework, to recheck whether keys were still valid. Dragos published two associated user-agent strings, `artex-agent` and `artex-judge-r121`, and could not find either in any public indicator list [1]. Security teams may want to add these strings to their CloudTrail detections. Agentic offensive tooling appearing after the theft suggests that the time between capturing a prompt and exploiting its credentials may shorten as such tooling spreads.

Because the bait prompts were processed alongside real customer traffic, Dragos concludes that real customer data was "almost certainly copied too" [1]. Dragos notes that one of the providers compromised in late October has already published a preliminary security notice [1]. Phala's preliminary report confirms an exposure. It describes canary evidence from a third-party researcher that is consistent with Dragos's late-October observations, although neither source names the other [1][4]. Phala describes two related intrusions, which it treats as a single incident. In one, an attacker "chained a legacy internet-facing service with management interfaces that relied on loopback access for trust" to create unauthorized confidential VMs and obtain credentials, which Phala dates to early November. In the other, a similar access pattern gave the attacker control of a newly deployed GPU inference server that was "serving production traffic without our full production security hardening," making in-flight request data visible. Phala's timeline places the start of the exposure window on October 21, before the confidential VMs were created [4]. Phala tells customers to treat DeepSeek V4 Flash 0731 requests routed through OpenRouter between October 21, 2026 00:00 UTC and November 8, 2026 05:35 UTC as exposed. Only canaries sent on October 21 and 22 were misused, but Phala keeps the wider window because the canaries

only sampled traffic [4]. Dragos also reports on the mid-July victim. That provider first attributed the activity to misconfigured tooling, then to a partner's data processing, and only later discovered compromised internal credentials. It had not publicly disclosed the incident when Dragos published [1].

For ZDR, the implication is direct. In Moore's words, "A privacy flag doesn't help when the attacker is already inside" [1]. ZDR governs what a provider intends to store. It offers no assurance about what an intruder positioned at request ingress can copy.

Retention by Design: Marketplaces, Resellers, and Undisclosed Hops

The second category of exposure required no exploitation at all. On an inference marketplace, a seller only had to list a model at the lowest price, turn on request logging, and wait [1]. Dragos observed sellers who appeared to be individuals on home broadband harvesting prompts and acting on what they found. In one case, a person opened two secret URLs in a desktop browser about a day and a half after the requests were sent and began guessing usernames. In another, someone on a Mac in France manually attempted root SSH logins with passwords taken from prompts and then tested AWS keys [1]. Prompts for the same model routed through a different seller were never touched. This led Dragos to conclude that "the leak follows the seller and not the marketplace as a whole" [1].

Route tracing exposed a wider transparency problem. Across roughly 2,000 traced routes, "several hundred" included a system that Dragos could not identify. In "a handful," the test image was fetched by a company that was not on the advertised path [1]. One router was found quietly forwarding some traffic into the marketplace. The router says it has since stopped this undisclosed hop [1]. Dragos also observed one seller adding about a thousand hidden tokens to every prompt [1]. That behavior raises integrity concerns, such as injected instructions or altered outputs, on top of the confidentiality concerns. In short, the customer's view of the supply chain often did not match the path the traffic actually took.

Privacy Settings Ignored

The third category involved organizations that were neither breached nor abusing credentials, yet still retained and processed prompts in violation of the privacy settings under which the prompts were sent. Between July and October, an unnamed Chinese AI lab fetched secret URLs from more than half of the prompts sent to its models, including about two-thirds of the requests marked "no data collection" or ZDR. The fetches typically came two to three days after the request [1]. Prompts sent to several U.S. hosting providers that resell the lab's models triggered the same scraper. Later, a git client on a different cloud tried to clone private repositories referenced in prompts, again only for that lab's models [1].

