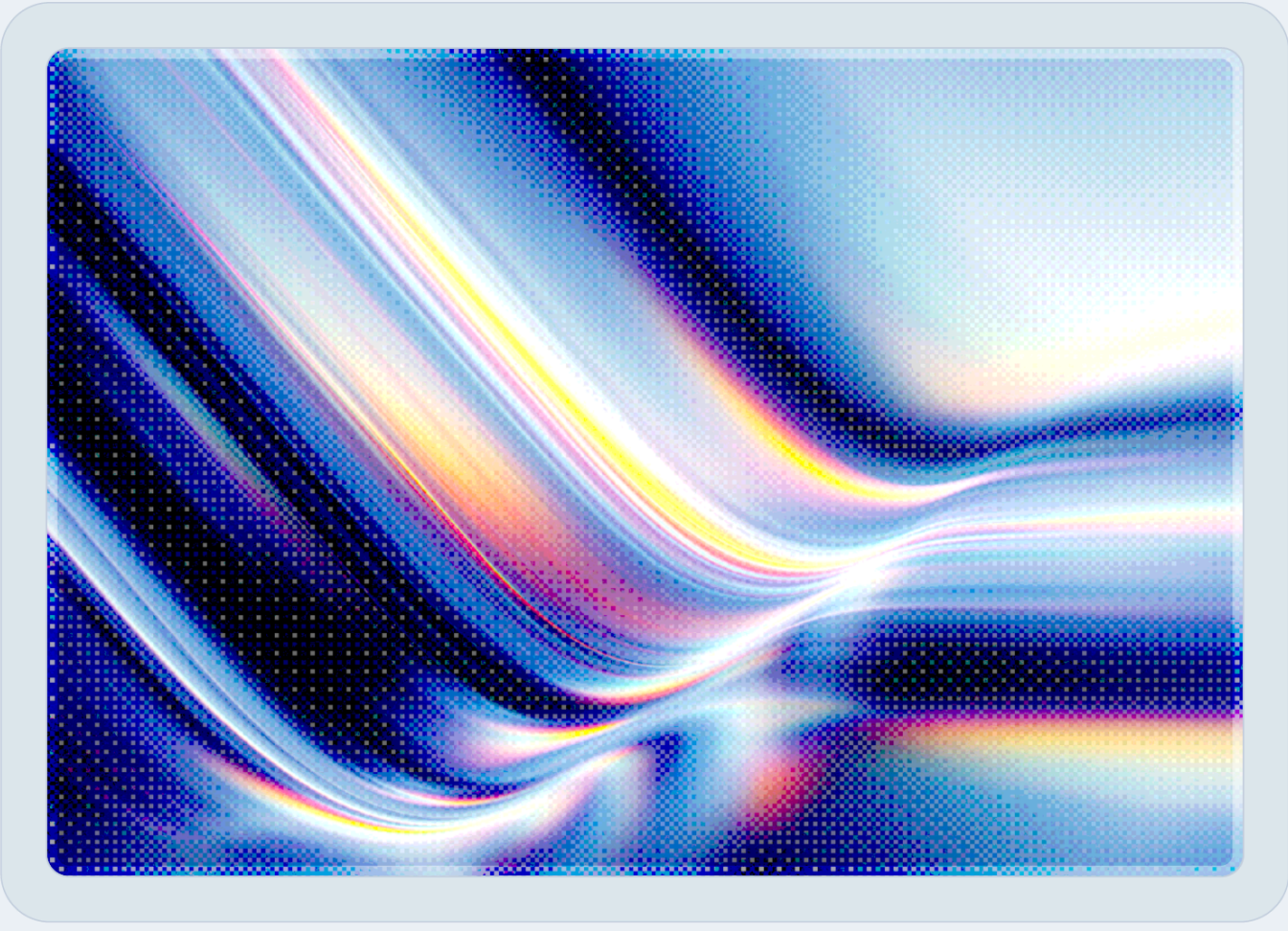
CSAI Foundation | Cloud Security Alliance

