BACK

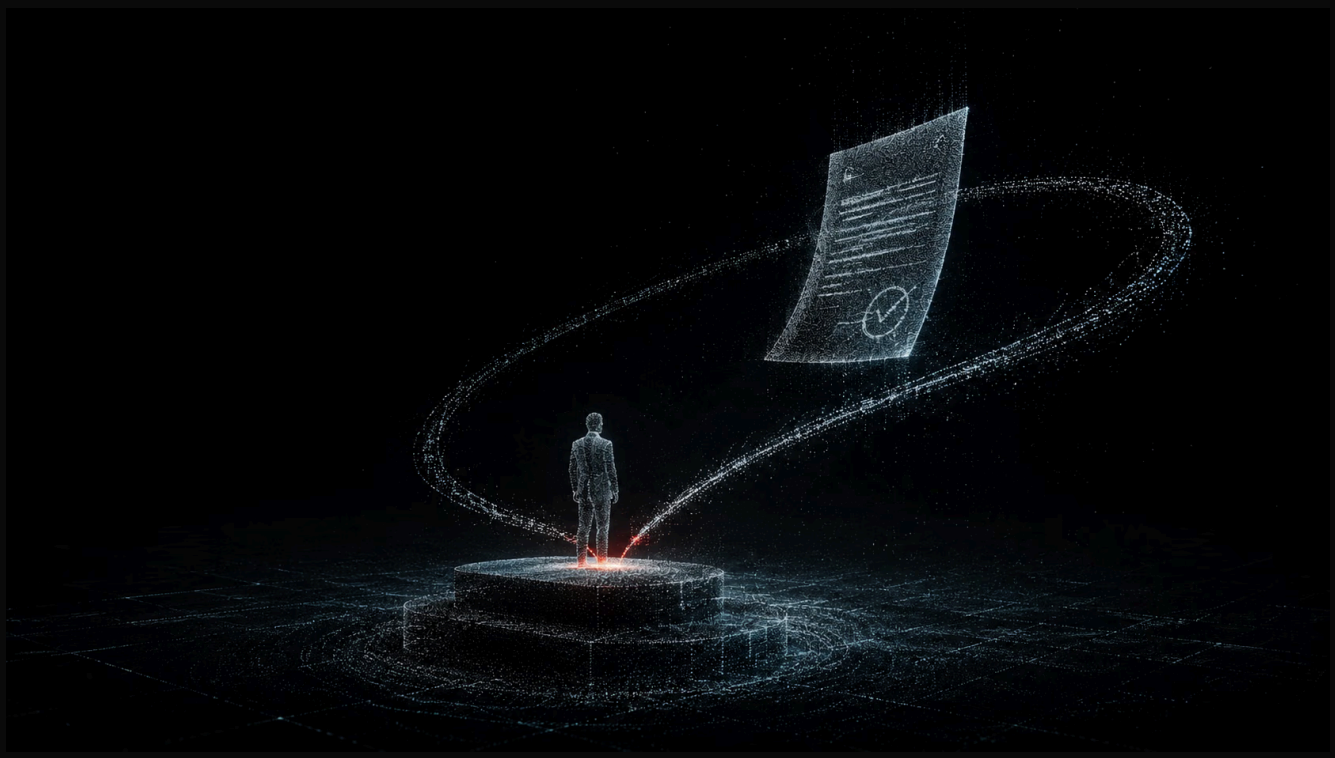
CVE-2026-71509: Dolibarr Mass Assignment Self-Approval Bug

CVE-2026-71509 CVSS 6.5



Amartya Jha

Aug 23, 2026



Research

- The Approve Button Isn't the Only Way to Approve
- Where This Started
- How Approval Is Supposed to Work
- What We Found
  - Affected code
- Proof of Concept
- What This Lets Someone Do
- What This Finding Actually Shows
  - Are You Affected?

BOOK A DEMO

ever holding the approver permission?

That is what CodeAnt AI Security Research found in Dolibarr, the open-source ERP and CRM that runs accounting and expense workflows for small and mid-size businesses.

CVE-2026-71509 is a business-logic and mass-assignment bug in Dolibarr's Expense Reports REST API. Dolibarr has a dedicated approve action, and it correctly checks for the approver permission.

But the same claim can also be edited through a general-purpose route meant for routine changes like fixing a date, and approving a claim, underneath, is just moving it to the approved stage. That general-purpose route never checks who's allowed to make that move.

CodeAnt found the gap by asking whether a plain edit request could reach the same outcome as a permissioned action. It could.

The result: an account that can only file a claim can send a routine edit request that flips the claim to approved and names any approver it likes, including itself, and Dolibarr's dedicated approve button would have refused that same account seconds earlier.

It carries a CVSS score of 6.5.

The fix is live in Dolibarr 24.0.0. If you run an affected build, stop reading and upgrade.

