CSAI Foundation

Cloud Security Alliance AI Safety Initiative

Marimo Pre-Auth RCE CVE-2026-39987: Exploited in Hours

Terminal WebSocket Authentication Bypass Grants Unauthenticated Shell Access to AI Notebook Servers

Unofficial AI-assisted Research

2026-04-11

Key Takeaways

- CVE-2026-39987 (GHSA-2679-6mx9-h9xc) is a pre-authentication remote code execution vulnerability in Marimo, an open-source AI-native reactive Python notebook, with a CVSS 4.0 score of 9.3; all versions before 0.23.0 are affected, and the issue is resolved beginning with version 0.23.0 [1][2][3].
- The terminal WebSocket endpoint `/terminal/ws` was deployed without authentication enforcement: while Marimo's other WebSocket endpoints correctly invoke `validate_auth()`, this endpoint checked only runtime mode and platform support before accepting connections, granting any unauthenticated caller a full interactive PTY shell as the user running the Marimo process [1][4].
- The Sysdig Threat Research Team detected active in-the-wild exploitation within 9 hours and 41 minutes of the vulnerability advisory's publication on April 8, 2026; no public proof-of-concept code existed at the time, indicating the attacker reverse-engineered a working exploit directly from the advisory description [5].
- The observed attack on a Sysdig honeypot executed a complete credential-theft operation in under three minutes, targeting environment variable files (`.env`), SSH keys, and file system content, demonstrating that the post-exploitation window for sensitive data exposure is extremely narrow [5].
- CISA has added CVE-2026-39987 to its Known Exploited Vulnerabilities catalog with a federal remediation deadline of April 11, 2026, reflecting the confirmed severity and exploitation status of the vulnerability [6].
- The speed of exploitation against Marimo — a niche platform with approximately 20,000 GitHub stars compared to the 100,000+ star counts of higher-profile AI tooling — indicates that threat actors are conducting broad surveillance of vulnerability advisory feeds rather than restricting attention to widely-deployed platforms; Sysdig threat researchers assess that this rapid weaponization likely involved AI assistance in exploit development [5].

Background

Marimo is an open-source, AI-native reactive Python notebook designed to address longstanding limitations of traditional Jupyter-style environments. Unlike conventional Python notebooks, Marimo represents notebook logic as a directed acyclic graph: when a cell changes, Marimo automatically propagates the effects to all dependent cells, enforcing reproducibility and eliminating the hidden state that makes Jupyter notebooks difficult to re-execute reliably [7][8]. The platform stores notebooks as pure Python files, making them compatible with standard version control workflows, and can execute them as scripts or deploy them as interactive web applications. Marimo has accumulated approximately 20,000 GitHub stars and has attracted an active developer community building data science workflows, interactive visualizations, and AI-assisted research environments [5].

Marimo's recent releases have added features that make it accessible to AI agent workflows, including a `--mcp` flag that converts any notebook into a Model Context Protocol (MCP) server, enabling AI systems to inspect, query, and reason about notebook state as a structured tool interface [8]. The separately released Marimo Pair feature allows AI agents to use the notebook environment as a reactive REPL for collaborative computational research, providing agents with a persistent, stateful Python environment with guaranteed reproducibility. These capabilities position Marimo as infrastructure in AI development pipelines, not merely a developer convenience — making the security properties of its server-side implementation directly relevant to organizations deploying AI systems.

Marimo operates as a local web server accessible over HTTP and WebSocket connections. Its architecture uses Starlette as the underlying web framework and defines multiple WebSocket endpoints for different classes of client interaction, including notebook execution management and, on supported platforms, an integrated terminal that allows users to run shell commands directly from the notebook interface. This terminal integration, while useful for development workflows, introduced the attack surface that CVE-2026-39987 exploits.

The vulnerability was disclosed on April 8, 2026, alongside the release of Marimo version 0.23.0, which contains the fix [3][9]. The advisory was published to the GitHub Security Advisory database as GHSA-2679-6mx9-h9xc and assigned CVE-2026-39987 with a CVSS 4.0 base score of 9.3, reflecting the combination of no authentication requirement, network accessibility, and full code execution impact [4][9].

Security Analysis

Vulnerability Mechanics

The root cause of CVE-2026-39987 is an inconsistency in authentication enforcement across Marimo's WebSocket endpoint implementations. Marimo employs Starlette's `AuthenticationMiddleware` to handle authentication state across the application. An important characteristic of this middleware is that it does not reject unauthenticated connections outright; instead, it attaches an `UnauthenticatedUser` marker to requests that fail authentication, leaving actual access enforcement to be implemented at the individual endpoint level through `@requires()` decorators or explicit `validate_auth()` calls [1][4].

