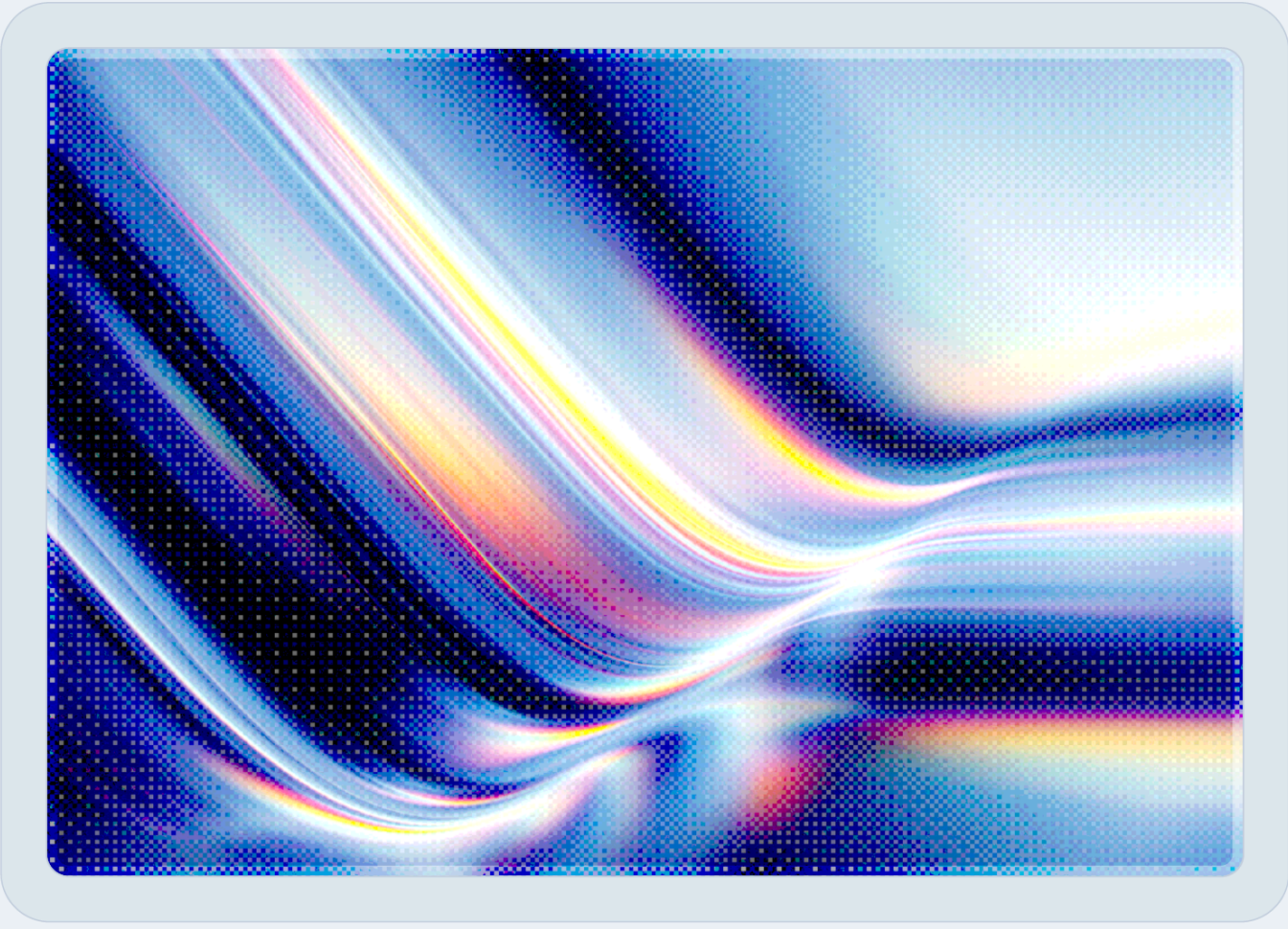
CSAI Foundation | Cloud Security Alliance

