INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-05-13 14:27 UTC

praison praisonai - praison praisonai Missing Authentication for Critical Function

CVE VULNERABILITY | **CRITICAL** | CVSS 9.8 | **CISA KEV**

SCC Item ID	SCC-CVE-2026-0165
Type	CVE Vulnerability
CVE ID	CVE-2026-44338
Severity	CRITICAL
CVSS Base Score	9.8
EPSS Score	0.0006 (17th percentile)
KEV Status	Yes — CISA Known Exploited Vulnerability
Affected Products	praison/praisonai versions 2.5.6 through before 4.6.34
Published	2026-05-12T00:00:00Z
Discovery Source	Vulncheck Kev

Executive Summary

A critical authentication bypass in PraisonAI, a multi-agent AI orchestration framework, allows any unauthenticated network caller to execute arbitrary AI agent workflows. Affected versions range from 2.5.6 through 4.6.33; the flaw is patched in 4.6.34 and has been added to the CISA Known Exploited Vulnerabilities catalog, indicating active exploitation. Organizations running PraisonAI in internet-accessible environments face immediate risk of unauthorized AI workflow execution, potential data exfiltration, and operational compromise.

Technical Analysis

CVE-2026-44338 (CVSS 9.8, CWE-306: Missing Authentication for Critical Function) affects PraisonAI versions 2.5.6 through 4.6.33. The legacy Flask API server ships with authentication disabled by default on two endpoints: GET/POST /agents (exposes workflow configuration) and POST /chat (triggers execution of the agents.yaml workflow). No token, credential, or session is required. An unauthenticated remote attacker with network access can enumerate configured agents and invoke arbitrary workflow execution directly. MITRE ATT&CK maps to T1190 (Exploit Public-Facing Application) for initial access and T1059 (Command and Scripting Interpreter) for subsequent execution via agent invocation. The issue is patched in version 4.6.34. CISA has added this CVE to the KEV catalog, confirming observed exploitation. EPSS score is 0.056% (17.32 percentile) as of initial publication, though KEV status supersedes EPSS as an exploitation signal. Primary

sources: NVD and CISA KEV (both T1 authorities). Note: Secondary sources in the source list contain references to unrelated CVE identifiers and should be disregarded for this vulnerability.

Action Checklist

1. Step 1: Containment. Isolate any internet-facing PraisonAI instances running versions 2.5.6 through 4.6.33. Block inbound access to the Flask API server ports (default 8080 or custom port) at the network perimeter until patching is complete. If isolation is not immediately possible, apply a WAF rule blocking unauthenticated POST requests to /chat and GET/POST requests to /agents.
2. Step 2: Detection. Query web server and application logs for unauthenticated requests to /agents and /chat endpoints across all PraisonAI instances. Look for requests lacking Authorization headers or valid session tokens. Review agents.yaml for unauthorized modifications. Check for anomalous outbound connections initiated by agent workflows, which may indicate post-exploitation activity. If SIEM is available, alert on T1190 and T1059 indicators correlated to PraisonAI process activity.
3. Step 3: Eradication. Upgrade all PraisonAI deployments to version 4.6.34 or later, which patches the missing authentication control. After upgrading, verify the Flask API server enforces authentication on /agents and /chat endpoints. Review and rotate any secrets, API keys, or credentials accessible to agents.yaml workflows, as these may have been exposed during the exploitation window.
4. Step 4: Recovery. After patching, validate that /agents and /chat endpoints return 401 or 403 for unauthenticated requests. Audit agents.yaml for unauthorized changes. Review workflow execution logs for the full exposure window (earliest known exploitation date through remediation) to identify any unauthorized workflow runs. Monitor outbound network traffic from PraisonAI hosts for residual suspicious activity for at least 72 hours post-patch.
5. Step 5: Post-Incident. Document the exposure window and any unauthorized workflow executions for incident records. Assess whether AI agent workflows had access to sensitive data stores, APIs, or credentials that require rotation or audit. Evaluate whether default-insecure configurations in AI orchestration tools are covered by your software deployment policy and hardening baseline. Consider adding authentication enforcement verification to your deployment checklist for all AI framework components.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate to senior IR leadership, legal, and privacy counsel immediately if workflow execution logs confirm unauthorized agent runs that accessed internal APIs, data stores, credentials, or any system containing PII, PHI, or regulated data — CVSS 9.8, active CISA KEV listing, and unauthenticated network exploitability with no preconditions collectively indicate maximum breach probability requiring breach notification assessment under applicable regulations (HIPAA, GDPR, state privacy laws).