The primary WebSocket endpoint, `/ws`, correctly calls `validate_auth()` during its connection handling, rejecting unauthenticated connections before establishing a session. The terminal endpoint, `/terminal/ws`, does not. Upon receiving a WebSocket connection, `/terminal/ws` verifies only that the server is running in the appropriate mode and that the host platform supports terminal functionality. Having satisfied those two checks, it proceeds to allocate a full pseudo-terminal (PTY) process and begin relaying input and output between the client and the shell — with no check of the caller's authentication state [1][4]. The result is that any attacker who can send a WebSocket handshake to a Marimo instance obtains an interactive PTY shell as the OS user running the Marimo process, with all privileges that user holds on the host system, regardless of whether Marimo authentication is enabled or what credentials that configuration requires.

The flaw is a classic case of incomplete security control implementation: the mechanism was in place, but its application was not consistent. The attack requires no credential guessing, no session token theft, and no prior knowledge of the target environment. A single network-reachable Marimo instance is sufficient, and the only precondition is that the host platform support terminal functionality (which encompasses all major operating systems in typical deployment configurations).

Exploitation Pattern and Timeline

The Sysdig Threat Research Team's honeypot observation provides detailed insight into attacker behavior following disclosure [5]. The first exploitation attempt arrived within 9 hours and 41 minutes of the advisory's publication, before any public proof-of-concept code had been released. The attacker's ability to construct a working exploit from the advisory description alone suggests either significant

technical capability or, as Sysdig researchers assess, AI-assisted exploit development — a capability that enables rapid weaponization of vulnerability disclosures that include sufficient technical detail about the affected code path.

The observed attack followed a tightly sequenced playbook. After establishing the unauthenticated terminal connection, the threat actor conducted rapid file system reconnaissance, moving within minutes to targeted credential harvesting. The attacker attempted to read `.env` files, which in development and AI infrastructure deployments commonly contain API keys for cloud services, LLM providers, database credentials, and application secrets. The attacker also enumerated and attempted to read SSH private key files, which would enable lateral movement to other systems in the environment. This full credential-theft operation — from initial WebSocket connection through active exfiltration attempts — was executed in under three minutes [5]. The speed and focus of the post-exploitation behavior suggests either prior familiarity with Marimo's deployment patterns or a playbook adapted from similar attacks on Python-based development tools.

The particular significance of this exploitation timeline extends beyond its speed. Marimo is not a widely-known enterprise platform. With approximately 20,000 GitHub stars, it is a fraction of the prominence of platforms like Langflow (145,000+ stars) or n8n (75,000+ stars) [5], both of which have experienced their own rapid post-disclosure exploitation in 2026 [10]. The fact that Marimo attracted exploitation at comparable speed despite its lower profile indicates that threat actors are not filtering vulnerability advisories by platform popularity or organizational adoption. Sysdig researchers assess that AI-assisted exploit development may be enabling this accelerated weaponization timeline [5], a hypothesis consistent with the absence of any prior public proof-of-concept at the time of exploitation — suggesting that AI development tooling with network-accessible vulnerabilities now faces an exploitation window measured in hours rather than days, regardless of platform prominence.

AI Development Tooling as an Elevated-Risk Target Class

The nature of Marimo's deployment context amplifies the impact of this vulnerability well beyond what a comparable RCE in a generic web application would represent. Marimo instances used for data science and AI development commonly operate with credentials and data artifacts that are concentrated in ways that may not be reflected in the organization's standard asset risk classification. Environment variable files accessible from a compromised Marimo terminal may contain API keys for commercial LLM providers, enabling an attacker to conduct inference at organizational cost, extract training data, or manipulate retrieval-augmented generation pipelines. SSH keys present in the development environment facilitate lateral movement to compute resources, model training infrastructure, and connected cloud environments.

Marimo's integration with MCP and its positioning as an agent-accessible computational environment adds a further dimension to the risk. An organization using Marimo as part of an AI agent workflow may expose the Marimo server to agent-generated traffic, increasing the effective exposure of the `/terminal/ws` endpoint. If the Marimo instance is reachable by both human developers and automated AI agents, the attack surface for CVE-2026-39987 is commensurately broader. Organizations that have adopted Marimo Pair or similar agentic integration patterns should treat this vulnerability as directly relevant to their AI pipeline security posture, not merely as a developer tool issue.