MONAI Label Zero-Day: AI Agents Exploit Medical Imaging Research Servers

Security implications of an autonomously discovered path traversal flaw in medical imaging annotation servers

2026-11-21

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

On November 20, 2026, Amazon Threat Intelligence reported that a threat actor used autonomous AI agents running the open-source Cairn framework to find and exploit a previously unknown vulnerability in MONAI Label, an open-source server for AI-assisted annotation of medical images. The agents then used the flaw to obtain a root shell on a university server [1][2][3]. The confirmed victims were university research servers, and none of the public reporting indicates that clinical systems or patient data were affected. According to Amazon, the agents took roughly 11 minutes to go from mapping the application's API to writing arbitrary files on the server [1][2]. Amazon assessed "with high confidence" that the flaw was a zero-day, and attributed the activity to a Chinese-nexus actor with moderate confidence [1][2].

The vulnerability is tracked as GitHub advisory GHSA-qc22-rmgc-vg94. It is a critical-severity path traversal flaw (CWE-22) in the `PUT /datastore/image` and `PUT /datastore/label` endpoints, and it affects `monailabel` versions 0.8.5 and earlier [4]. Authentication is disabled by default, so any client that can reach a default deployment can exploit it without credentials. With authentication enabled, the low-privilege `ANNOTATOR` role is still enough [4].

A fix was merged into the source tree on October 19, 2026, but as of this writing no patched release exists. Version 0.8.5, published to PyPI in November 2024, is still the newest installable package, and no CVE identifier has been assigned [4][5][6][7]. Organizations running MONAI Label should treat every reachable instance as exposed. They should apply the advisory's workarounds now and look for signs of earlier compromise, because turning on authentication after the fact does not remove an attacker who is already present [4][7].

The incident matters for more than the single flaw. It is among the first publicly documented cases in which offensive AI agents, working toward a goal set by a threat actor, found a novel vulnerability during a live intrusion and turned it into a working exploit [1][3]. CSA documented a related case in October 2026, in which an AI agent found and chained zero-days during a live breach of DIVD [15]. The MONAI Label case also suggests that research-grade medical AI tooling, which in CSA's assessment is frequently deployed by small academic teams with default settings, is now within reach of automated reconnaissance and exploitation at machine speed.

Background

MONAI Label is part of Project MONAI, an open-source initiative for deep learning in medical imaging, but it is a separate program from the core MONAI framework [8]. Research and clinical teams run it as a server to label radiology scans, train segmentation models, and serve those models to viewers such as 3D Slicer and OHIF. It can store images in a local datastore or connect to a DICOMweb server as its image source [8]. In CSA's assessment, this means MONAI Label servers are likely to sit next to sensitive imaging datasets and GPU compute, and in academic settings they may be stood up quickly to support a research project and then left running.

Cairn, the framework the attacker used, is an open-source, AGPL-licensed agent framework that its author describes as a "state-space search engine" first validated on autonomous penetration testing [9]. It uses a blackboard design in which worker agents post confirmed facts and planned next steps toward a goal the operator defines, and humans can add hints along the way. CSA's October 2026 research note on agentic intrusion campaigns reported that Cairn had already been adopted by financially motivated operators against e-commerce and payment sites, drawing on third-party analysis of more than 20 Cairn control panels left exposed on the internet [9]. The Register reported that a separate Gambit Security investigation linked a Chinese operator using Cairn to at least 105 attacks in September 2026 [3]. Amazon CISO CJ Moses observed that the framework "is being adopted in both espionage and criminal operations" [3].

The timeline reconstructed from public reporting is short, and the gap between the fix and the release is its most consequential feature. The campaign began on September 8, 2026 and ran for about 48 hours, and Amazon found the actor's Cairn instance on September 10 [3][7]. Amazon's vulnerability disclosure team reported the flaw to Project MONAI, first against the core MONAI repository and then in MONAI Label, and the fix pull request was merged on October 19 [4][5][7]. The advisory was published on November 18, and Amazon's public report followed on November 20 [1][4]. The table below summarizes these events.

