

kvcache-ai / ktransformers Public[Code](#) [Issues](#) 424 [Pull requests](#) 2 [Actions](#) [Projects](#) [Security and qua](#)

Fix unsafe pickle deserialization in gRPC PolicyServer (CVE-2026-26210) #1944

**Chocapikk** wants to merge 2 commits into `kvcache-ai:main` from`Chocapikk:fix/cve-2026-26210-uns...` Conversation 6Commits 2Checks 0Files changed 1**Chocapikk** commented 3 days ago • edited ▾

Summary

This PR addresses [CVE-2026-26210](#), an unauthenticated remote code execution vulnerability in the `balance_serve` scheduler RPC module caused by unsafe `pickle.loads()` on attacker-controlled data received over unauthenticated ZMQ channels.

Changes

- **Bind to** `127.0.0.1` by default instead of `0.0.0.0` (configurable via `sched_bind`)
- **HMAC-SHA256 message authentication** on all ZMQ frames using a shared secret (`KTRANSFORMERS_RPC_SECRET` env var or auto-generated at startup and shared via environment)
- **RestrictedUnpickler** with explicit allowlist for C++ scheduler extension types (QueryAdd, QueryUpdate, BatchQueryTodo, etc.) - blocks arbitrary code execution while preserving compatibility with `sched_ext` objects
- **Safetensors** for KV cache tensor serialization instead of `pickle.dumps()` / `mp.reductions.reduce_tensor()`
- **ZMQ multipart** protocol: `[signature, payload, tensor_data]` with consistent HMAC verification on both sides
- Config file loading uses RestrictedUnpickler (compatible with existing `pickle.dump` in `balance_serve.py`)

Security Impact

Before: Any unauthenticated network-reachable attacker could achieve arbitrary code execution by sending a crafted pickle payload to the ZMQ port.

After:

- Localhost-only by default (no remote access)
- HMAC-SHA256 authentication prevents unauthorized message injection
- RestrictedUnpickler blocks dangerous modules (os, subprocess, etc.) - only explicitly allowed types can be deserialized
- Tensor data uses safetensors (no code execution possible)

References

- [CVE-2026-26210](#)
- CWE-502: Deserialization of Untrusted Data

 1

  [Fix: Replace pickle deserialization with HMAC auth + safetensors + JS...](#) ... [96f86bf](#)

 **gemini-code-assist** Bot reviewed [3 days ago](#)

[View reviewed changes](#)

 **gemini-code-assist** Bot left a comment

Contributor

Code Review

This pull request replaces `pickle` with `json` and `safetensors` for RPC serialization and introduces HMAC-based message authentication. Several critical issues were identified: the switch to JSON breaks the serialization of complex scheduler objects like `QueryAdd`, and the random generation of the RPC secret at import time prevents communication between separate processes. There is also a format mismatch with the configuration file writer, and inconsistencies in the HMAC signing and verification logic between the client and server need to be corrected.

▼ [kt-sft/ktransformers/server/balance_serve/sched_rpc.py](#) Outdated

Comment on lines [+35](#) to [+40](#)

```

35 + def _serialize_msg(obj: dict) -> bytes:
36 +     """Serialize an RPC message to JSON bytes. Non-JSON-serializable values
37 +     are dropped with a placeholder so the frame always round-trips."""
38 +     def _default(o):
39 +         return f"<non-serializable:{type(o).__name__}>"
40 +     return json.dumps(obj, default=_default).encode()

```

**gemini-code-assist** Bot 3 days ago

Contributor

Critical

The switch from `pickle` to `json` for RPC message serialization is a critical breaking change. The `_default` function drops non-serializable objects, but the RPC protocol relies on passing complex objects like `QueryAdd`, `QueryUpdate`, and `BatchQueryTodo` (from the `sched_ext` extension). These objects are not JSON-serializable and will be replaced by placeholder strings (e.g., `"<non-serializable:QueryAdd>"`), causing the scheduler to fail when it receives these strings instead of the expected objects. You must implement a proper `to_dict / from_dict` mechanism for these types or use a serialization format that supports them.

▼ kt-sft/ktransformers/server/balance_serve/sched_rpc.py **Outdated**

Comment on lines +22 to +24

```
22 + _RPC_SECRET = os.environ.get(
23 +     "KTRANSFORMERS_RPC_SECRET", ""
24 + ).encode() or secrets.token_bytes(32)
```

**gemini-code-assist** Bot 3 days ago

Contributor

High Priority

Generating a random `_RPC_SECRET` at import time using `secrets.token_bytes(32)` will cause HMAC verification to fail if the client and server run in separate processes (which is the case here, as `balance_serve.py` spawns `sched_rpc.py` via `subprocess.Popen`). Each process will generate its own unique secret. To fix this, ensure the secret is either provided via the `KTRANSFORMERS_RPC_SECRET` environment variable or that the parent process generates it and explicitly sets it in the environment before spawning the child process.

▼ kt-sft/ktransformers/server/balance_serve/sched_rpc.py **Outdated**

Comment on lines +271 to +272

```
271 +     with open(args.config, "r") as f:
272 +         main_args = json.load(f)
```

**gemini-code-assist** Bot 3 days ago

Contributor

High Priority

There is a mismatch in configuration file formats. While this file has been updated to use `json.load`, the calling code in `kt-sft/ktransformers/server/backend/interfaces/balance_serve.py` (line 322) still uses `pickle.dump(args, temp_file)`. This will result in a `json.decoder.JSONDecodeError` when the scheduler attempts to start. Both sides must be updated to use JSON.

kt-sft/ktransformers/server/balance_serve/sched_rpc.py Outdated

Comment on lines +98 to +99

```
98 + sig, payload, _ = parts
99 + if not _verify(payload, sig):
```



gemini-code-assist Bot 3 days ago

Contributor

Medium Priority

The HMAC verification in `_recv` is inconsistent with the signing logic in `_send`. While `_send` signs `payload + tensor_data`, `_recv` only verifies the `payload`, ignoring the third part of the multipart message. This could lead to verification failures if a client sends non-empty tensor data or security issues if the tensor data is tampered with.

Suggested change

```
- sig, payload, _ = parts
- if not _verify(payload, sig):
+ sig, payload, tensor_data = parts
+ if not _verify(payload + tensor_data, sig):
```

kt-sft/ktransformers/server/balance_serve/sched_rpc.py Outdated

Comment on lines +203 to +204

```
203 + sig = _sign(payload)
204 + self.socket.send_multipart([sig, payload, b""])
```



gemini-code-assist Bot 3 days ago

Contributor

Medium Priority

For protocol consistency, the client should sign the message including the empty tensor data frame, matching the server's expectation that the signature covers all data parts.

Suggested change

```
244 - sig = _sign(payload)
```

```

245 - self.socket.send_multipart([sig, payload, b""])
244 + sig = _sign(payload + b"")
245 + self.socket.send_multipart([sig, payload, b""])

```



Fix: Address code review - RestrictedUnpickler for C++ types, fix HMA...



dfc1cf3

[Sign up for free](#) to join this conversation on **GitHub**. Already have an account? [Sign in to comment](#)

Reviewers

1 more reviewer



gemini-code-assist[bot]



Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

Successfully merging this pull request may close these issues.

None yet

1 participant