The recurring pattern of rapid exploitation against AI development platforms in 2026 — spanning Flowise, Langflow, and now Marimo — establishes a trend that warrants a structural response rather than incremental patch management. Each of these platforms occupies a similar position in AI development pipelines: they are accessible over the network, they hold credentials for critical AI infrastructure, they are often deployed by individual developers or small teams without enterprise-grade security oversight, and they offer attackers a direct path to the data and services powering AI applications. The cluster of exploitation activity against this category of software suggests that AI development tooling has become a recognized and actively targeted class in offensive threat actors' operational playbooks.

Recommendations

Immediate Actions

Organizations running any version of Marimo before 0.23.0 should treat this as an emergency requiring immediate action. The remediation is an upgrade to Marimo version 0.23.0 or later [2][3]. For typical installations, the upgrade can be performed via pip (`pip install --upgrade marimo`); containerized or production-pinned deployments may require compatibility review before updating. The upgrade should be executed across all environments where Marimo is deployed, including local developer workstations, shared development servers, containerized deployments, and any cloud-hosted instances.

Any Marimo instance that was network-accessible prior to patching should be treated as potentially compromised. Incident response activities should include a review of WebSocket connection logs for requests to `/terminal/ws` from unexpected source addresses, inspection of shell command history on the host system for the account running Marimo, and audit of file access logs for reads of credential-bearing files including `.env` files and SSH key directories. Rotate all credentials that may have been accessible from the Marimo server, including API keys for LLM providers, database connection strings,

cloud service credentials, and any secrets stored in environment variables or configuration files reachable from the Marimo working directory. If SSH keys were accessible, treat any system reachable by those keys as potentially compromised and conduct host-level forensics accordingly.

If an immediate upgrade is operationally infeasible, the interim containment measure is to remove all public network exposure of the Marimo instance and restrict access to trusted internal IP ranges via firewall rules or network access control lists. Blocking the `/terminal/ws` path at a reverse proxy layer provides additional defense in depth until the upgrade is completed. Do not assume that enabling Marimo's authentication configuration is sufficient mitigation: the vulnerability bypasses authentication enforcement entirely, and authentication state does not protect the affected endpoint in vulnerable versions.

Short-Term Mitigations

Beyond emergency patching, organizations should establish Marimo and similar AI development tools within their formal vulnerability management programs. AI notebook platforms and agent builder tools are commonly deployed as development-category software, which can exempt them from the asset inventory and patch management SLAs applied to production infrastructure. This categorization is no longer appropriate: tools like Marimo now serve as active components of production AI pipelines, hold production-grade credentials, and face the same adversarial attention as production-facing systems. Marimo instances should be enrolled in authenticated vulnerability scanning, included in software composition analysis pipelines, and subject to patch SLA targets consistent with the CVSS severity of identified CVEs.

Network architecture for Marimo deployments should reflect the principle of least-access exposure. Marimo instances intended for local or small-team development use should run on loopback addresses (127.0.0.1) rather than being bound to all interfaces (0.0.0.0). Where Marimo is used in shared environments or deployed on servers, access should be gated behind authentication proxies, VPN boundaries, or network segmentation that restricts connectivity to authorized users. The integrated terminal feature, given its demonstrated attack surface, should be disabled in environments where it is not actively required using the `--no-terminal` server flag, reducing the exposed endpoint footprint.

Credential hygiene practices for the AI development environment should be reviewed as a direct consequence of this vulnerability. API keys, cloud credentials, and database connection strings used in Marimo notebooks should be scoped to the minimum necessary permissions, stored in secrets management systems rather than plaintext environment files where possible, and rotated on a regular

schedule. Monitoring for anomalous usage of AI infrastructure credentials — unexpected API call volumes, off-hours access, or calls from unexpected source IP addresses — provides a detection capability that can surface credential theft even when endpoint-level exploitation goes unobserved.

Strategic Considerations

CVE-2026-39987 illustrates a structural pattern that organizations investing in AI development infrastructure should internalize as a planning assumption rather than a surprise: AI-native development tools are being targeted by sophisticated threat actors at a pace and scale that reflects their value as a gateway to AI infrastructure. Three of the most significant AI developer tool exploitations in early 2026 — affecting Flowise, Langflow, and Marimo — share a common profile: open-source platforms, network-accessible APIs, missing or incomplete authentication enforcement, and deployment contexts rich with credentials for connected AI and cloud services. The speed from disclosure to exploitation across these incidents, now measured in hours rather than days, compresses the remediation window to a point where reactive patch management is no longer an adequate sole control.

