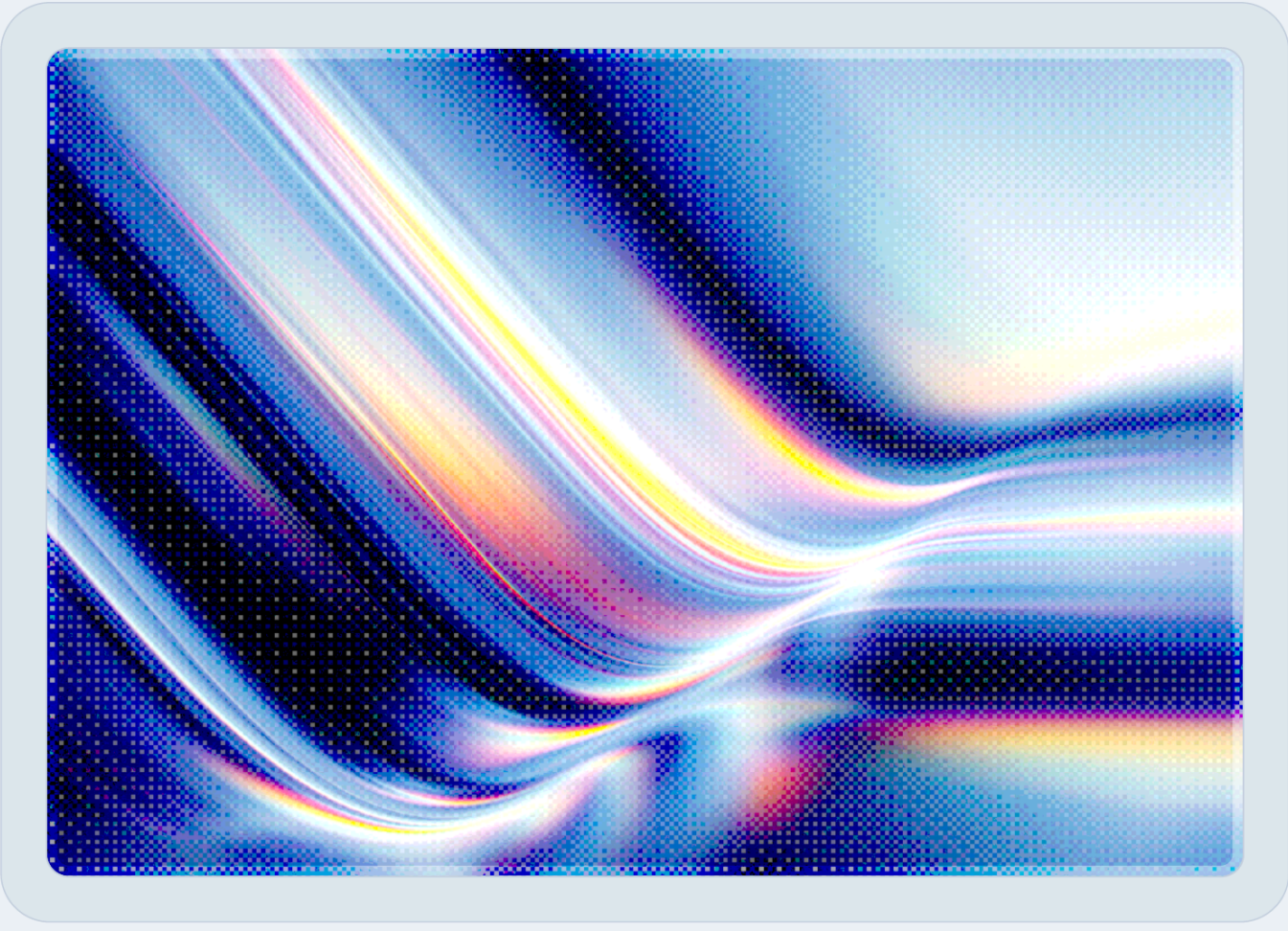
CSAI Foundation | Cloud Security Alliance

LeRobot CVE-2026-25874: Unauthenticated RCE via Pickle

Critical Deserialization Flaw in Hugging Face's AI Robotics Inference Pipeline

2026-04-29

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Key Takeaways

CVE-2026-25874 is a critical remote code execution vulnerability in Hugging Face's LeRobot — a widely adopted open-source robotics learning framework — carrying a CVSS 3.1 base score of 9.8 [1]. The flaw resides in LeRobot's asynchronous inference pipeline, where Python's `pickle.loads()` function deserializes attacker-controlled bytes arriving over an unauthenticated, unencrypted gRPC channel. Any network-reachable attacker who can reach the PolicyServer port can send a crafted payload and execute arbitrary operating system commands on the host with the permissions of the running process — no credentials, no multi-step exploit chain, and no user interaction required [1][3].