Recovery Notes	After applying PraisonAI 4.6.34, verify authentication enforcement on both /agents and /chat endpoints from an unauthenticated client before returning the service to production — any 200 response indicates the patch did not apply correctly. Rotate all API keys, tokens, and credentials referenced in agents.yaml regardless of whether exploitation is confirmed, as the unauthenticated attack surface provided full read access to workflow configurations during the exposure window. Maintain 72-hour enhanced outbound network monitoring on all PraisonAI hosts post-patch, since agent workflows can establish persistent callbacks or exfiltration channels that survive the authentication fix.
Forensic Artifacts	Flask/gunicorn access logs showing POST /chat and GET/POST /agents requests without Authorization headers — HTTP 200 responses to these endpoints during the exposure window confirm successful unauthenticated workflow execution (path: /var/log/praisonai/access.log or configured gunicorn log path) | agents.yaml file with modification timestamps and diff against baseline — unauthorized workflow definitions or injected tool configurations in this file indicate attacker persistence or capability expansion beyond the initial authentication bypass | Process execution records (auditd execve events or Sysmon Event ID 1) showing the PraisonAI Python process spawning child processes such as sh, bash, curl, wget, or python subprocesses — this artifact is specific to T1059 (Command and Scripting Interpreter) execution via injected agent workflows | Outbound network connections from the PraisonAI host to external IPs or domains during the exposure window — agent workflows executing attacker-supplied tasks can initiate exfiltration, C2 callbacks, or lateral movement; capture via pcap, NetFlow, or firewall egress logs filtered on the PraisonAI host source IP | PraisonAI application-level workflow execution logs recording submitted agent task payloads and outputs — if PraisonAI logs workflow inputs and outputs (check configured log level and output directory), these entries directly evidence what instructions an attacker submitted and what data the agents returned

Per-Action IR Details

Step 1: Containment — Immediately isolate any internet-facing PraisonAI instances running versions 2.5.6 through 4.6.33. Block inbound access to the Flask API server ports (default 8080 or configured equivalent) at the network perimeter or firewall until patching is complete. If isolation is not immediately possible, apply a WAF rule blocking unauthenticated POST requests to /chat and GET/POST requests to /agents.

NIST Phase: Containment

Reference: NIST 800-61r3 §3.3 — Containment Strategy

Controls: NIST IR-4 (Incident Handling), NIST SC-7 (Boundary Protection), CIS 4.4 (Implement and Manage a Firewall on Servers), CIS 4.5 (Implement and Manage a Firewall on End-User Devices)

Compensating: On Linux hosts, immediately execute: `sudo iptables -I INPUT -p tcp --dport 8080 -j DROP` (adjust port if PraisonAI is configured on a non-default port — check the PraisonAI config file or running process with `ss -tlnp | grep python`). On Windows, use: `netsh advfirewall firewall add rule name='Block PraisonAI' dir=in action=block protocol=TCP localport=8080`. If a WAF is not available, deploy NGINX as a reverse proxy with a minimal config block that requires HTTP Basic Auth for /agents and /chat paths as an immediate stopgap. A 2-person team can accomplish the iptables/netsh block in under 5 minutes per host.

Evidence: Before isolating, capture a full netstat/ss snapshot to document active connections to port 8080: `ss -tnp sport = :8080 > /tmp/praisonai_active_conns_$(date +%Y%m%d%H%M%S).txt`. Preserve the current agents.yaml file with hash: `sha256sum /path/to/agents.yaml`. Dump Flask process memory if feasible using `procdump` (Linux: `gcore `) to capture in-memory workflow state that may reflect what an attacker submitted. Record the process tree: `ps auxf | grep praisonai`.

Step 2: Detection — Query web server and application logs for unauthenticated requests to /agents and /chat endpoints across all PraisonAI instances. Look for requests lacking Authorization headers or session tokens.

Review agents.yaml for unauthorized modifications. Check for anomalous outbound connections initiated by agent workflows, which may indicate post-exploitation activity. If SIEM is available, alert on T1190 and T1059 indicators correlated to PraisonAI process activity.

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2 — Detection and Analysis

Controls: NIST AU-6 (Audit Record Review, Analysis, And Reporting), NIST AU-3 (Content Of Audit Records), NIST SI-4 (System Monitoring), CIS 8.2 (Collect Audit Logs)

Compensating: Run the following grep against Flask/gunicorn access logs to surface unauthenticated hits on exploitable endpoints: `grep -E '(POST /chat|GET /agents|POST /agents)' /var/log/praisonai/access.log | grep -v 'Authorization'`. If PraisonAI is behind NGINX, query: `awk '$7 ~ /^\/agents|^\/chat/ && $9 == 200' /var/log/nginx/access.log` — HTTP 200 responses to these endpoints without auth indicate successful exploitation. For outbound connection anomalies, capture with: `tcpdump -i eth0 -nn -w /tmp/praisonai_outbound.pcap 'src host '` and parse with Wireshark for unexpected DNS, HTTP/S, or shell traffic. Use Sigma rule for MITRE T1190 (Exploit Public-Facing Application) mapped to Flask process spawning child processes: look for Python spawning `sh`, `bash`, or `curl` as child processes via `ausearch -m execve` or Sysmon Event ID 1 filtered on parent process matching the PraisonAI Python PID.