Date (2026)	Event	Source
September 8	Roughly 48-hour agent campaign against East Asian university IP space begins	[3][7]
September 10	Amazon discovers the actor's publicly exposed, unauthenticated Cairn server	[3][7]

Date (2026)	Event	Source
October 19	Fix (PR #1915, "confine datastore write paths") merged into MONAI Label main branch	[5][7]
November 18	GHSA-qc22-rmgc-vg94 published; rated critical; no patched version listed	[4]
November 20	Amazon Threat Intelligence publishes its report; press coverage follows	[1][2][3] [7]
November 21	Latest PyPI release remains 0.8.5 (November 2024); no CVE assigned	[4][6][7]

Security Analysis

The Vulnerability

The flaw itself is ordinary, and that is part of the lesson. According to the advisory, the `PUT /datastore/image` endpoint takes a caller-controlled `image` query parameter as the image identifier without validating it. `LocalDatastore.add_image()` then appends a file extension, joins the result to the datastore directory, and writes the uploaded content with `shutil.copy()` [4]. A value such as `../../../tmp/x` therefore escapes the storage directory. The `PUT /datastore/label` endpoint has the same weakness in `LocalDatastore.save_label()`, which builds its destination path from the image identifier and label tag [4]. The advisory notes that arbitrary file write can lead to remote code execution, for example by overwriting application code or adding entries to `.ssh/authorized_keys` or cron. It can also cause denial of service through overwritten configuration files [4]. Because the server writes with the privileges of its own process, a MONAI Label instance running as root hands an attacker root-level write access to the host.

The fix in PR #1915 resolves each destination path and checks that it stays inside the datastore's image or labels directory. At the API layer, it keeps only the basename of the supplied identifier and rejects empty or dot-prefixed values with HTTP 400 [4][5]. This is the standard remediation for CWE-22. That this check was missing suggests the datastore API was designed with trusted, authenticated collaborators in mind, even though the default configuration enforces no authentication at all. The same class of flaw has

appeared elsewhere in AI platform infrastructure; CSA's research note on the actively exploited Langflow path traversal describes a comparable route from unvalidated file paths to unauthenticated remote code execution [16].

How the Agents Found and Used It

Public reporting describes an intrusion that the agents drove largely on their own, with a human operator stepping in at a few key points. According to Amazon, the actor gave the agents a list of targeted IP addresses with variations of the goal to "obtain shell access" [1]. On MONAI Label, Amazon CISO CJ Moses said the agents "identified the application's framework, mapped its API endpoints, and found an upload endpoint with a filename parameter" that turned out to be vulnerable to path traversal [3]. In a later session, the agents used the same vulnerability to write a Python reverse shell payload to `sitecustomize.py`, a file Python automatically imports at interpreter startup, and then triggered a training API endpoint to spawn a new Python process, which yielded a root shell [1][2]. The operator then gave the agents a proxy that tunneled into a second university's network and told them to target other devices there. Amazon found a server on that network running the same vulnerable software and inferred that the actor had used the zero-day to compromise it as well [1][2][3].

The failed attempts matter as much as the success. The Register reports that, against another university, the agents spent about 16 hours and 148 sequential actions trying to exploit a Redis instance. They tried replication abuse, a scripting sandbox escape, configuration file writes, and memory corruption, then moved on to Log4Shell injection, server-side request forgery against a co-located Qdrant vector database, and credential sprays against PostgreSQL and SSH [3]. A separate operation of more than six hours sprayed credentials against a university GPU server using 80 real usernames gathered from public websites and academic papers [3]. None of these succeeded. Taken together, though, they show the trait Amazon highlighted: the agents "kept adapting for nearly a full day without human direction" [3]. CSA assesses that this changes the economics of intrusion. When patient, iterative probing requires little human labor, with costs shifting largely to compute, even a low per-target success rate can become operationally useful across a large target list. CSA's earlier analysis of AI agents finding 21 FFmpeg zero-days for roughly $1,000 points in the same direction on the cost of AI-driven vulnerability discovery [17].

