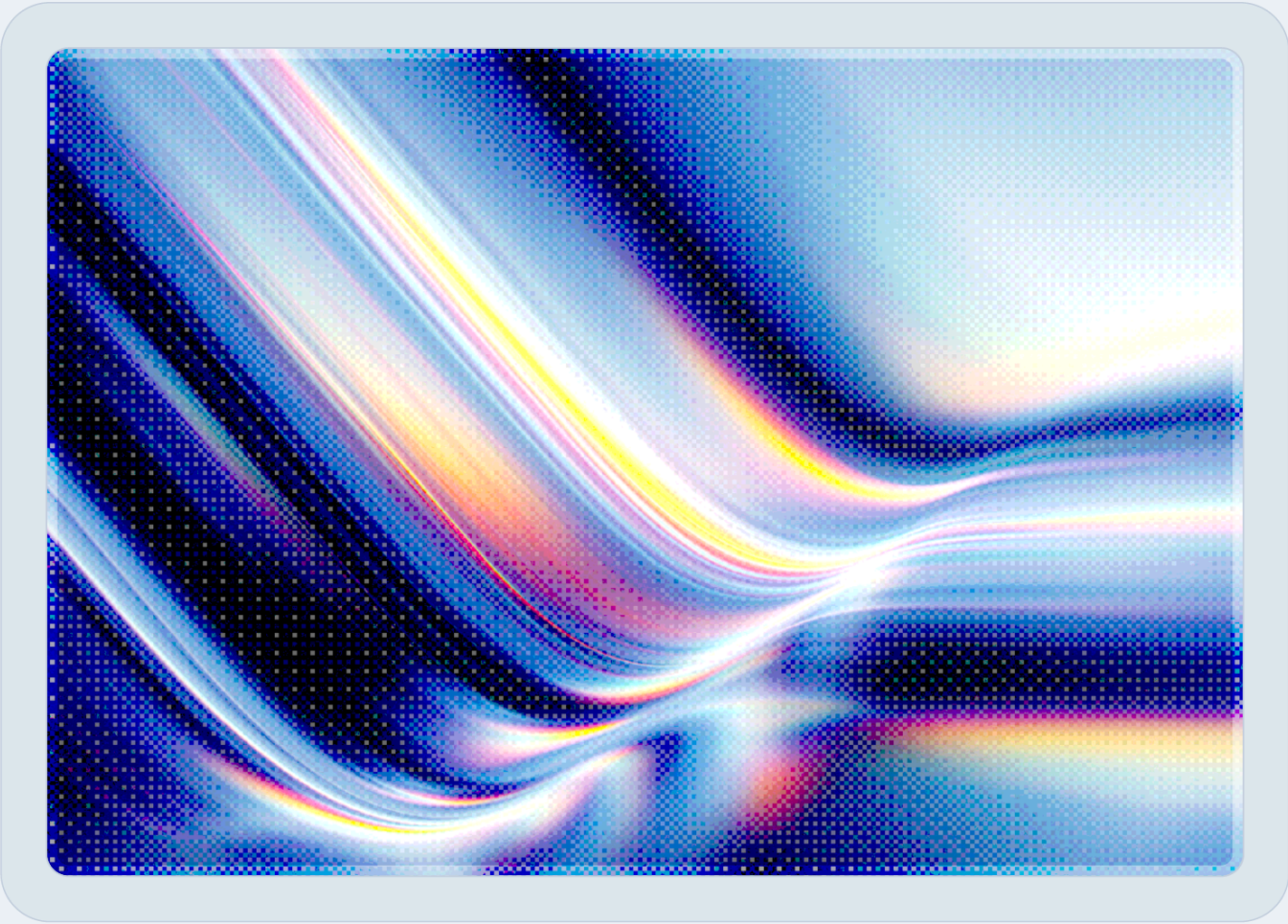
CSAI Foundation | Cloud Security Alliance

