BACK

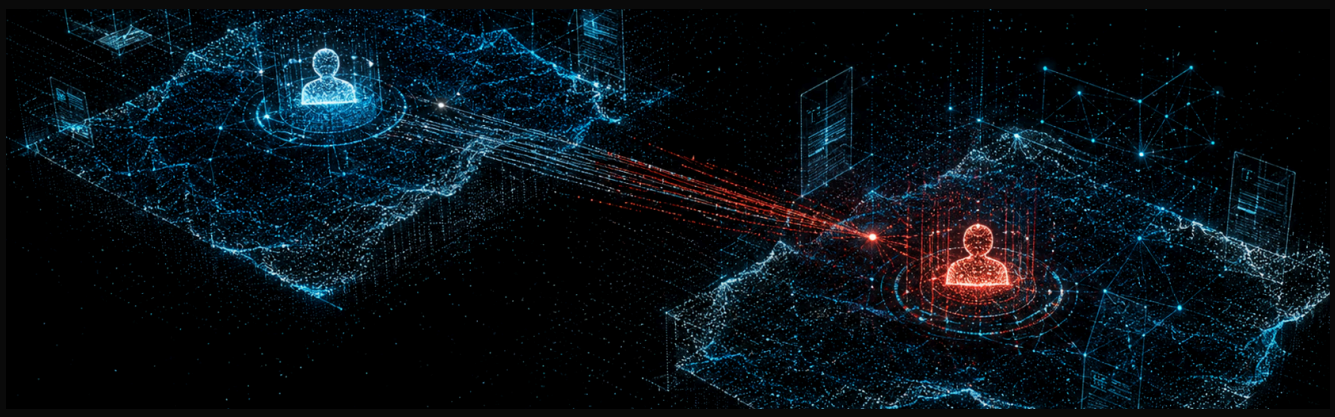
CVE-2026-71505: Dolibarr BOLA Enables Portal Account Takeover

CVE-2026-71505 CVSS 8.1



Amartya Jha

Aug 23, 2026



What if the account you handed your sales team to onboard new clients could quietly reset the customer-portal password of any company in your system, including the companies it is not even allowed to read?

That is what CodeAnt AI Security Research found in Dolibarr, the open-source ERP and CRM that thousands of small and mid-size businesses run their operations on.

CVE-2026-71505 is a broken object level authorization bug, a textbook BOLA filed as CWE-639, in Dolibarr's Third Parties REST API.

A low-privilege key holding nothing but the create-companies right can overwrite another company's customer-portal password, sign in as that company, and read its invoices. No admin rights, no CSRF token, no action from the victim.

It carries a CVSS score of 8.1, High.

Dolibarr is maintained largely by volunteers, with no funded security function and one small commercial sponsor behind it. That team had this shut in the 24.0.0 release before a word of it went public, so installs that upgraded were never exposed to public disclosure. If you run an affected build, stop reading and upgrade.

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

Where the Check Went Missing

Dolibarr's Third Parties API has two routes that touch the same rows. The one that reads a company's customer-portal accounts asks two questions before it answers: can this caller read companies at all, and can this caller read this particular company.

The route that sets a portal password asks only the first. It confirms the role and goes straight to the database, so the second question never gets asked on the write path. That gap means any company number written into the URL is a company the caller gets to change. Permission to see it was never part of the requirement.

The account that pulls this off needs nothing more than the create-companies right, the grant you hand a salesperson onboarding a new client, a partner account, or an integration syncing companies out of a CRM. The asymmetry shows up plainly in the responses. Reading the victim company comes back `403 Forbidden`. Writing a new portal password to that same company comes back `200 OK` one request later.

From there the caller sets the password, signs into the customer portal as that company, and reads its invoices. No CSRF token to forge, no reset email to intercept, no cracking, and nothing the victim has to do. Severity lands at high, CVSS 8.1, with the full vector in the table below. Dolibarr 24.0.0 closes it.

Attribute	Details
CVE	CVE-2026-71505
Affected component	Dolibarr Third Parties REST API (`putSocieteAccount()`)
Vulnerability	Broken Object Level Authorization (CWE-639)
CVSS	8.1 High
Required permission	The create-companies right only
Impact	Customer-portal account takeover, cross-company invoice read, stored password hash disclosure
Fixed version	Dolibarr 24.0.0
Discovered by	CodeAnt AI Code Security Team

Where This Started

At CodeAnt AI, we keep chasing one question: which open-source packages actually run real businesses, and what critical bugs are hiding inside them?