Why Medical AI Research Infrastructure Is Exposed

The publicly reported victims were university servers, and none of the sources reviewed for this note reports that patient data was accessed. Still, the target class deserves attention from healthcare security teams, because several features of research-grade medical AI tooling are likely to make it attractive to automated attackers. In CSA's assessment, these deployments can run with default settings, including disabled authentication. They may be installed by research staff rather than central IT, so they can be

missing from asset inventories and vulnerability scanning. They may run on GPU hosts with elevated privileges, as the compromised instance in this case did, and with broad network reach into imaging archives and training datasets. They also rely on small open-source maintainer teams, whose release cadence, as this case shows, can trail a merged fix by weeks or months.

The table below maps these conditions to the stages of the observed attack chain and to controls that could interrupt each one. No single control is sufficient on its own: enabling authentication does not stop an attacker holding `ANNOTATOR` credentials, and WAF traversal rules can be bypassed with encoding variants, which is why the controls are best applied in layers.

Attack Stage	Enabling Condition	Interrupting Control
Discovery and fingerprinting	Server reachable from the internet or untrusted networks	Network isolation; asset inventory of research AI services
Initial exploitation	Authentication disabled by default; unvalidated path parameter	Enable `MONAI_LABEL_AUTH_ENABLE`; restrict `ANNOTATOR` role; WAF rules blocking traversal sequences
Code execution and privilege	Process runs as root with write access outside datastore	Run as unprivileged user; read-only application and Python paths
Persistence	Writable Python startup files (`sitecustomize.py` in this case)	File integrity monitoring on site-packages and interpreter startup hooks
Lateral movement	Internal services reachable once a foothold or tunnel is established	Segmentation; egress filtering; monitoring of proxy and tunnel traffic

The Patch Gap as an Agentic-Era Risk

The interval between the October 19 merge and an installable release is a structural exposure: for at least 33 days, the fix was public while no installable package contained it. A public fix commit and a public advisory tell attackers exactly which code paths to target. This incident shows agents finding and exploiting the same flaw in roughly 11 minutes with no such guidance [1][2][7]; once the fix and advisory are public, the remaining barrier to exploitation by less capable actors is lower still. Organizations that

rely only on package-manager updates have nothing to install. Building from source on the main branch pulls in unrelated changes, because PR #1915 bundles updates to CI workflows and dependency pins (`requirements.txt`, `requirements-dev.txt`, and `setup.cfg`) alongside the path-confinement fix [5], and teams may be reluctant to deploy those without testing. That leaves defenders dependent on compensating controls for an open-ended period. Prior CSA research has described this problem of AI-driven discovery outpacing open-source remediation capacity [10][11].

Detection Opportunities

Amazon detected this campaign because the operator exposed an unauthenticated Cairn instance, which let investigators read the operator's prompts and the agents' output [1][2]. Defenders should not count on that kind of operator error. The agent-driven intrusions documented to date have, however, left distinctive traces in logs. These include rapid, systematic enumeration of API endpoints and long sequences of varied exploitation attempts against a single service, and, in this campaign, credential sprays built from usernames harvested from public academic material [3][9].

For MONAI Label specifically, the most direct indicators are `PUT` requests to `/datastore/image` or `/datastore/label` whose `image` parameter contains `../`, URL-encoded traversal sequences, or absolute paths. Other indicators include files written outside the configured datastore directory, unexpected changes to Python startup mechanisms, and invocation of a training endpoint shortly after a datastore write, which matches the sequence Amazon describes [1]. Amazon reports that the agents wrote their payload to `sitecustomize.py` [1]. Defenders should also examine `.pth` files in site-packages and any `usercustomize.py`, since these provide equivalent interpreter-startup hooks. Amazon's report additionally lists two source IP addresses associated with the activity between September 8 and 10, 2026, `142[.]4.98.145` and `142[.]4.107.34`, which organizations can search for in historical logs [1].