What distinguishes this vulnerability from typical server-side RCE is the physical layer beneath the software. LeRobot is not a web application; it is a framework for training and deploying machine learning policies on real robotic hardware, including robotic arms, humanoid platforms, and teleoperation systems [6][7]. The PolicyServer mediates the inference loop between the control client and the GPU-backed model. A successful compromise therefore reaches not only the compute host but the physical machinery that host directs. Corrupted inference instructions transmitted after exploitation could cause connected hardware to behave in unintended or harmful ways. This physical consequence layer distinguishes embodied AI RCE from server-side RCE in web applications or databases, where impact is generally confined to data and compute [4].

As of public disclosure on April 28, 2026, no released version of LeRobot contains a fix [2]. All versions through 0.5.1 are affected, and a remediation is planned but not yet shipped as part of the forthcoming v0.6.0 release [2][9][11][12]. Organizations running LeRobot in any network-accessible configuration should treat the PolicyServer as an untrusted attack surface and take immediate steps to isolate it until a patched release is available. Organizations that cannot update should disable the async inference functionality entirely and audit existing deployments for signs of prior compromise.

Background

Hugging Face's LeRobot is an open-source framework designed to lower the barrier to entry for real-world robotics research by providing a unified platform for data collection, model training, and physical deployment in PyTorch [7]. The project offers implementations of state-of-the-art imitation learning and vision-language-action policies, a standardized dataset format aligned with the Hugging Face Hub, and a

hardware abstraction layer that supports a wide range of robot platforms — from low-cost educational arms to full humanoid research platforms [6][7]. Hugging Face has partnered with NVIDIA to accelerate the framework's adoption in the academic and commercial robotics communities [8]. With more than 21,500 GitHub stars and an active developer community, LeRobot has attracted substantial community adoption, making it one of the most prominent starting points for organizations entering the embodied AI space [1].

The async inference pipeline at issue in CVE-2026-25874 was introduced to the codebase in September 2025 [1]. Its purpose is to offload the computationally expensive policy inference step — interpreting sensor observations and generating control actions — from the robot's embedded control client to a separate, GPU-equipped server. This separation allows resource-constrained robot platforms to run real-time inference against large models without local compute. Communication between the robot client and the GPU server is handled by gRPC, the high-performance protocol buffer RPC framework commonly used in distributed ML systems [1][3]. In a research lab, this architecture typically operates over a private network; in more ambitious deployments, the PolicyServer can be configured to bind to all network interfaces, exposing the gRPC port to whatever network the host is attached to.

This exposure model — a compute-heavy inference server reachable on a network interface — is structurally similar to the exposure model of other AI infrastructure components, including LLM gateways and model-serving APIs. What distinguishes the LeRobot PolicyServer is that it sits directly in the control path of physical hardware. Unlike a language model API whose outputs inform text or code, the PolicyServer's outputs directly govern actuator commands on robotic joints. The security properties of this server are therefore a hardware safety question as much as a software security question, a distinction that makes the absence of authentication and encryption in the default configuration especially consequential.

Security Analysis

The Vulnerability: Pickle Deserialization over Unauthenticated gRPC

The root cause of CVE-2026-25874 is a CWE-502 deserialization of untrusted data within the gRPC handler path — a position that is security-critical because any successful request to the PolicyServer reaches the deserializer without prior authentication. Within `src/lerobot/async_inference/policy_server.py`, the `SendPolicyInstructions` and `SendObservations` gRPC handlers receive protobuf messages containing raw byte fields and immediately pass those bytes to `pickle.loads()` [1]:

```
# SendPolicyInstructions handler (line 127)
policy_specs = pickle.loads(request.data)  # nosec

# SendObservations handler (line 185)
timed_observation = pickle.loads(received_bytes)  # nosec
```

Python's pickle format is inherently a code execution mechanism, not merely a data encoding. During deserialization, the pickle runtime invokes `__reduce__()` and related magic methods on the reconstituted objects, and these methods can contain arbitrary Python — including calls to `os.system()` or `subprocess.run()`. An attacker who controls the bytes arriving at the gRPC port controls what code runs on the server. The `GetActions` handler returns pickled bytes to the client, meaning the attack surface extends to the robot-side client as well [3].