# Auth Bypass in AI Orchestration: CVE-2026-44338

PraisonAI's Fail-Open API Design and the Sub-Four-Hour Exploit Window

2026-05-14

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

# Key Takeaways

- CVE-2026-44338 is a CVSS 7.3 (High) authentication bypass affecting PraisonAI versions 2.5.6 through 4.6.33, caused by authentication being disabled by default in the framework's legacy Flask API server.
- Targeted scanning of exposed instances began within three hours and 44 minutes of the public advisory becoming available — consistent with an accelerating pattern of near-immediate post-disclosure exploitation across the agentic AI tooling ecosystem.
- Two unprotected API endpoints allow unauthenticated callers to enumerate agent configurations (`GET /agents`) and trigger pre-configured workflows without credentials (`POST /chat`).
- Organizations running PraisonAI should upgrade immediately to version 4.6.34, audit network exposure of all agent API surfaces, and treat any internet-facing agentic orchestration endpoint as critical infrastructure requiring authentication controls.
- This vulnerability reflects a documented structural risk pattern that security researchers have flagged across open-source agentic AI frameworks: authentication is repeatedly shipped as optional operator configuration rather than enforced by default, creating exposure whenever frameworks migrate from development contexts to production infrastructure.

# Background

PraisonAI is an open-source Python framework for building and coordinating multi-agent AI workflows. Created by MervinPraison and distributed via PyPI, it enables developers to define multiple specialized AI agents in configuration files and orchestrate their interactions through a unified interface. The framework supports integration with large language models and exposes an API server that allows external callers to list configured agents and submit tasks for execution. PraisonAI has been used across developer and research contexts for building agentic applications, research prototypes, and automated workflows.

On May 11, 2026, GitHub published security advisory GHSA-6rmh-7xcm-cpxj disclosing CVE-2026-44338, a high-severity authentication bypass in PraisonAI's Flask-based API server [1]. The vulnerability was assigned a CVSS 3.1 base score of 7.3 (High) with a network attack vector (AV:N) and no required privileges or user interaction. It carries three Common Weakness Enumeration identifiers: CWE-306 (Missing Authentication for Critical Function), CWE-668 (Exposure of Resource to Wrong Sphere), and CWE-1188 (Insecure Default Initialization of Resource) [2]. Security researcher Shmulik Cohen of Sysdig received credit for the discovery. While the CVE was coordinated privately before the advisory became public, the National Vulnerability Database record appeared on May 8, 2026, and the GitHub advisory became publicly visible on May 11, 2026 [1][3]. The affected version range spans 2.5.6 through 4.6.33, and the issue was resolved in version 4.6.34.

The breadth of the affected version range and the network-accessible nature of the vulnerable service mean that exposure depends on deployment choices rather than version alone. Organizations that have deployed PraisonAI in shared environments, on servers accessible beyond a developer's local machine, or integrated into automated pipelines that bind the API server to network interfaces should treat remediation as urgent.

---

# Security Analysis

## The Fail-Open Design Pattern

The root cause of CVE-2026-44338 is not a transient coding error — it is a design pattern: the framework ships with authentication disabled as the default configuration, requiring operator action to reach a secure state. The legacy Flask API server in PraisonAI sets `AUTH_ENABLED = False` and `AUTH_TOKEN = None` by default. The `check_auth()` helper function returns `True` whenever authentication is disabled, meaning the security gate is structurally bypassed when the server starts in its default configuration [1]. This is a fail-open design: in the absence of operator intervention to enable authentication, every request is treated as trusted.

When launched directly, the server binds to `0.0.0.0:8080`, making it accessible on all network interfaces rather than only the local loopback address. The combination of fail-open authentication and a promiscuous bind address means that any network-reachable instance of PraisonAI running the API server in its default configuration is an unauthenticated service endpoint. Because exposure is the default state — not the result of misconfiguration — no operator error is needed to place a PraisonAI API server in an unauthenticated, network-accessible condition.

## Exposed Attack Surface

Two API endpoints lack functional protection in unpatched versions. The `GET /agents` endpoint returns configured agent metadata, including agent definition file names and agent lists drawn from the operator's `agents.yaml` configuration. This allows an unauthenticated caller to enumerate the agent topology — learning what agents are configured, what roles they serve, and what files define their behavior — without supplying any credentials. The `POST /chat` endpoint executes PraisonAI workflows without authentication validation; a caller submitting a minimal JSON body (`{"message": "text"}`) can trigger the full preconfigured agent workflow [1]. The security consequences of that workflow execution depend entirely on what tools, permissions, and capabilities the operator has granted to the configured agents.