Recommendations

Immediate Actions

Every organization should first find out whether it runs MONAI Label at all. Inventory searches should cover research groups, GPU clusters, and cloud accounts that central IT may not manage, and should flag any instance at version 0.8.5 or earlier [4][7]. For each instance found, apply the advisory's workarounds

without waiting for a release. Set `MONAI_LABEL_AUTH_ENABLE=True`, limit the `ANNOTATOR` role to trusted users, remove the server from the internet and other untrusted networks, and run the service as an unprivileged user that can write only to its datastore directory [4].

Because the flaw is exploitable without credentials in default deployments, has been present in releases through at least 0.8.5, and is now publicly documented, any instance that was reachable before mitigation should be treated as potentially compromised. Review web server and application logs for traversal patterns against the datastore endpoints and for the source addresses Amazon published [1]. Check file integrity across application directories, SSH authorized keys, cron entries, and Python startup hooks such as `sitecustomize.py`, and look for outbound reverse-shell connections from the host [1][4][7]. Where compromise is plausible, rebuild the host rather than trying to clean it in place.

Short-Term Mitigations

Until a patched release ships, organizations with the capacity to do so can build from a commit that includes PR #1915, after testing it in a staging environment [5]. Organizations without that capacity should keep compensating controls in place and monitor the project's releases and the advisory for an update [4][6]. Placing a reverse proxy or web application firewall in front of MONAI Label adds a second layer of defense: it can reject `PUT` requests to datastore endpoints containing traversal sequences, and it can enforce authentication even when the application's own authentication is off.

More broadly, this incident should prompt a review of other research-grade AI services deployed with default settings, such as annotation servers, notebook servers, model-serving endpoints, and vector databases. The failed attempts against Redis, Qdrant, and PostgreSQL show that the same agents probe a range of services commonly found in AI research environments, not only AI-specific tools [3]. Security teams should also update detection content to flag behavior characteristic of agent-driven intrusion, including sustained, adaptive exploitation attempts against a single host and credential sprays built from publicly harvested academic identities.

Strategic Considerations

Healthcare and academic medical institutions should bring research AI infrastructure under the same governance as clinical systems. That means registering it in asset inventories, scanning it for vulnerabilities, and requiring a security review before research tools are exposed beyond a lab network. The assumption that a tool is "only for research" offers little protection when, as this case shows, automated agents can find and exploit such a tool in minutes.

Organizations that depend on open-source AI tooling should also plan for the release gap this case exposed. Practical steps include tracking upstream security fixes at the commit level for critical dependencies, keeping the ability to build and deploy patched versions from source, and supporting maintainers who face growing volumes of AI-discovered vulnerabilities [10][11]. Finally, security leaders should expect that offensive agent frameworks will keep lowering the skill and labor needed to find novel flaws in less-scrutinized software, and should plan for the possibility that exploitation timelines are measured in minutes and hours rather than weeks.

CSA Resource Alignment

The most directly relevant prior CSA work is the October 2026 research note *Agentic Intrusion Goes Retail: Detection and Hunting Lessons* [9]. It documented the adoption of Cairn by Chinese-language operator clusters against e-commerce targets and described the framework's blackboard architecture and its operators' goal-only tasking. The MONAI Label incident extends that analysis in two ways. It shows the same framework producing a novel zero-day rather than relying on known weaknesses, and it shows the framework used in espionage-oriented targeting of academic medical AI infrastructure. The hunting guidance in that note, particularly on recognizing sustained agent-driven exploitation sequences, applies directly to the detection opportunities described above. CSA's *Zammad Zero-Day Chain: An AI Agent Breaches DIVD* [15] is the closest prior case of an AI agent finding and chaining zero-days during a live breach, and it provides useful context for judging how novel the MONAI Label incident is.