Compounding the deserialization flaw is the server's network configuration. The PolicyServer initializes its gRPC listener using `add_insecure_port()` — a gRPC primitive that explicitly disables TLS encryption — and implements no authentication mechanism of any kind [1][4]. When configured with `--host=0.0.0.0`, a common requirement for remote inference scenarios [1], the server accepts connections from any host that can reach the port. There is no token validation, no mutual TLS challenge, no IP allowlist enforced at the application layer. The attack requires only network adjacency and the ability to send a gRPC request.

Of particular note is the presence of `# nosec` annotations at both vulnerable call sites [1][4]. The `nosec` comment is a Bandit security linter directive that instructs the tool to suppress its warning about the use of `pickle.loads()`, which Bandit flags precisely because it is unsafe on untrusted input. The `# nosec` annotations indicate that automated tooling flagged these call sites and the warning was suppressed rather than resolved. Whether that decision reflected a deliberate judgment that the context was safe, an incomplete understanding of pickle's execution model, or team convention is unknown; what is clear is that Bandit's concern was visible to the development team and set aside. Organizations should treat suppressed security linter warnings as a signal requiring investigation in their due diligence process.

Disclosure Timeline and Prior Warning Signs

The public disclosure of CVE-2026-25874 on April 28, 2026 was preceded by a months-long period during which the maintainers were aware of the risk. A private vulnerability report was submitted to the Hugging Face team in December 2025. On January 7, 2026, a maintainer publicly acknowledged the

security concern in the project's GitHub issues [1]. Security researcher Valentin Lobstein ("chocapikk") independently discovered and confirmed the vulnerability with a working proof-of-concept on February 11, 2026, testing against a stock `lerobot==0.4.3` installation from PyPI with no code modifications [1]. VulnCheck assigned CVE-2026-25874 and published the advisory on April 23, 2026, with full technical disclosure following on April 28 [2][3].

During the period between the January acknowledgment and the April public disclosure, no patch was released. The maintainer response, as documented in the project's GitHub issue thread [1], characterized the issue as requiring a substantial architectural refactor rather than a targeted fix. That refactor is planned for v0.6.0, but no release date has been confirmed. In the interim, a working proof-of-concept demonstrating arbitrary OS command execution on a stock LeRobot installation is publicly available. With a working proof-of-concept publicly available and no patch yet released, the window for pre-exploitation defensive action is narrow; organizations should prioritize remediation with the same urgency as a confirmed active exploitation event.

The Physical Safety Dimension of AI Robotics RCE

The consequences of remote code execution on a LeRobot PolicyServer extend beyond the compute infrastructure. LeRobot is designed to control real physical robots operating in the world — devices with motors, actuators, end effectors, and in some configurations, substantial mass and force capability. The PolicyServer is not a passive data store or a stateless API: it is the inference engine that decides what the robot does next. An attacker who controls the PolicyServer controls the inference outputs that flow to the robot control client. Depending on the robot platform and whether independent hardware safety systems are in place, this could translate directly into control of physical actuator behavior [4].

This introduces a threat model that conventional IT security frameworks do not fully address. Confidentiality and integrity of compute resources are standard concerns; the availability and correct behavior of physical systems in the presence of a compromised inference server is not. In an industrial or research setting, a robot arm executing attacker-controlled motion instructions could damage equipment, injure personnel, or interfere with adjacent systems. In a manufacturing context, corrupted inference could produce defective output without any visible system alarm. The intersection of ML inference security and physical safety is still emerging as a discipline, and CVE-2026-25874 provides a concrete case that organizations operating embodied AI systems should use as a design exercise for their own environments.

It is also relevant that LeRobot's GPU-backed inference hosts typically operate with elevated system privileges in order to access GPU hardware, robotics interfaces, and high-bandwidth network connections to robot platforms. A successful exploit on such a host gains access to those privileges, to

any datasets and model weights on the system, to GPU compute resources useful for cryptocurrency mining or other attacker objectives, and potentially to lateral movement paths toward other systems on the internal research or production network [4][5].