Now here is how we found it, and how it works.

The Approve Button Isn't the Only Way to Approve

A Dolibarr expense claim moves through stages, filed, approved, paid, closed, and there's a dedicated approve action that guards the move to approved. It checks for the approver permission, correctly, every time.

Setting that field to approved is functionally the same thing as clicking the approve button. The edit route just doesn't know that. It sees a field, writes what the request sent, and never asks whether the caller is allowed to approve anything.

The account that pulls this off needs only the everyday right to file an expense claim, the grant almost everyone submitting expenses already has. No approver permission, no finance permission, nothing beyond that.

Send the transition to the dedicated approve action and it comes back 403 Forbidden. Send the identical status change through the edit route and it comes back 200 OK. What that opens up runs further than one claim.

A filer can approve their own claim and write their own name on the approver line, approve a subordinate's claim outright, and push a claim from draft straight to closed with no step in between.

Severity is Medium, CVSS 6.5, with the full vector in the table below. Dolibarr 24.0.0 closes it.

Attribute	Details
CVE	CVE-2026-71509
Affected component	Dolibarr Expense Reports REST API (generic update route)
Issue	Mass Assignment (CWE-915), with Missing Authorization (CWE-862)
CVSS	6.5 Medium (`CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:N/I:H/A:N`)
Required permission	The everyday right to file an expense claim
Fixed version	Dolibarr 24.0.0
Discovered by	CodeAnt AI Code Security Team

Where This Started

One of our researchers kept coming back to ERP platforms, because an ERP is where a company's money workflows actually live, the approvals and sign-offs that decide who gets paid.

Dolibarr's expense workflow keeps the person who files a claim apart from the person who approves it, which is exactly the separation you want around money. It has a proper approve button, and that button checks that you hold the approver permission. That part works.

But the same claim can also be edited through a general-purpose route, the kind you'd use to fix a typo. And approving a claim, underneath, is just moving it to the approved stage.

We weren't just checking whether the approve button was guarded; we were asking whether the same outcome was reachable a different way. So we asked whether that plain edit route would stop us from editing the stage to approved.

It did not. We sent that edit from an account that could only file claims, and the claim came back approved, with our own name recorded as the approver. Then we did it to a teammate's claim, the one the real approve button had just refused us. Approved again.

We kept following that trail, one request at a time, until we landed here.

How Approval Is Supposed to Work

Think of an expense claim as a form that moves through stages: draft, submitted, approved, paid, closed. Moving it forward isn't simply editing a field. Each move is an action, and Dolibarr hands each action to a different person on purpose. One person files the claim. A different person approves it. Finance marks it paid.

Three separate permissions, so the person who spends the money is never the one who signs it off.

Dolibarr backs that up with dedicated actions. There's an approve action and a mark-as-paid action, and each one checks that you actually hold the matching permission. Those are correct.

So editing that one field quietly becomes an approval, and the edit route never asks whether you're allowed to approve anything.

It's the equivalent of an expense form where the approved-by-manager box is just another field you can fill in yourself, and while you're there, you write your manager's name on the approver line too.

What We Found

The edit route runs two checks and then trusts the request body.

```
// htdocs/expensereport/class/api_expenserepo
if (!DolibarrApiAccess::$user->hasRight('expe
    throw new RestException(403);
}
$result = $this->expensereport->fetch($id);
if (!DolibarrApi::_checkAccessToResource('exp
    throw new RestException(403, 'Access not
}
foreach ($request_data as $field => $value) {
    $this->expensereport->$field = $this->_ch
}
if ($this->expensereport->update(DolibarrApiA
```

The first check asks whether you can file claims. The second asks whether this particular claim is yours to touch. Neither asks whether you may perform the approval the body is about to encode.

The loop then copies every field from the request onto the claim, and the only filter it runs rejects internal plumbing, never a workflow field.

Those values land in the database with nothing but an integer conversion in the way. The update routine takes the acting user only to stamp who last modified the row. It never consults that user's rights.

The ownership check, the second gate above, looks like a safety net but isn't one. It only asks whether the claim was filed by you or by someone in your reporting line.

It passes for your own claims and for every claim filed by anyone who reports up to you. That's why the same request that approves your own claim also approves a subordinate's.

A team lead who was deliberately never given the approver permission can still sign off every claim their reports file, which is the exact line that permission exists to draw.

Affected code

- Vulnerable edit route: `htdocs/expensereport/class/api_expensereports.class.php:504`, the generic update handler
- Checks it actually runs: `hasRight('expensereport', 'creer')` (claimant right only, line 506), `_checkAccessToResource()` (ownership only, line 515)
- Check it never runs: any test of whether the caller holds the approve or payment permission before writing `fk_statut`, `fk_user_approve`, or `fk_c_paiement`
- Where the values land unchecked: `htdocs/expensereport/class/expensereport.class.php:625`, `update()`, lines 643-647

Proof of Concept

Every request below ran against a throwaway Dolibarr in local Docker, hardened to a production profile with install locked, production mode on, HTTPS forced, and CSRF at its strictest. None of that hardening touched the outcome.