This distinction is significant for risk assessment. A PraisonAI deployment whose agents have access only to read-only knowledge bases presents a materially different risk profile than one where agents are authorized to execute code, access credentials, write files, or call external APIs. CVE-2026-44338 does not determine what an attacker can accomplish; it removes the authentication barrier that would otherwise prevent them from trying. The CVSS 7.3 base score reflects a moderate impact assumption across confidentiality, integrity, and availability dimensions, but for deployments where agents operate with significant downstream authorization, the business risk may exceed what the base score implies.

## Exploitation Timeline

Sysdig threat researchers observed the first targeted exploitation attempt at 17:40 UTC on May 11, 2026 — three hours, 44 minutes, and 39 seconds after the GitHub advisory became publicly available [3]. The activity consisted of two scanning passes approximately eight minutes apart, with each pass executing roughly 70 requests across approximately 50 seconds. The traffic originated from IP address 146.190.133.49, a DigitalOcean host in AS14061, with a User-Agent string identifying the tool as "CVE-Detector/1.0" [3]. The observed activity targeted only the `GET /agents` endpoint, consistent with an initial reconnaissance phase focused on confirming vulnerable instances before any attempt at workflow execution.

The speed of exploitation reflects a pattern Sysdig characterized as an emerging norm across the agentic AI tooling ecosystem. Similar rapid post-disclosure targeting was observed following vulnerability disclosures in Marimo, LMDeploy, and Langflow [3]. Security researchers, including Sysdig's threat research team and Black Duck AI researcher Vineeta Sangaraju, have characterized this speed as consistent with an emerging norm in which attackers leverage automated scanning pipelines

incorporating AI-assisted components to move from advisory publication to active exploitation in hours rather than days [3][4]. The implication for defenders is that patch timelines measured in days may be structurally insufficient for AI infrastructure components.

### Threat Scenarios

The practical consequences of an exploited CVE-2026-44338 instance depend on the authorization scope granted to configured agents, but three primary exploitation scenarios follow from the technical analysis. The scenario documented in the wild is reconnaissance: the `/agents` endpoint itself becomes an intelligence asset, with configuration details about internal tooling, workflow design, and agent file locations informing follow-on targeting even before any workflow execution is attempted [3]. Where agents have been authorized to execute code, access file systems, or call external services, unauthenticated workflow execution enables data exfiltration or lateral movement through systems the agents are permitted to touch. In financially motivated scenarios, an attacker sends repeated POST requests to `/chat` to exhaust the LLM API credits associated with the deployment's API keys, inflicting direct financial cost on the operator without requiring any code execution capability.

---

## Recommendations

### Immediate Actions

Organizations running PraisonAI should prioritize three actions immediately. Any deployment running versions 2.5.6 through 4.6.33 should be upgraded to version 4.6.34 or later; the GitHub security advisory provides upgrade guidance [1]. Prior to or in parallel with upgrading, operators should audit whether the PraisonAI API server is reachable outside the local host; any deployment accessible from a network segment broader than a single developer machine, and particularly any deployment reachable from the internet, should be isolated pending remediation. Operators should also review agent tool permissions to understand what an unauthenticated attacker could have triggered during the exposure window — if agents have authorizations broader than the deployment requires, that scope should be reduced regardless of patch status.

## Short-Term Mitigations

For organizations that cannot upgrade immediately, several controls reduce exposure without requiring a code change. Placing the PraisonAI API server behind a reverse proxy configured to enforce authentication — HTTP Basic, OAuth 2.0, or an API key validated at the proxy layer — prevents unauthenticated access even when the underlying server's default configuration is unchanged. Network segmentation restricting API server access to known client addresses provides a complementary layer of defense. Enabling detailed access logging at the proxy or network layer creates visibility into probing activity; "CVE-Detector/1.0" was the User-Agent string observed in the May 11 scanning activity and may be useful for historical log review, but defenders should note that User-Agent strings are trivially configurable and should not be treated as a reliable indicator for future exploitation attempts [3]. Behavioral detection — specifically, unauthenticated requests to `/agents` or `/chat` at anomalous rates or volumes — provides more durable signal.

## Strategic Considerations

CVE-2026-44338 is a specific vulnerability in a specific framework, but it exemplifies a structural risk pattern that security researchers have documented across the open-source agentic AI ecosystem: frameworks designed primarily for developer productivity have repeatedly shipped authentication as optional operator configuration rather than an enforced default, creating exposure when these frameworks migrate from development workstations to shared test environments and production infrastructure [3][4]. As this pattern recurs across agentic AI tooling, the risk is not isolated to any single framework. Organizations scaling agentic deployments without systematic AI asset inventory cannot remediate vulnerabilities they have not catalogued — a gap that security practitioners have increasingly flagged as agentic AI adoption accelerates.