Dragos describes further cases of retention beyond the original request. One large U.S. provider was still crawling a secret URL more than two months after a single call to its search endpoint [1]. In another case, a secret link appeared in a Feishu chat 24 days after the original request. Within two minutes it was opened from six devices in Beijing and Shanghai. Only the first eight characters of the link had been disclosed to the provider, so whoever shared it must have been reading a stored copy of the full prompt [1]. Dragos associates this request with the ZDR flag. However, its account of the routing is inconsistent: it describes the prompt as sent "to one Chinese provider" and also as sent "to a U.S.-based provider with the ZDR flag." The ZDR association in this case should be read with caution.

Novita's disclosure shows how such flows can arise without malicious intent. A routing configuration issue caused a forwarding path meant for specific customers to apply more broadly. As a result, some MiniMax requests sent directly to Novita or through OpenRouter between May 28 and November 11, 2026 were "forwarded to MiniMax's official API rather than processed on Novita-hosted infrastructure," where request contents "may have been accessed or retained by a third party" [6]. Novita observed access to embedded URLs and hostnames but no use of AWS test keys. It suggests automated content review as a possible explanation and says it is confirming this with MiniMax [6]. Atlas Cloud has separately reported that a third party opened links from prompts sent to MiniMax's official API, including MiniMax prompts sent through Atlas Cloud in the third and early fourth quarters of 2026 [8]. Whatever the cause, a customer who chose a specific host for data residency or retention reasons ended up under a different processor's terms without being told.

Retention Outside the Request Path

Together AI's disclosure describes a fourth pattern. Together collected prompt samples from serverless inference for software testing and stored them in a private Hugging Face repository. While Together was developing its public "xorl" GitHub repository, "an AI coding agent committed a credential" that granted access to that repository. During the exposure window, an unauthorized party obtained the credential and used it to download the stored samples [5]. The affected data came from MiniMax M3, GLM-5.2, and Kimi K2.6 requests, primarily from June 24, July 4, July 8, and July 14, 2026. The attacker searched the samples for credentials and tested them, and five of those credentials were Dragos canaries [5]. Because Dragos began seeding in July [1][4], those canaries must have come from the July sample dates rather than June 24. Together's notice does not say whether the sampled requests had been sent with ZDR enabled.

Together has since made ZDR the default for new accounts and introduced scoped, short-lived credentials and rules for AI coding agents [5]. The incident shows that prompts retained for legitimate operational purposes take on the security posture of every system and developer workflow that touches them, including autonomous coding agents.

The Verification Gap

Taken together, these findings show a gap between how ZDR is represented and how it can be assured. The table below summarizes the failure modes and asks two questions about each. Would a correctly honored ZDR policy have prevented it? Could customer canaries have detected it?

Failure mode	Example from the research	Would a correctly honored ZDR policy prevent it?	Detectable by customer canaries?
Intrusion at provider ingress or inference host	Three provider compromises [1]; Phala inference VM takeover [4]	No	Yes, if the token is acted on
Seller-side logging on a marketplace	Individual sellers harvesting prompts [1]	Not applicable (sellers not bound)	Yes, if the token is acted on
Undisclosed forwarding to another processor	Router-to-marketplace hop; Novita forwarding to MiniMax API [1][6]	Only if the downstream party is also bound	Yes, with route tracing or if the token is acted on
Policy non-compliance by the model operator	Lab scraping ZDR-flagged prompts [1]	Yes	Yes, if the token is acted on
Breach of operationally retained samples	Together AI test-sample exfiltration [5]	Depends on definition: only if ZDR explicitly covers testing samples	Yes, if the token is acted on

The detection column has an important limit. Canaries reveal leakage only when someone acts on a token, for example by fetching a URL or trying a key. Novita, for instance, observed URL access but no use of the AWS keys [6]. A passive collector that stores prompts and never acts on their contents would not be detected.