The actor is a non-admin holding exactly the read and file-a-claim rights, without approve, mark-as-paid, or admin, plus its own API key. The fixture adds a claimant A, a subordinate B who reports to A, and one draft report each.

Control. Prove the dedicated actions refuse this account.

```
PUT  /expensereports/{id}          → 403 (m
```

Exploit. Send the same transition through the generic edit route.

```
PUT /api/index.php/expensereports/{SELF}
{"fk_statut":5,"fk_user_approve":<attacker id
```

The caller has approved its own claim. The same request against the subordinate's report, the one the dedicated approve route just refused, also returns 200, now naming user B as the filer and the attacker as the approver.

Confirm. Read the rows back. Both moved from draft to approved, and both were the only rows the payment workboard selects: EUR 12,000.00 combined, approval date still `NULL`, last-modified stamp holding the real caller's id in both rows.

A final PUT carrying a closed status, an approver id, and a payment mode wrote the closed status straight from draft, with no validation, approval, or payment step in between. Every one of those transitions returned 200, from the same key the approve and setpaid routes had refused minutes earlier.

What This Lets Someone Do

The file-a-claim permission is the everyday grant for anyone who submits expenses, which is close to everyone. This bug turns that routine right into approval authority.

Self-approval removes the one control that keeps a filer from paying themselves, so the approver permission becomes a label rather than a check. The reporting-line loophole extends it from one person to a whole team.

The damage doesn't stop at the claim. A fraudulent claim now arrives at the payment desk already approved, because approved is exactly the status the

That name is whatever the attacker put in the request, so a claim can list anyone as its approver while the edit log still shows who really made the change.

The one reliable tell is the blank approval date. The legitimate path always stamps it. This shortcut never does, so approved claims with no approval date are the signature to hunt for.

What This Finding Actually Shows

CVE-2026-71509 wasn't caused by a missing button-level check. Dolibarr's dedicated approve action does check the right permission, correctly, every time.

The failure was that the same outcome, moving a claim to approved, was also reachable through a general-purpose edit route that had no idea it was performing an approval. It saw a status field like any other field, and wrote whatever the request sent.

We found this not by asking whether the edit returned 200, but by comparing what each route was allowed to do against what the changed record actually meant. Setting a claim to approved is an approval, whoever does it, so it should require the approval permission, wherever in the API that write happens to arrive.

If you want that kind of test run against your own APIs, start with a free CodeAnt pentest.

Are You Affected?

If you run Dolibarr through the affected builds and anyone holds the everyday expense-filing permission, that account can approve claims through the edit route today. Upgrade to Dolibarr 24.0.0.

Dolibarr 24.0.0 brings the generic route under the discipline the dedicated buttons already had. The workflow fields are refused on the edit route rather than written through it.

The broader lesson outlives this one endpoint. When an object has a status field and a set of permissioned actions that move it between statuses, a general-purpose edit that can reach that field is quietly performing those actions without asking who may.

If a general route can write a workflow status, test that field like an action, not like ordinary data.

Related research. This finding is one node in the split-brain authorization pattern, the pillar behind all nine findings. It rides the same blind mass-assignment shape as CVE-2026-71504, where the same unfiltered write loop in the Members API sets a linked user's password and reaches administrator. Here the fields written are workflow states rather than credentials, and the result is a filer approving their own claim, and their whole team's.

[FAQ]

Frequently Asked Questions

What is CVE-2026-71509? +

Can CVE-2026-71509 let a user approve their own expense claim? +

What is a business logic vulnerability, and how does CVE-2026-71509 illustrate it? +