Identity Confusion by Design: The Grafana MCP SSRF

What a Session-ID Authentication Flaw Reveals About Every MCP Server

2026-09-08

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Key Takeaways

Security researchers at Pillar Security disclosed two chained flaws in the open-source Grafana MCP server: an authentication bypass rooted in treating session identifiers as credentials, and a server-side request forgery (SSRF) vulnerability tracked as CVE-2026-19516 (CVSS 9.1) in the `grafana_api_request` tool [1][2][3]. Chained together, the two flaws let an unauthenticated caller invoke tools using the MCP server's own Grafana service-account privileges, then redirect the server's outbound requests toward internal infrastructure and cloud metadata endpoints — in Pillar's proof of concept, far enough to retrieve AWS Instance Metadata Service (IMDSv2) credentials [1]. The evidence assembled below suggests the root cause is not a coding mistake specific to Grafana, but a category error the Model Context Protocol's own specification anticipated but did not prevent: `Mcp-Session-Id` values are defined for conversation-state tracking, and the spec merely recommends they be cryptographically generated while separately noting that servers "SHOULD implement proper authentication for all connections" — leaving the two concerns easy to conflate in practice [4]. Grafana shipped mcp-grafana v1.1.0 with optional bearer-token authentication for its SSE and streamable-HTTP transports within eight days of the report, and independent internet-wide scanning suggests this pattern is not confined to Grafana: Censys identified over 12,500 internet-accessible MCP services in an April 2026 scan, and the subset of servers the researchers examined in detail were all reachable without any authentication [1][6]. The underlying pattern — an MCP server that holds a privileged backend credential and can be tricked into using it, or its network position, on a caller's behalf — is a textbook confused-deputy problem, and it is likely to keep recurring as long as MCP server authors and deployers continue treating protocol-layer identifiers as trust boundaries [7].

Background

On September 2, 2026, the AI agent security firm Pillar Security published a detailed technical writeup describing two vulnerabilities it had discovered and reported in Grafana's official Model Context Protocol server, `mcp-grafana` [1]. The project connects AI agents and MCP-compatible clients to a Grafana instance so that a language model can query dashboards, metrics, and alerts on a user's behalf, and it has accumulated roughly 1.9 million cumulative downloads on Docker Hub, a figure that suggests wide deployment within the MCP ecosystem, though comparative download counts for other MCP observability integrations were not available [1]. Grafana itself is a foundational component of many

organizations' monitoring stacks, frequently holding read access to infrastructure telemetry, incident data, and — through its data source configurations — credentials or proxied access to the systems it observes.

Pillar reported both findings to Grafana through the company's Intigriti-hosted bug bounty program on August 2, 2026. Grafana accepted both reports on August 10 and released mcp-grafana v1.1.0 the same day, adding optional bearer-token authentication for the server's SSE and streamable-HTTP transports. CVE-2026-19516, covering the SSRF component, was published on August 11, and Pillar's finding was added to Grafana's security hall of fame on August 12 [1][3]. What makes the case worth a closer look is not the disclosure timeline but the architecture of the failure itself, because it is a failure mode built directly into how the Model Context Protocol's HTTP transports are commonly implemented — not an idiosyncrasy of one vendor's code.

MCP defines a client-server architecture in which an AI agent (the "host," acting through an MCP client) connects to a server that exposes tools, resources, and prompts. When a server uses the Streamable HTTP transport, it may issue an `Mcp-Session-Id` header at initialization to track which requests belong to the same logical conversation across a stateless HTTP connection [4]. The specification is explicit that this identifier should be "globally unique and cryptographically secure" and separately states, in its security warning for the transport, that servers "SHOULD implement proper authentication for all connections" [4]. Those are two different requirements addressing two different problems — session continuity and caller identity — and the Grafana MCP server, in its pre-1.1.0 form, effectively used the first as a stand-in for the second.

Security Analysis

Issue one: a session identifier is not a credential

The first flaw Pillar identified was that mcp-grafana validated the *format* of a session ID rather than authenticating the party presenting it. Because the server never issued a credential check independent of the session header, a remote caller who had never authenticated could generate a locally crafted, session-shaped string — for example, matching the pattern `mcp-session-<uuid>` — and attach it to `tools/list` and `tools/call` requests. The server accepted these requests as if they came from a legitimate, already-initialized session and proceeded to execute the requested tool, including `grafana_api_request`, using the Grafana service-account bearer token configured for the MCP

server itself [1]. From Grafana's point of view, the resulting API activity was fully authenticated and audit-logged — it simply reflected the wrong principal. The caller had authenticated nothing, yet the request that reached Grafana carried the server's own privileged identity.

This is a distinction the MCP specification tries to draw sharply but that implementers can easily blur, and a related protocol proposal, SEP-1359 on protocol-level sessions, makes the point explicitly: "Session IDs identify conversation context only. Every request with authorization requirements MUST include valid authentication credentials independently of any session ID" [5]. Grafana's initial implementation had, in effect, allowed session continuity to substitute for authorization. Because MCP session IDs travel in a plain HTTP header alongside — but functionally separate from — any `Authorization` header, the two are structurally easy to conflate during implementation. One plausible explanation for why this conflation recurs is that MCP client tooling and quick-start examples tend to emphasize getting the session handshake working before wiring up authentication — though no survey confirms this pattern — which would make the two concerns easier to merge conceptually during development, particularly for internal-facing tools where "getting it working" often precedes "locking it down."

