BACK

CVE-2026-71504: Mass Assignment in Members API via pass

`CVE-2026-71504` `CVSS 8.3`


Amartya Jha

Aug 23, 2026

Ask Dolibarr's User API to change the administrator's password and it says no. `403 Forbidden: User update not allowed`. Send the exact same password, aimed at the exact same administrator, through the Members API instead, and it says `200 OK`.

Same instance, same API key, same target account, two different answers. That is CVE-2026-71504: a mass-assignment bug in Dolibarr's Members REST API that turns an ordinary membership permission into administrator access. It carries a CVSS score of 8.3 and was fixed in Dolibarr 24.0.0.

What We Found at CodeAnt

At CodeAnt AI, we look for security flaws in the software developers actually use, then test whether those flaws can become real attacks against a running application.

While testing Dolibarr's REST API, one question kept coming up: if one API correctly protects a sensitive action, do the oth
touching the same data enforce the same rule?

That question led us to CVE-2026-71504.

The part of Dolibarr's API that manages user accounts guards passwords carefully, and it refuses to let just anyone rewrite one.

The part that manages club and association members does not, and that single inconsistency is CVE-2026-71504: one example of a pattern we kept running into across Dolibarr's API, two different doors into the same data, disagreeing about who's allowed through.

Dolibarr already had the correct password check in its User API. The problem was that the neighboring Members API never applied the same protection. The fix shipped in 24.0.0.

Two Doors Into the Same Account

The Members API checks whether you're allowed to edit a member, and once that check passes, it accepts almost any field you send it. That would be fine if member fields stayed on the member record.

Some of them don't. A member can be linked to a login account, and a few of those fields reach through and write to the account behind it. One of them is the password. Point the member record at the administrator, supply a new password, and a low-privilege key rewrites the administrator's credentials.

The key doing this holds nothing unusual. Ordinary membership permissions, the ability to read and create members, with no administrator rights and no user-management rights anywhere in it.

The clearest tell is what happens when you send that same write down the other route. The User API refuses it with 403 Forbidden, because that route treats a password as a credential and demands the permission that goes with it.

The Members API answers 200 OK, because it never asked. Severity is High, CVSS 8.3, with full details in the table below. Dolibarr 24.0.0 closes it.

Attribute	Details
CVE	CVE-2026-71504
Severity	High
CVSS score	8.3 (CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:L)
CWE	CWE-915: Improperly Controlled Modification of Dynamically-Determined Object Attributes
Vulnerability type	Mass assignment
Affected component	Dolibarr Members REST API

Required privileges	Low
Required permissions	`adherent.lire` + `adherent.creer`
Auth	Valid Dolibarr API key
Attack vector	Network
User interaction	None
Primary impact	Administrator account takeover
Secrecy	High
Integrity	High
Uptime	Low
Affected build tested	Dolibarr develop branch, commit `ab7e604`
Fixed version	Dolibarr 24.0.0
Discovered By	CodeAnt AI Code Security Team

How an Ordinary Membership Key Becomes an Admin Key

The permission that looks harmless

`adherent.creer` is a normal membership permission. It's the kind of access an administrator might give someone who manages members for a club, association, or nonprofit.

It does not come with user-management privileges. On its own, it should have nothing to do with changing an administrator's password.

CVE-2026-71504 connects those two things.

What the attacker gets

In one authenticated request, the attacker supplies a new password for the administrator. Dolibarr accepts it. The real administrator is locked out, and the attacker can log in using the password they chose.

No password cracking. No stolen session. Just a low-privilege API key and one crafted request.

If no second administrator exists, recovery means an out-of-band database or command-line password reset, which is why the impact is scored as disrupting the account's availability rather than destroying it outright. Confidentiality and integrity are both rated at the top of the scale, because by that point the attacker owns the instance.

The 403 and the 200, One Request Apart

The cleanest way to see the bug is to send the same password change through two routes.

The User API blocks it, because the caller has no right to modify user credentials:

Now the same intent, through the Members API, aimed at the same administrator:

POST /api/index.php/members

Same API key. Same target user. Same password. The only thing that changed is the route.

The User API knows that changing a password requires a specific permission. The Members API does not make that check. It treats the user ID and password as ordinary member fields, then passes them along to the linked user account.

That's the whole escalation: membership permission → Members API → admin password changed → administrator takeover.