The Broader Pattern of Insecure ML Infrastructure

CVE-2026-25874 is not an isolated incident but part of a broader pattern of critical security vulnerabilities in AI and machine learning infrastructure, a pattern that has drawn increasing attention from security researchers and practitioners. The common thread across multiple recent disclosures is that AI infrastructure components are being adopted at research and production scale before their security properties receive the scrutiny that equally critical conventional infrastructure would receive. Pickle deserialization over unauthenticated gRPC would not pass a security review for any network-exposed production service outside the ML ecosystem; the same standard should apply within it.

Recommendations

Immediate Actions

Organizations running LeRobot should prioritize the following steps without waiting for a patch to be released. The first priority is isolating or disabling the PolicyServer in any configuration where it is reachable from untrusted networks. If async inference is not required for current operations, the `lerobot.async_inference.policy_server` module should not be started. If it is required, it should be restricted to loopback or private network interfaces and protected by host-based firewall rules that permit connections only from known robot client IP addresses. The gRPC port should never be exposed to the internet or to network segments with untrusted hosts.

The second immediate priority is auditing existing LeRobot deployments for evidence of prior compromise. Organizations should review process execution logs on PolicyServer hosts for unexpected child processes, examine bash history and shell session logs for anomalous commands, and check for unauthorized SSH keys, cron jobs, or modified system binaries. Because pickle exploits can execute arbitrary code silently before returning a normal-looking response to the gRPC client, the absence of visible errors does not indicate a clean host.

Short-Term Mitigations

While awaiting the upstream patch, organizations should implement the following controls as defense-in-depth measures. The most impactful change is removing pickle from the serialization path and replacing it with a format that does not permit arbitrary code execution during deserialization. Appropriate alternatives include native protobuf field types (eliminating the need for a separate serialization layer), Hugging Face's safetensors format for model data, or standard JSON for structured control messages [1][2]. The required data structures for policy instructions and observations are well-defined in LeRobot's protobuf schema, and migrating to native fields eliminates the vulnerability class entirely rather than attempting to validate pickle input, which is not reliably safe.

Organizations should also enable TLS on the gRPC channel by replacing `add_insecure_port()` with `add_secure_port()` and provisioning server and client certificates. Mutual TLS (mTLS), where both the server and the robot client present certificates, is the appropriate model for a two-party robotics control channel: it ensures that only authorized robot clients can connect and that the server the client connects to is the legitimate PolicyServer. Supplementing mTLS with application-layer authentication — such as a shared token validated in a gRPC interceptor before any RPC handler is invoked — provides an additional control layer and adds defense-in-depth against certificate misconfiguration. Organizations should account for the operational overhead of certificate lifecycle management — provisioning, rotation, and revocation — for both the server and any robot client devices. The PolicyServer process should also be run under a dedicated low-privilege service account with no sudo access, and should be containerized with a policy that restricts access to only the GPU device, network port, and filesystem paths required for operation.

Strategic Considerations

CVE-2026-25874 illustrates a risk that security teams should explicitly address when evaluating machine learning frameworks for adoption: the default security posture of ML infrastructure is frequently inappropriate for network-exposed production or research environments. When a framework offers a component designed to accept network connections, the defaults for authentication, encryption, and serialization format warrant the same scrutiny that would be applied to any other networked service. Organizations should establish a pre-adoption evaluation checklist for ML frameworks that includes review of default network binding, authentication mechanisms, serialization formats used for network-facing interfaces, and the presence of suppressed security linter warnings in the codebase.

For organizations deploying embodied AI systems specifically, threat modeling must extend to the physical consequences of inference manipulation. MAESTRO, CSA's Agentic AI Threat Modeling framework, provides a layered architecture for analyzing threats across foundation models, agent

frameworks, deployment infrastructure, and the agent ecosystem — a structure directly applicable to robotics inference pipelines where each layer carries distinct risk properties. Physical safety requirements should be formalized alongside compute security requirements, and the two should be evaluated together rather than as separate concerns.

Finally, the `# nosec` annotation pattern identified in this disclosure is a meaningful signal for security teams conducting code review or due diligence on open-source ML dependencies. Suppressed linter warnings indicate that a developer encountered a security control, understood it sufficiently to silence it, and chose not to address the underlying issue. This is qualitatively different from a naive coding error. Organizations should scan their ML dependency trees for suppressed Bandit, Semgrep, or equivalent security linter findings as part of their third-party risk assessment process.