Evidence: Preserve complete Flask/gunicorn access logs covering the full exposure window (first deployment of versions 2.5.6–4.6.33 through detection): `cp /var/log/praisonai/access.log /evidence/praisonai_access_$(date +%Y%m%d).log && sha256sum /evidence/praisonai_access_*.log`. Capture agents.yaml modification timestamps and diff against your last known-good backup: `stat agents.yaml; git log -- agents.yaml` (if version-controlled). Export outbound network flow data from the PraisonAI host covering the exposure window — specifically any connections to external IPs initiated by the Python process. If auditd is enabled, collect: `ausearch -m execve --start | grep -A5 python`.

Step 3: Eradication — Upgrade all PraisonAI deployments to version 4.6.34 or later, which patches the missing authentication control. After upgrading, verify the Flask API server enforces authentication on /agents and /chat endpoints. Review and rotate any secrets, API keys, or credentials accessible to agents.yaml workflows, as these may have been exposed during the exploitation window.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST SI-2 (Flaw Remediation), NIST IA-5 (Authenticator Management), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management), CIS 5.2 (Use Unique Passwords)

Compensating: Upgrade via pip in the PraisonAI virtualenv: `pip install --upgrade praisonai==4.6.34` then verify: `pip show praisonai | grep Version`. After upgrade, perform an immediate authentication enforcement test using curl from an external host (or a host outside the trust boundary): `curl -i -X POST http://:8080/chat -H 'Content-Type: application/json' -d '{"message":"test"}'` — the response MUST be HTTP 401 or 403; any 200 response indicates patching has not resolved the authentication bypass. Rotate all secrets in agents.yaml by scanning for API key patterns: `grep -E '(api_key|secret|token|password)' agents.yaml` and regenerate each credential at its originating service. Use `git secret` or `ansible-vault` to prevent plaintext secrets in agents.yaml going forward.

Evidence: Before running the upgrade, preserve a forensic copy of the installed PraisonAI package files: `pip show -f praisonai > /evidence/praisonai_pkg_manifest_pre_patch.txt`. Capture a before-and-after diff of the authentication middleware code — specifically the Flask route decorators for /agents and /chat — to confirm the fix is applied: `python -c "import praisonai; import inspect; print(inspect.getfile(praisonai))"` then diff the route handler source. Document all credential rotation actions with timestamps for the incident record, as any API keys held in agents.yaml during the exposure window must be treated as compromised.

Step 4: Recovery — After patching, validate that /agents and /chat endpoints return 401 or 403 for unauthenticated requests. Audit agents.yaml for unauthorized changes. Review workflow execution logs for the full exposure window (earliest known exploitation date through remediation) to identify any unauthorized workflow runs. Monitor outbound network traffic from PraisonAI hosts for residual suspicious activity for at

least 72 hours post-patch.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST IR-4 (Incident Handling), NIST AU-6 (Audit Record Review, Analysis, And Reporting), NIST SI-7 (Software, Firmware, And Information Integrity), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: Run a structured validation script post-patch: `for endpoint in /agents /chat; do echo "Testing $endpoint:"; curl -s -o /dev/null -w "%{http_code}" -X POST http://:8080$endpoint; echo; done` — both must return 401 or 403. For agents.yaml integrity, compute a SHA-256 baseline immediately after patching and compare hourly for 72 hours: `watch -n 3600 'sha256sum agents.yaml'`. For outbound monitoring without a SIEM, run Wireshark/tshark in capture mode on the PraisonAI host for 72 hours: `tshark -i eth0 -f 'src host and not dst net ' -w /tmp/praisonai_recovery_monitor.pcap` and review for unexpected destinations. If osquery is available, use the `process_open_sockets` table filtered on the PraisonAI Python PID to catch any lingering agent-initiated outbound connections.

Evidence: Reconstruct a complete timeline of workflow executions during the exposure window from PraisonAI application logs — document each workflow name, input payload, and output if logged, as these represent the blast radius of any unauthorized agent runs. Preserve network flow records (NetFlow, pcap, or firewall logs) from the exposure window covering outbound traffic from PraisonAI hosts, since agent workflows can be weaponized to exfiltrate data or pivot to internal APIs. Capture the post-patch authentication test results (curl output with HTTP response codes) as documentary evidence of remediation effectiveness.