The sub-four-hour exploitation window observed here has direct implications for vulnerability management program design. Organizations using agentic AI tooling should evaluate whether their patching SLAs distinguish AI infrastructure components as a category, whether their asset discovery processes surface agentic framework deployments reliably, and whether their patch deployment pipelines can operationalize response on a timeline measured in hours. A patch management program designed for a 30-day critical remediation window provides limited protection when exploitation begins in the afternoon of the same day as disclosure.

---

# CSA Resource Alignment

CVE-2026-44338 maps to threat models, control frameworks, and governance guidance that CSA has developed for the agentic AI security domain.

**MAESTRO** (Multi-Agent Environment, Security, Threat, Risk, & Outcome) is CSA's threat modeling framework for agentic AI systems [5]. This vulnerability is an instance of the risk class MAESTRO maps at the orchestration layer — a failure of explicit trust decisions at an agent system's external interface. MAESTRO establishes that each layer of a multi-agent system — from the model interface through the orchestration layer to external tool integrations — requires explicit trust decisions rather than inherited or assumed trust. The fail-open design at the center of this vulnerability contradicts that principle at the foundational level: the orchestration entry point makes no trust decision at all, granting implicit trust to any network caller.

**AARM** (Agentic AI Runtime Management) addresses the secure configuration and governance of agentic runtimes, including the access control requirements that govern who may invoke agent workflows [6]. CVE-2026-44338 illustrates the class of risk AARM is designed to prevent: a runtime component shipped with authentication disabled, exposing workflow execution to unauthenticated callers, and requiring explicit operator action to achieve a baseline security posture. AARM's controls for runtime API authentication, secure default configuration, and network exposure management are directly applicable to PraisonAI deployments and provide a control baseline for evaluating agentic frameworks broadly [6].

**The AI Controls Matrix (AICM)** provides a structured control framework for AI systems organized across the deployment lifecycle [7]. Relevant domains include access management controls governing authentication to AI service endpoints, configuration management controls requiring secure defaults at initial deployment, and monitoring and logging controls providing detection capability for unauthorized access. Organizations conducting AICM-based assessments should treat agentic orchestration API endpoints as in-scope assets for access control and configuration management controls, not as out-of-scope infrastructure components.

**The Agentic Trust Framework** establishes that trust in agentic AI systems must be explicitly conferred through verifiable identity and authorization rather than assumed based on network position or installation defaults [8]. CVE-2026-44338 is a case study in the consequences of the opposite design philosophy. The Agentic Trust Framework's zero trust approach — assume no caller is trusted until identity is verified and authorization is granted — applies directly to API surface design for orchestration frameworks. Security teams evaluating agentic AI tooling should assess whether each framework's default behavior embeds this principle or defers it to operator configuration, and should treat any framework that defaults to unauthenticated access as requiring compensating controls before deployment in shared or networked environments.

# References


[1] GitHub, Inc. "GHSA-6rmh-7xcm-cpxj: PraisonAI Missing Authentication in Legacy Flask API Server." GitHub Security Advisory, May 2026.

[2] NIST. "CVE-2026-44338 Detail." National Vulnerability Database, May 8, 2026.

[3] Clark, M. (Sysdig Threat Research). "CVE-2026-44338: PraisonAI Authentication Bypass in Under 4 Hours and the Growing Trend of Rapid Exploitation." Sysdig Blog, May 2026.

[4] SecurityWeek. "Hackers Targeted PraisonAI Vulnerability Hours After Disclosure." SecurityWeek, May 2026.

[5] Cloud Security Alliance. "Agentic AI Threat Modeling Framework: MAESTRO." CSA AI Safety Initiative, February 2025.

[6] Cloud Security Alliance. "AARM: Finding a Path to Secure the Agentic Runtime." CSA AI Safety Initiative, April 2026.

[7] Cloud Security Alliance. "AI Controls Matrix (AICM)." CSA Artifact, 2025.

[8] Cloud Security Alliance. "The Agentic Trust Framework: Zero Trust Governance for AI Agents." CSA AI Safety Initiative, February 2026.