OpenRouter's response shows the practical effect of this gap. Its incident post says it has "off-boarded three providers," "removed 125 endpoints from Zero Data Retention (ZDR) eligibility," and notified all potentially affected customers. The 125 endpoints include some removed as a precaution, beyond those the researchers flagged [2]. OpenRouter is also "adapting the researchers' testing approach so we can

run these checks on an ongoing basis" and "conducting technical verification of network paths." It will require "appropriate assurances and validation before restoring ZDR eligibility" [2]. OpenRouter reports that the research found no breach of OpenRouter itself, and it has not publicly named the three off-boarded providers [2]. The post links to notices from Atlas Cloud [8], Novita, Phala, and Together, but it notes that not every affected provider has posted publicly [2]. Atlas Cloud says it removed its models from OpenRouter by its own decision [8]. Moore sums up the underlying problem: "a privacy flag only binds the first hop" [1].

This has significant consequences for assurance programs. Attestations, data processing agreements, and policy reviews establish what a provider has promised. They do not show what happens to plaintext at runtime across subprocessors the customer may not know exist. Canary-based testing is an imperfect instrument. It samples traffic rather than observing all of it, and an activation shows that a prompt escaped without always showing which hop leaked it [1][4]. It is nonetheless a practical method that produces evidence rather than representations. Its findings led OpenRouter to off-board three providers and suspend ZDR eligibility for 125 endpoints [2].

Recommendations

The recommendations below are organized by time horizon. Immediate actions focus on finding and containing exposure that may already have happened. Short-term mitigations reduce what can leak and improve detection. Strategic considerations address how much of the inference supply chain an organization should trust without evidence.

Immediate Actions

Organizations that have sent traffic through AI routers, inference marketplaces, or third-party hosts of open-weight models since mid-2026 should first establish their exposure. They should review the provider notices from Phala [4], Together AI [5], Novita [6], and Atlas Cloud [8], along with any OpenRouter customer notification [2], and match the affected models and time windows against their usage logs. Any credential, token, connection string, or private URL that may have appeared in a prompt during those windows should be rotated. In agentic systems, that includes credentials an agent could have read from files, environment variables, or tool outputs, not only those a user typed. Security teams should also search cloud audit logs for the ARTEX user-agent strings published by Dragos [1] and for unusual enumeration of secrets stores and Bedrock configuration.

Teams should also confirm that their router and gateway configurations actually enforce the restrictions they intend. On OpenRouter, this means checking that ZDR enforcement and data-collection restrictions are set at the account or request level, and that the provider allowlist is explicit rather than driven by price [3]. A Thinkst Canarytoken [7] placed in the context that production agents already send provides a free first signal. Canaries should be placed where the agent is instructed not to act on them. When triaging activations, teams should exclude the organization's own egress and tool-fetch infrastructure, as well as any legitimate provider URL-fetch features such as web-search plugins. Any activation that remains should be treated as a security incident.

Short-Term Mitigations

Over the coming quarter, organizations should reduce what can leak and improve their ability to detect leaks. Secrets should be filtered or redacted before prompts leave the organization's boundary. As Dragos notes, however, such filters miss sensitive content that is not shaped like a credential [1]. Agents should run with scoped, short-lived credentials so that a captured secret has a narrow blast radius and a short useful life. Canary seeding should become a continuous program. Issuing unique tokens per provider and per privacy setting allows activations to be attributed, and AWS-style decoy credentials should be tied to empty, logged accounts.

Procurement and vendor risk teams should update their AI provider assessments to require disclosure of every hop that receives plaintext, including resellers and upstream APIs. Contracts should oblige providers to give notice of any routing change that alters the processing party. Contracts should also define ZDR to cover operational sampling, testing datasets, and abuse-review pipelines. The Together AI and Novita cases each fell outside the narrow definition of "no routine storage after processing" [3][5][6]. Organizations should avoid inference marketplaces for any workload that contains sensitive data. Dragos notes that some marketplace capacity comes from stolen keys and that some services accept only cryptocurrency [1]. This guidance gives up some of the cost and model-access benefits that marketplaces offer. The seller-harvesting evidence, though, indicates that those benefits do not justify the exposure for sensitive workloads.