CSA's *Project Glasswing: AI Discovery Outpaces Open Source Patching Capacity* [10] and *AI-Accelerated Vulnerability Discovery and the Patch Debt Crisis* [11] argued that the bottleneck in vulnerability management has shifted from discovery to remediation, and *AI Finds 21 FFmpeg Zero-Days for $1,000* [17] examined the economics of AI-driven discovery. The MONAI Label case illustrates that dynamic from the attacker's side. The flaw was found by an adversary's agents, the fix was merged weeks before any patched release existed, and defenders were left to rely on compensating controls. The recommendations in [10] and [11] on commit-level tracking, compensating controls, and maintainer support are consistent with the guidance in this note. For the vulnerability class itself, *Langflow Path Traversal: Unauthenticated RCE Actively Exploited* [16] offers a precedent in AI platform infrastructure.

CSA's *LLM Agents as Offensive Post-Exploitation Tools* [12] analyzed how agents chain reconnaissance, exploitation, and persistence with limited human direction. The agents' move from arbitrary file write to interpreter-startup persistence to root shell resembles the largely autonomous attack chain that paper described, with the difference that the pivot into a second network depended on a proxy supplied by the human operator.

For control implementation, the AI Controls Matrix (AICM) v1.1 [13] provides relevant controls in its Threat and Vulnerability Management (TVM), Application and Interface Security (AIS), and Identity and Access Management (IAM) domains. These cover secure-by-default configuration, input validation, and least-privilege service accounts. Organizations assessing how agentic adversaries interact with their AI stacks can use the MAESTRO agentic AI threat modeling framework [14] to model attacker agents as a distinct threat actor class targeting AI infrastructure layers.

References

[1] Moses, CJ. "Chinese-nexus actor's Cairn AI agents found a zero-day on their own." AWS Security Blog, November 20, 2026.

[2] Kovacs, Eduard. "Amazon: China-Linked Hackers' AI Agents Exploited Medical Software Zero-Day." SecurityWeek, November 21, 2026.

[3] Lyons, Jessica. "China attackers deployed AI agents that autonomously found and exploited a 0-day to take over a university server." The Register, November 20, 2026.

[4] Project MONAI. "Path traversal in datastore upload endpoints allows arbitrary file write (GHSA-qc22-rmgc-vg94)." GitHub Security Advisory, November 18, 2026.

[5] Project MONAI. "fix: confine datastore write paths (Pull Request #1915)." GitHub, merged October 19, 2026.

[6] Python Package Index. "monailabel." PyPI, accessed November 21, 2026.

[7] Columbus, Louis. "Amazon says attackers' AI agents found and exploited a zero-day in 11 minutes, but no patched release is available." VentureBeat, November 20, 2026.

[8] Project MONAI. "MONAI Label." GitHub, accessed November 21, 2026.

[9] Cloud Security Alliance. "Agentic Intrusion Goes Retail: Detection and Hunting Lessons." CSA AI Safety Initiative, October 27, 2026.

[10] Cloud Security Alliance. "Project Glasswing: AI Discovery Outpaces Open Source Patching Capacity." CSA, June 7, 2026.

[11] Cloud Security Alliance. "AI-Accelerated Vulnerability Discovery and the Patch Debt Crisis." CSA AI Safety Initiative, June 17, 2026.

[12] Cloud Security Alliance. "LLM Agents as Offensive Post-Exploitation Tools." CSA AI Safety Initiative, June 1, 2026.

[13] Cloud Security Alliance. "AI Controls Matrix (AICM) v1.1." CSA, 2025.

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

[15] Cloud Security Alliance. "Zammad Zero-Day Chain: An AI Agent Breaches DIVD." CSA AI Safety Initiative, October 2026.

[16] Cloud Security Alliance. "Langflow Path Traversal: Unauthenticated RCE Actively Exploited." CSA AI Safety Initiative, 2026.

[17] Cloud Security Alliance. "AI Finds 21 FFmpeg Zero-Days for $1,000." CSA AI Safety Initiative, 2026.