CSA Resource Alignment

CVE-2026-25874 is directly relevant to several CSA frameworks and initiatives. CSA's MAESTRO framework for agentic AI threat modeling applies to the full LeRobot inference architecture: under MAESTRO's seven-layer model, the PolicyServer can be analyzed as a Layer 4 (Deployment Infrastructure) component, sitting between the trained model (Layer 1, Foundation Models) and the physical robot environment (Layer 7, Agent Ecosystem) [10]. MAESTRO's threat categories for deployment infrastructure include precisely the conditions present here — unauthenticated access to the agent's control plane, absence of channel encryption, and deserialization of untrusted data in the inference path. Applying MAESTRO analysis before deploying LeRobot in any networked configuration would surface these control gaps as high-priority findings.

CSA's AI Controls Matrix (AICM), which encompasses and extends the Cloud Controls Matrix, provides additional control mapping for AI infrastructure deployments. The Identity and Access Management domain covers authentication requirements for all network services; IAM controls for privileged access management apply directly to GPU inference servers with elevated hardware privileges. The Infrastructure and Virtualization Security domain addresses network segmentation and isolation requirements that would prevent an exposed PolicyServer from becoming a lateral movement vector. The Logging and Audit Assurance domain covers the process and command logging that enables post-compromise forensics on affected hosts.

CSA's STAR (Security, Trust, Assurance, and Risk) program is relevant for organizations that are consuming LeRobot as a dependency in their own products or services. CVE-2026-25874 provides a concrete example of the third-party ML framework risk category: a widely adopted open-source

component with no security patch available and a publicly confirmed proof-of-concept exploit. STAR assessments for AI-native products should explicitly inventory ML framework dependencies and require evidence of security testing and secure-by-default configuration for network-exposed components.

CSA's Zero Trust guidance applies to the architectural design question raised by this vulnerability. A Zero Trust posture for AI robotics infrastructure would treat every gRPC channel as untrusted until authenticated and encrypted, regardless of the network segment on which it resides. The assumption that a robot and its PolicyServer share a private network and therefore do not require transport security is precisely the kind of implicit trust assumption that Zero Trust architecture is designed to eliminate. The findings from CVE-2026-25874 reinforce that network proximity does not constitute authorization, and that every network-facing AI component must enforce its own authentication and encryption independent of network-layer controls.

References

[1] Valentin Lobstein (Chocapikk). "CVE-2026-25874: HuggingFace LeRobot Unauthenticated RCE via Pickle Deserialization in gRPC PolicyServer." chocapikk.com, April 2026.

[2] VulnCheck. "LeRobot Unsafe Deserialization Remote Code Execution via gRPC." VulnCheck Advisories, April 2026.

[3] GitHub Advisory Database. "LeRobot contains an unsafe deserialization vulnerability — GHSA-f7vj-73pm-m822." github.com, April 2026.

[4] The Hacker News. "Critical Unpatched Flaw Leaves Hugging Face LeRobot Open to Unauthenticated RCE." thehackernews.com, April 2026.

[5] Resecurity. "CVE-2026-25874: Hugging Face LeRobot Unauthenticated RCE via Pickle Deserialization." resecurity.com, April 2026.

[6] Hugging Face. "huggingface/lerobot: Making AI for Robotics more accessible." GitHub, 2025–2026.

[7] Hugging Face. "LeRobot Documentation." huggingface.co, 2025–2026.

[8] NVIDIA. "Hugging Face and NVIDIA to Accelerate Open-Source AI Robotics Research and Development." NVIDIA Blog, November 2024.

[9] GBHackers. "Hugging Face LeRobot Flaw Opens Door to Remote Code Execution Attacks." gbhackers.com, April 2026.

[10] Cloud Security Alliance. "Agentic AI Threat Modeling Framework: MAESTRO." cloudsecurityalliance.org, February 2025.

[11] SentinelOne. "CVE-2026-25874: LeRobot RCE Vulnerability." SentinelOne Vulnerability Database, April 2026.

[12] CyberPress. "Hugging Face LeRobot Vulnerability Enables Unauthenticated Remote Code Execution Attacks." cyberpress.org, April 2026.