Strategic Considerations

In the longer term, organizations must decide how much of the inference supply chain they are willing to trust without evidence. Dragos frames the options as a spectrum [1]. At one end, self-hosting on infrastructure the organization controls is the only option in which no third party reads prompts, though it carries substantial hardware cost. Self-hosting shifts the burden of hardening to the organization rather than removing it, as the Phala host compromise shows [4]. Direct contracts with first-tier providers are

"the smallest exposure you can buy." Routers are workable when the customer, not the price signal, chooses the upstream providers [1]. Organizations should tier workloads by sensitivity and deliberately map each tier to one of these options.

At the ecosystem level, ZDR should evolve from a self-declared policy tag into a property that can be verified. That would likely require independent, continuous canary testing by routers and assessors. It would also require published route transparency, attestation of inference-host configurations, and third-party audit evidence that covers runtime data handling rather than documented policy alone. Phala's incident is instructive because it happened on a platform built around confidential VMs, yet a newly deployed server served production traffic before full hardening [4]. Confidential computing and hardware attestation can strengthen assurance, but only if attestation covers the actual serving path and is checked continuously.

CSA Resource Alignment

CSA's **AI Controls Matrix (AICM) v1.1** [9] provides the control vocabulary for the gaps these incidents expose. Its Supply Chain Management, Transparency, and Accountability (STA) domain addresses the lack of subprocessor visibility shown by undisclosed hops and reseller forwarding. Its Data Security and Privacy Lifecycle Management (DSP) domain covers the retention and data-flow controls that ZDR claims are meant to satisfy. AICM's shared-responsibility model spans model providers, orchestrated service providers, application providers, and AI customers, and it maps closely to the router, provider, and marketplace chain described here. Organizations can use it to assign which party is accountable at each hop.

The **AICM v1.1 Auditing Guidelines for Application Providers** [10] translate those controls into evidence expectations for each control and support continuous compliance monitoring. The Dragos findings suggest that auditors evaluating applications built on third-party inference should ask for evidence of runtime data-flow verification, such as canary results or route tracing, alongside policy documents and data processing agreements.

CSA's **Data Security within AI Environments** [11] maps AICM DSP controls across the AI data lifecycle, including inference. It also discusses privacy-enhancing technologies and vendor risk assessment for AI platforms. Its lifecycle framing helps make clear that prompts retained for testing, abuse review, or caching are new data stores that need their own controls, as the Together AI incident shows.

Finally, the **STAR for AI** program and its **Auditors Guidance Document** [12] provide the assurance pathway through which AI providers can demonstrate their control implementation to customers. The verification gap described in this note argues that STAR for AI assessments of inference providers should pay specific attention to runtime prompt handling, subprocessor disclosure, and retention claims. The goal is third-party evidence that a ZDR designation holds in practice, not only on paper.

References

[1] Moore, HD. "Three breaches, endless leaks: Your AI supply chain needs canaries." Dragos, November 19, 2026.

[2] OpenRouter. "Data Processing Incidents on Inference Providers." OpenRouter, November 22, 2026.

[3] OpenRouter. "Zero Data Retention." OpenRouter Documentation, accessed November 23, 2026.

[4] Phala CERT Team. "Phala November Preliminary Incident Report." Phala, November 10, 2026.

[5] Prakash, Vipul Ved. "Security event: Prompt leakage for requests in June and July of 2026." Together AI, November 21, 2026.

[6] Novita AI. "Privacy Notice Regarding MiniMax Requests on Novita." Novita AI, November 22, 2026.

[7] Thinkst Applied Research. "Canarytokens." Thinkst, accessed November 23, 2026.

[8] Atlas Cloud. "Security notice: MiniMax's official API and how we protect your data." Atlas Cloud, November 23, 2026.

[9] Cloud Security Alliance. "AI Controls Matrix v1.1." CSA, 2026.

[10] Cloud Security Alliance. "AICM v1.1 Auditing Guidelines for Application Providers (AP)." CSA, 2026.

[11] Cloud Security Alliance. "Data Security within AI Environments." CSA, December 2025.

[12] Cloud Security Alliance. "Auditors Guidance Document STAR for AI Certification." CSA, March 2026.

© 2026 Cloud Security Alliance. AI-assisted Rapid Research.