Issue two: the server as an SSRF proxy

The second and more severe flaw, CVE-2026-19516 (CVSS 9.1), lived in the `grafana_api_request` tool itself. That tool accepted a caller-supplied `X-Grafana-URL` header, which the server used to determine the destination of its outbound API call — along with caller control over the HTTP method, path, and body of that request [1][2]. Grafana's own backend prevented the server-account bearer token from being forwarded to hosts outside the configured Grafana instance, which limited straightforward credential theft. But the MCP server itself still automatically connected to whatever destination the caller specified, from its own network vantage point. An attacker who could reach the MCP server — including, thanks to the first flaw, one who had never authenticated to anything — could direct its outbound requests toward internal services, administrative interfaces, or cloud metadata endpoints that were never meant to be reachable from outside the network.

Pillar demonstrated the practical severity of this by chaining the SSRF into an IMDSv2-style attack against AWS's Instance Metadata Service: a caller-controlled `PUT` request carrying the appropriate TTL header first obtained a metadata session token, which a follow-up caller-controlled request then used to retrieve the host's IAM role credentials [1]. In cloud deployments, IMDS-credential theft via SSRF is widely regarded as one of the most consequential outcomes the bug class can produce, because it converts a request-forwarding flaw into full compromise of whatever privileges the host's cloud identity carries — well beyond anything Grafana itself controls. Pillar's own framing of the two flaws together is

instructive: "the caller provides the instruction and the server provides the reach," and security depends on keeping those two things connected rather than letting an unauthenticated instruction borrow an authenticated server's network position [1].

A pattern, not an anomaly

Neither flaw is exotic by conventional web application security standards. Broken authentication has populated vulnerability taxonomies for roughly two decades, and SSRF has been formally recognized as a top-tier web application risk since OWASP added it as its own category in the 2021 Top 10 [12]. What is distinctive here is the setting: an MCP server sits at the intersection of a language model's non-deterministic tool-calling behavior, a backend service's privileged credential, and — very often — a network position with reach into infrastructure a browser-based client would never have. Censys's April 2026 internet scan identified more than 12,500 MCP services running on the open internet, and the servers Censys examined in detail were reachable without authentication, discoverable simply by probing the protocol's own tool-listing and resource-enumeration endpoints [6]. The OWASP MCP Top 10 project independently designated "insufficient authentication and authorization" and "token mismanagement" as two of its ten flagship risk categories for exactly this reason [11]. The Grafana case is therefore best read as a well-documented instance of a structural problem, not an isolated implementation bug — a confused-deputy pattern in which a server with more authority than its caller is manipulated into exercising that authority on the caller's behalf [7].

It is also worth noting what did *not* fail here: the MCP specification's own security warning for Streamable HTTP transports already told implementers to validate the `Origin` header, bind local servers to localhost, and implement proper authentication [4]. Grafana's server satisfied the letter of "supports authentication" once tokens were configured for outbound Grafana calls, while leaving inbound caller authentication optional until v1.1.0. That gap between what a specification recommends and what a default configuration enforces appears to be where most of the practical risk in this incident lived.

Recommendations

Immediate Actions

Organizations running mcp-grafana should upgrade to v1.1.0 or later immediately and set `MCP_GRAFANA_SERVER_TOKEN` to require a bearer token from every caller on SSE and streamable-HTTP transports; unauthenticated requests should be rejected with HTTP 401 before any tool executes [1]. Security teams should also inventory every MCP server they operate — not just Grafana's — and

confirm each one enforces inbound authentication independent of session-ID validation, since the same pattern can exist wherever a server treats a protocol-level identifier as a proxy for identity. Any MCP tool that accepts a caller-supplied destination, host, or URL parameter (as `grafana_api_request` did through `X-Grafana-URL`) warrants immediate review, because that is the shape of an SSRF-capable interface regardless of the specific vendor.

Short-Term Mitigations

Beyond the immediate patch, teams should implement destination allowlisting for any MCP tool capable of making outbound HTTP requests, resolving hostnames before applying restrictions and re-validating the resolved address at connection time to close DNS-rebinding gaps [1]. Cloud metadata services should be blocked by default from any network segment an MCP server occupies — via IMDSv2 enforcement, hop-limit restrictions, or network policy — so that an SSRF finding cannot cascade into cloud credential theft even if a future authentication gap reappears. Arbitrary HTTP method and header forwarding should be avoided in MCP tool implementations unless there is a specific, narrowly scoped reason for it, and MCP service accounts should be restricted to the minimum backend permissions the tool's declared function actually requires, so that a confused-deputy exploit yields the smallest possible blast radius.