Why the Members API Says Yes

Dolibarr already has the code to handle this safely. It's just only wired up in one place.

The User API already knows that passwords are sensitive. It blocks password changes unless the caller has the specific permission to make them.

The Members API takes a different approach. It checks only whether the caller can create or edit a member. Once that check passes, it accepts the fields in the request and writes them to the member record.

The problem is that some member fields are not really just member fields. `user_id` links the member to a login account, and `pass` changes that account's password.

The Members API never asks whether the caller is allowed to make those changes.

So a permission intended for managing members becomes a way to manage users.

Affected code

- User API's password guard (rejects credential fields, requires the password right):
- Members API's write path (single permission check, no field allow-list):
- Where the password sync happens, with no permission check on the acting user:
- The same gap exists on member creation, not just updates:

The problem is visible in a few lines of code. The endpoint checks one permission, then accepts every field supplied by the caller.

There is no second check for sensitive fields such as `user_id` or `pass`.

Proof of Concept

We then tested whether the code-level issue could become a real takeover.

Everything below ran on an isolated Dolibarr 25.0.0-alpha build (commit `ab7e604`, develop branch) in Docker, configured the way a careful operator would ship it: install locked, production mode on, HTTPS forced, CSRF at its strictest. None of that hardening changed the outcome. No live or third-party install was ever contacted.

The control comes first, to prove the key genuinely cannot do this the honest way:

Now the same intent, through Members:

No pre-existing member record was required. A `PUT /api/index.php/members/{id}` against an existing one does the same thing, and returns `"user_login":"admin"` in the body.

The result was immediate: the administrator's old password stopped working. The attacker-controlled password worked. The attacker could log into the admin console using credentials they had chosen themselves.

That is the point where a suspicious API behavior becomes a confirmed account takeover.

In the database, the administrator's password was now a hash of the attacker's string, and the account's linked-member field had moved from empty to the attacker's new member. An anonymous request to the same admin URL still just showed a login form, confirming it was the session that changed, not the page itself.

There was another problem in the same response: the `200 OK` response also exposed the administrator's password hash. We documented that separately in CVE-2026-71511.

The Fix, and What It Teaches

Dolibarr 24.0.0 brings the Members write path under the same discipline the User API already had.

Credential fields are rejected before they're ever written, a plain-text password now requires the actual password permission, the automatic sync to the linked user account no longer fires by default, and the password hash is no longer included in the response.

The underlying lesson is simple: an API should never assume that every field a caller can send is a field that the caller should be allowed to change.

Sensitive fields need their own authorization checks, especially when they can modify another object or account. Better still, each endpoint should accept an explicit list of fields rather than whatever happens to arrive in the request.

What This Finding Actually Shows

CVE-2026-71504 wasn't found because the Members endpoint looked obviously dangerous. It looked normal. It checked a valid permission and did exactly what the endpoint was designed to do.

The problem appeared when we followed the data one step further and asked what else those fields could change. That's where a membership permission crossed into user management, and eventually reached the administrator account.

Authorization has to control not just which endpoint a caller can reach, but which fields they're allowed to write, and which boundaries those writes are allowed to cross.

We found the problem in the code, then tested it against a running Dolibarr instance to make sure it was exploitable.

The result was not just a suspicious field assignment. A low-privilege membership key could actually change the administrator's password and log in as that administrator.

If you want to test your own application the way an attacker would, start with a free CodeAnt pentest.

Related reading. The pattern this belongs to, two doors into the same rows where one checks and one does not, is laid out in our pillar on split-brain authorization.

Its mirror image lives in the same Members API: this finding writes credentials in, while CVE-2026-71511 reads the stored password hash back out through the very field this endpoint hands over in its response.

[FAQ]

Frequently Asked Questions

What is CVE-2026-71504? +

What permissions are required to exploit CVE-2026-71504? +

What is a mass assignment vulnerability, and how does CVE-2026-71504 work? +

Why does the User API block this write but the Members API allow it? +

Can the administrator recover access after CVE-2026-71504 is exploited? +

What permissions does an attacker need to exploit CVE-2026-71504? +

Which Dolibarr versions are affected, and how is CVE-2026-71504 fixed? +

More from CodeAnt

VIEW ALL

[GET STARTED]

START PENTEST BOOK A DEMO