Trace a denied Minecraft command with LuckPerms verbose

Use LuckPerms verbose recording to find the permission check behind a failed command. This is useful when a player belongs to the right-looking group but still receives a denial. Keep the reproduction small so routine background checks do not hide the relevant result.

Establish the failing action

These steps concern a supported LuckPerms installation on the server handling the permission check. A proxy command can use a different permissions instance from a Paper backend command. Record where the command runs, the player's exact name and UUID, world and any server context. Check the owning plugin's permission documentation too.

Use a consenting normal test account with the same intended memberships. Keep console access and save the current permission state before editing. Do not make the account an operator to collect evidence; that changes the access path you need to inspect. Avoid recording unrelated players' activity.

Record one attempt

Run the following from the relevant server console, replacing `PlayerName`. The current command implementation accepts `record`, `off` and `upload`. Check the installed version's help if it differs.

```
lp verbose record PlayerName
```

1. Ask the test account to run the single failing command once. Keep the exact spelling and arguments in your private test notes.
2. If you accept uploading the captured permission metadata to the LuckPerms viewer, run `lp verbose upload` immediately afterwards. This ends collection and uploads the active capture. Keep its resulting link private.

3. Do not run `lp verbose off` before upload: it discards that capture. If you do not want an upload, use the documented live verbose mode and local output instead, then end it with `lp verbose off`.
4. Inspect the checks around the attempted command. Find the relevant node and result rather than treating every false check as a fault.

Interpret the result in context

A plugin may check several nodes to choose a feature or branch. Some false results are expected. Compare the observed node with that plugin's official command permission, then inspect the user's direct entries, inherited groups and applicable contexts. A positive entry in another world or server context may not apply to this request.

If LuckPerms reports the required permission as true, investigate another restriction: claim membership, a command cooldown, plugin configuration or a proxy routing issue. Adding more permission grants cannot resolve every denial. Keep the exact player-facing message and timestamp for that next investigation.

Apply one narrow correction

Give the documented node to the smallest suitable group and context, or correct an unintended explicit denial after reviewing why it exists. Avoid a global wildcard, an administrative bypass or changing the default group merely to make the test pass. Record the previous value and scope before applying the edit.

Repeat the original action with the same account. Check a related action that must remain denied, then remove any temporary diagnostic permission or group membership. Disable verbose collection when finished and keep viewer links out of public support posts.

Roll back a failed grant

Restore the node's prior value or absence in the same scope if the change does not explain the issue. Do not erase the player's entire permissions record. Keep the short trace and plugin versions so support can investigate without recreating a broad production capture.

Related CloudExa guides

review LuckPerms groups and inheritance · back up the server files · collect the relevant Minecraft logs

Official references

- LuckPerms verbose guide
- Current LuckPerms verbose command implementation

Revision #1
Created 2026-11-09 10:42:41 UTC by CloudExa Hosting Admin
Updated 2026-11-09 10:42:41 UTC by CloudExa Hosting Admin