Strategic Considerations

At a program level, organizations deploying MCP servers should stop treating protocol-layer conveniences — session IDs, connection state, transport-level headers — as substitutes for an explicit authentication and authorization boundary, and should require every MCP server in their environment to demonstrate that inbound caller identity is verified independently of session continuity before it is approved for production use. Given that thousands of internet-reachable MCP servers have already been found running without authentication [6] — regardless of whether that exposure was intentional or an oversight — security teams should also build MCP-specific discovery and continuous scanning into their existing attack-surface-management programs, since shadow or unmanaged MCP deployments will not surface through conventional asset inventories. Finally, this incident is a useful, low-drama case study for internal AI governance conversations: it demonstrates concretely how "the caller provides the instruction, the server provides the reach" applies well beyond Grafana, to any MCP server that holds a backend credential its caller does not.

CSA Resource Alignment

This incident sits squarely within territory CSA's Model Context Protocol research has already covered. CSA's **Agentic MCP Security Best Practices Guide** [8] addresses the exact failure mode at the heart of the Grafana disclosure: it requires that remote MCP server connections use OAuth 2.1 with mandatory PKCE, that session tokens be bound to cryptographic proof of the requesting agent's identity so a stolen token cannot be replayed by a different caller, and that high-privilege tool operations require just-in-time re-authorization rather than relying on permissions granted once at connection time. Grafana's pre-1.1.0 server violated all three principles at once: it required no OAuth 2.1 authentication flow of any kind before v1.1.0, it accepted a caller-crafted session identifier in place of a bound credential, and it let a single tool invocation reach far beyond what the calling context should have justified.

Confused Deputy Attacks on Autonomous AI Agents [7] is CSA's dedicated treatment of the broader pattern this incident exemplifies: a privileged agent or server manipulated into misusing the authority it holds on a caller's behalf. That note's account of agents inheriting credentials and permissions they cannot adequately verify describes precisely what happened when mcp-grafana executed `grafana_api_request` with its own service-account token in response to an unauthenticated, self-issued session ID. The **MAESTRO** [9] threat-modeling framework offers the structural lens for tracing this failure across layers — the vulnerability spans MAESTRO's deployment/infrastructure layer (an internet-reachable server with a privileged credential) and its security/compliance layer (missing authentication and egress controls), and threat modelers assessing their own MCP deployments should walk both layers explicitly rather than treating authentication as a single checkbox.

More broadly, this incident reinforces why CSA's **AI Controls Matrix (AICM v1.1)** [10] treats non-human and machine-caller identity as a first-class Identity and Access Management concern for agentic systems, and why AICM's Application and Interface Security domain calls for validating trust boundaries at every point where an AI agent's tool call crosses into a privileged backend. Readers assembling a broader MCP governance program should treat AICM as the control vocabulary that ties this incident's specific lessons — authenticate the caller independently of session state, and never let a tool forward an arbitrary destination without an allowlist — into an auditable enterprise baseline.

References

[1] Pillar Security. "Valid, But Never Issued: Session Spoofing and SSRF in Grafana MCP." Pillar Security Blog, September 2, 2026.

[2] SC Media. "Grafana Fixes Critical SSRF Flaw Affecting Grafana MCP Servers." SC Media, September 2026.

[3] SentinelOne. "CVE-2026-19516: mcp-grafana SSRF Vulnerability." SentinelOne Vulnerability Database, 2026.

[4] Model Context Protocol. "Transports – Streamable HTTP Security Warning and Session Management." Model Context Protocol Specification, 2025-03-26.

[5] modelcontextprotocol/modelcontextprotocol GitHub repository. "SEP-1359: Protocol-Level Sessions for MCP." Model Context Protocol Enhancement Proposals, 2026.

[6] Censys. "MCP Servers on the Internet." Censys Research Blog, 2026.

[7] Cloud Security Alliance. "Confused Deputy Attacks on Autonomous AI Agents." Cloud Security Alliance AI Safety Initiative, March 23, 2026.

[8] Cloud Security Alliance. "Agentic MCP Security Best Practices Guide." Cloud Security Alliance AI Safety Initiative, March 27, 2026.

[9] Cloud Security Alliance. "Agentic AI Threat Modeling Framework: MAESTRO." Cloud Security Alliance, February 6, 2025.

[10] Cloud Security Alliance. "AI Controls Matrix (AICM) v1.1." Cloud Security Alliance, 2026.

[11] OWASP Foundation. "OWASP MCP Top 10." OWASP MCP Top 10 Project, 2026.

[12] OWASP Foundation. "A10:2021 – Server-Side Request Forgery (SSRF)." OWASP Top 10:2021, 2021.