Organizations should evaluate AI development tooling against an explicit security maturity baseline that includes: a demonstrated history of responsible disclosure handling and rapid patching, the presence of security engineering resources in the development team, adoption of automated security testing in the CI/CD pipeline, and documented security architecture for network-accessible components. Platforms that cannot demonstrate these characteristics should be isolated to environments where the blast radius of a compromise is bounded by compensating network and credential controls, rather than deployed in positions of broad trust within AI development pipelines.

The indication that attackers are using AI assistance to accelerate exploit development from vulnerability advisories deserves particular strategic attention. If the effective window between disclosure and weaponization has collapsed to under ten hours — even for niche platforms — then the security model that relies on disclosure-to-patch lead time as a defensive buffer requires revision. Organizations should invest in pre-emptive vulnerability scanning that identifies affected software before disclosures land, and should establish processes that treat critical-severity AI tooling CVEs as P0 incidents with same-day response expectations rather than standard patch cycle targets.

CSA Resource Alignment

CVE-2026-39987 maps to several threat categories and control domains within the CSA AI Controls Matrix (AICM), CSA's MAESTRO framework, and related AI security guidance.

Within the AICM's AI Supply Chain Security domain, this vulnerability illustrates the risk posed by open-source AI tooling with incomplete security controls in organizations' AI development pipelines. The AICM's guidance on AI Application Provider responsibilities addresses the need for organizations deploying AI platforms to assess the security maturity of those platforms before integrating them into workflows that handle sensitive data or hold privileged credentials. The pattern of missing authentication enforcement on a privileged endpoint aligns with the AICM's access control guidance, which addresses the expectation that network-accessible AI application components enforce authentication independent of the middleware configuration inherited from the underlying framework.

MAESTRO's seven-layer agentic AI threat model is relevant in two distinct ways for this CVE. At the integration and orchestration layer, the Marimo vulnerability demonstrates the risk introduced when AI development platforms serve as both human-accessible development environments and agent-accessible computational resources: the same WebSocket endpoint that a human developer uses for terminal access is reachable by any network-connected entity, including AI agents that have been granted access to the Marimo instance. MAESTRO's threat model addresses this class of risk in AI agent infrastructure, treating missing authentication controls on agent-accessible APIs as requiring compensating controls at the network layer when application-layer controls are absent or incomplete.

CSA's Zero Trust guidance is directly applicable to the deployment architecture recommended for Marimo and similar AI development tools. The Zero Trust principle of least-privilege network access — that no component should be network-reachable unless there is an explicit, validated authorization for that connectivity — would have prevented the opportunistic exploitation observed in the Sysdig honeypot by confining Marimo access to authenticated, authorized network paths. Organizations implementing Zero Trust architectures for AI development infrastructure should enforce identity-verified access controls at the network boundary for all AI tooling regardless of whether the application layer implements its own authentication.

References

[1] GitHub Advisory Database. "GHSA-2679-6mx9-h9xc: Marimo: Pre-Auth Remote Code Execution via Terminal WebSocket Authentication Bypass." GitHub Security Advisory Database, April 2026.

[2] The Hacker News. "Marimo RCE Flaw CVE-2026-39987 Exploited Within 10 Hours of Disclosure." The Hacker News, April 10, 2026.

[3] GitHub. "Releases · marimo-team/marimo." GitHub, April 2026.

[4] Endor Labs. "Root in One Request: Marimo's Critical Pre-Auth RCE (CVE-2026-39987)." Endor Labs Blog, April 2026.

[5] Sysdig Threat Research Team. "Marimo OSS Python Notebook RCE: From Disclosure to Exploitation in Under 10 Hours." Sysdig, April 2026.

[6] CISA. "Known Exploited Vulnerabilities Catalog." Cybersecurity and Infrastructure Security Agency, accessed April 11, 2026.

[7] marimo-team. "GitHub – marimo-team/marimo: A reactive notebook for Python." GitHub, accessed April 11, 2026.

[8] Marimo. "marimo – a next-generation Python notebook." marimo.io, accessed April 11, 2026.

[9] NIST. "NVD – CVE-2026-39987." National Vulnerability Database, April 2026.

[10] The Hacker News. "Critical Langflow Flaw CVE-2026-33017 Triggers Attacks within 20 Hours of Disclosure." The Hacker News, March 2026.