Step 5: Post-Incident — Document the exposure window and any unauthorized workflow executions for incident records. Assess whether AI agent workflows had access to sensitive data stores, APIs, or credentials that require rotation or audit. Evaluate whether default-insecure configurations in AI orchestration tools are covered by your software deployment policy and hardening baseline. Consider adding authentication enforcement verification to your deployment checklist for all AI framework components.

NIST Phase: Post Incident

Reference: NIST 800-61r3 §4 — Post-Incident Activity

Controls: NIST IR-4 (Incident Handling), NIST IR-8 (Incident Response Plan), NIST SI-2 (Flaw Remediation), NIST CM-6 (Configuration Settings), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process), CIS 2.2 (Ensure Authorized Software is Currently Supported)

Compensating: Build a PraisonAI-specific hardening checklist item: before any AI orchestration framework is deployed or updated, verify authentication is enforced on all API endpoints using the curl test from Step 4 — codify this as a CI/CD pipeline gate if the team uses GitHub Actions or GitLab CI. For the lessons-learned record, enumerate every external API, data store, and credential referenced in agents.yaml workflows that was in scope during the exposure window — this list drives the downstream credential rotation and data breach assessment. Subscribe to PraisonAI's GitHub releases (https://github.com/MervinPraison/PraisonAI/releases) for future security advisories and add the project to your vulnerability management feed via OSV or GitHub Dependabot alerts.

Evidence: Compile the final incident record with: (1) full exposure window (first affected version deployed through patch applied), (2) list of all workflow executions found in logs during the window with payloads and outputs, (3) inventory of all credentials and API keys present in agents.yaml during exposure, (4) outbound connection summary from the 72-hour post-patch monitoring period, and (5) authentication validation test results confirming remediation. This record supports any regulatory breach notification assessment if agent workflows accessed PII, PHI, or regulated data stores.

Detection Guidance

Query application and web server logs for requests to /agents and /chat endpoints that lack Authorization headers. In environments logging HTTP request headers, filter for requests to these paths where no token or session cookie is present. Example log pattern (nginx/Apache combined log format): look for POST /chat or

GET /agents with HTTP 200 responses and no Authorization header. In SIEM environments, correlate PraisonAI process execution spikes with inbound unauthenticated API calls. For host-based detection, monitor for unexpected child processes spawned by the PraisonAI Flask process, consistent with T1059 execution via agent workflow invocation. If agents.yaml is stored on disk, alert on unexpected file modification events. No public IOC hashes or IP indicators are confirmed at this time; behavioral detection is the primary signal.

Indicators of Compromise

Type	Value	Context	Confidence
URL	`/agents`	Unauthenticated access to this PraisonAI Flask API endpoint exposes workflow configuration; requests without Authorization headers are suspicious	HIGH
URL	`/chat`	Unauthenticated POST to this PraisonAI Flask API endpoint triggers agents.yaml workflow execution; unauthenticated requests are a primary exploitation indicator	HIGH

Framework Mappings

MITRE-ATTACK

- **T1190** — Exploit Public-Facing Application
- **T1059** — Command and Scripting Interpreter

NIST-800-53R5

- **CA-8** — Penetration Testing
- **RA-5** — Vulnerability Monitoring and Scanning
- **SC-7** — Boundary Protection
- **SI-2** — Flaw Remediation
- **SI-7** — Software, Firmware, and Information Integrity
- **CM-7** — Least Functionality
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **IA-2** — Identification and Authentication (Organizational Users)

OWASP-TOP10-2021

- **A07:2021** — Identification and Authentication Failures

CIS-V8

- **6.3** — Require MFA for Externally-Exposed Applications
- **7.3** — Perform Automated Operating System Patch Management
- **7.4** — Perform Automated Application Patch Management

ISO-27001-2022

- **A.8.8** — Management of technical vulnerabilities

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1190	Exploit Public-Facing Application	Initial-Access
T1059	Command and Scripting Interpreter	Execution

Sources

Source	URL	Tier
vulncheck_kev	https://nvd.nist.gov/vuln/detail/CVE-2026-44338	T1
PraisonAI Authentication Bypass Vulnerability CVE-2026-44338	https://www.linkedin.com/posts/dfir-lab_threatintel-dfir-cybersecur...	T3
CVE-2026-4438: GNU C Library Information Disclosure Flaw	https://www.sentinelone.com/vulnerability-database/cve-2026-4438/	T3
CVE-2026-4438 - Red Hat Customer Portal	https://access.redhat.com/security/cve/cve-2026-4438	T3
CVE-2026-4433: Tenable OT Information Disclosure Flaw	https://www.sentinelone.com/vulnerability-database/cve-2026-4433/	T3
CISA KEV	https://www.cisa.gov/known-exploited-vulnerabilities-catalog	T1

Generated 2026-05-13 14:27 UTC by TJS Security Command Center