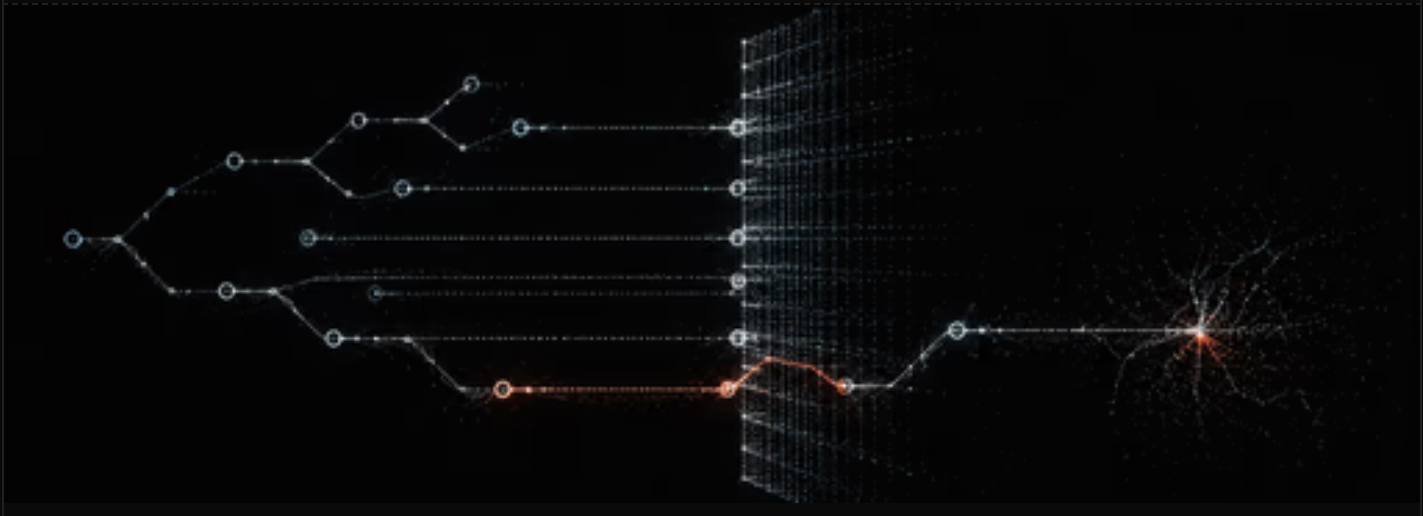
What is the difference between the dedicated approve route and the generic PUT here? +

What privileges does an attacker need to exploit CVE-2026-71509? +

Which Dolibarr versions are affected, and how is CVE-2026-71509 fixed? +

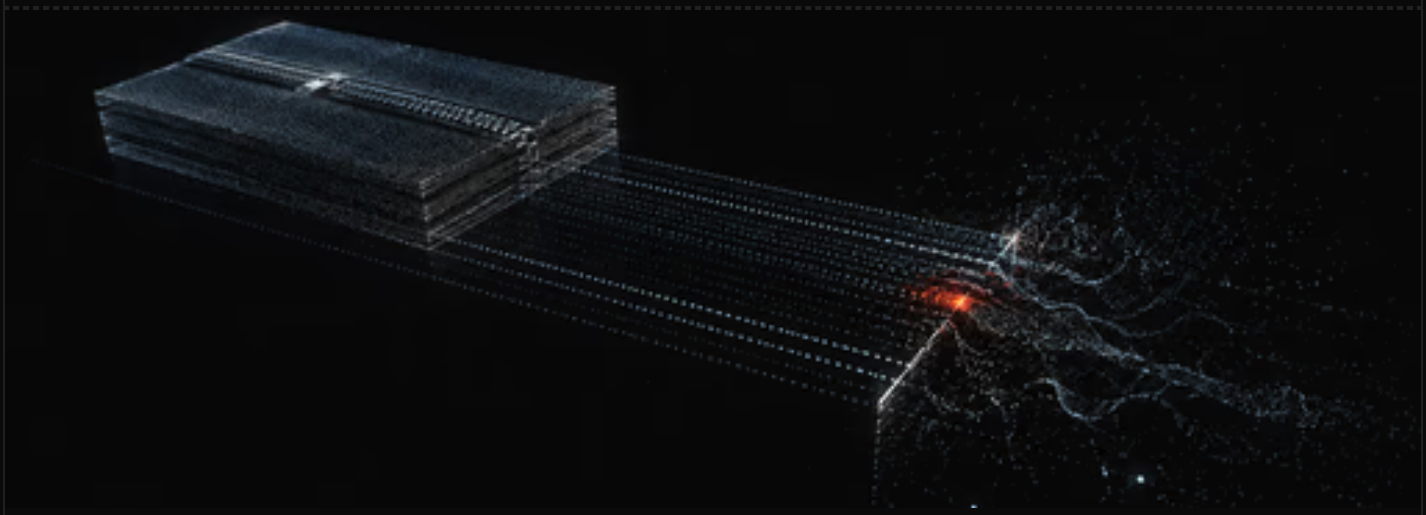
More from CodeAnt

VIEW ALL



Mar 3, 2026

CVE-2026-29000: pac4j-jwt Auth Bypass PoC With a Public Key



Mar 9, 2026

CVE-2026-28292: simple-git RCE via Case-Sensitivity Bypass

CVE-2026-28292 CVSS 9.8



Mar 11, 2026

CodeAnt AI Product Resources Security Reseach Customer Enterprise

[GET STARTED]

Find out what's already exploitable in your codebase.

START PENTEST BOOK A DEMO

355 Bryant St, San
Francisco, CA 94107

Copyright © 2026 CodeAnt AI. All rights reserved.