One of our researchers kept coming back to ERP platforms, because an ERP holds a company's customers, its money, and the portal those customers log into, all in one place. Dolibarr is exactly that kind of platform, an open-source ERP and CRM that organizations around the world use to run invoicing, accounting, and a customer-facing portal.

It began as a routine read of the REST API. Then one detail caught our eye. The route that reads a company's customer-portal accounts ran two separate authorization checks, back to back. One asked whether the caller was allowed to read companies at all. The second asked whether the caller was allowed to read this specific company. That second check is the careful one.

So we could not put down the next question. If the read route checks ownership this carefully, does every route that touches the same rows do the same? We started pulling on the sibling routes, one at a time. Most of them held. Then we reached the route that sets a company's portal password. It ran the first check and skipped the second.

We pointed it at a company our key was not allowed to read. The read came back 403. The write came back 200. Then we logged into that company's portal with the password we had just set, and its invoices loaded on the screen.

That is where we landed.

How the Endpoint Is Supposed to Check Ownership

Think of two questions any request to change a company has to pass. The first is a role question: are you the kind of user who edits companies at all? The second is an object question: is this the particular company you are allowed to edit?

It is the difference between a building keycard and an office door. The keycard proves you belong in the building. The door still has to check that this specific office is yours to enter. A right is the keycard. The per-company ownership check is the office door.

The read route for portal accounts checks both, the keycard and the door. That is why it returns 403 when a key asks for a company it does not own. The write route that sets a portal password checks the keycard and never checks the door. So any company number you write into the URL is the company you get to change.

What We Found

The read route runs both stages, one after the other:

The ownership check on line 2241 is the whole ballgame. The write route, `putSocieteAccount()`, lives about three hundred lines down in the same class, and it opens with the role check alone, then goes straight to the database:

The lookup is keyed on the company number from the URL, with no ownership reconciliation. And it gets worse at the assignment step, where the request body is copied onto the object field by field with no allowlist, so a password field in the body rides straight through:

The shared update routine then spots the plaintext password and hashes it into the real verifier the portal login checks against. So a missing ownership check, plus an unfiltered copy of the request body, together turn create-companies into set-any-customer's-portal-password.

Affected code

- Vulnerable route: `htdocs/societe/class/api_thirdparties.class.php`, `putSocieteAccount()` (lines 2527-2533), the portal-account write handler
- Check it runs: `hasRight('societe', 'creer')`, the role check only
- Check it never runs: `DolibarrApi::_checkAccessToResource('societe', $id)`, the per-company ownership check that the sibling read route, `getSocieteAccounts()` (line 2241), does run
- Where the request body lands unfiltered: the field-assignment loop at lines 2550-2561, which copies any field, including a plaintext password, straight onto the account object
- Sibling routes carrying the same omission, also fixed in 24.0.0: create, replace-or-create, and both delete paths for portal accounts

The 403 and the 200, One Request Apart

Every request below hit a Dolibarr container on our own laptop, a develop-branch build at commit `ab7e6040`, hardened to a production profile with install locked, production mode on, HTTPS forced, and CSRF at its strictest, seeded with fixture companies we created. No live deployment and no real customer's portal was ever touched.

The attacking principal is a non-admin key carrying exactly two rights, read-companies and create-companies.

Control. Prove the same key is refused a plain read.

Both come back:

This user may not see company 2.

Exploit. Change only the method, and hand it a new password.

The write succeeds against the exact object the same key was refused a request earlier. The 200 body even hands back the account login and its pre-update stored verifier, a hash the caller had no read access to, because the routine serializes the object before the new hash lands on it.

Confirm. Verify the takeover, then walk through the portal. Reading the row afterward confirms the verifier changed and that the last-modified stamp now records user id 8, the attacker. A verify of the attacker's chosen password against the new hash returns true, and the victim's original returns false.

The takeover then needs no cracking and no victim action. Log into the WebPortal as that company with the password just set, request the invoice list, and the portal renders `DOL02-CONFIDENTIAL-260804`, the marker stamped onto the victim's invoice, at 100 EUR, status Paid.

The invoice controller (`htdocs/webportal/controllers/invoicelist.controller.class.php`, lines 77-85) filters on the logged-in company's id, so the session sees exactly that company's books.

Who Holds This Right, and What They Walk Away With

The create-companies right (`llx_rights_def.id = 122`) is a routine grant. In a real deployment it is the salesperson onboarding a new client, the partner account wiring up a customer, or the integration user syncing companies from a CRM.

None of those roles was ever meant to imply the power to read and rewrite the credentials of every other company in the tenant. One junior sales login, or one API key baked into a middling integration, is the whole prerequisite.

What that principal walks away with is a full customer-portal account takeover, with no interaction from the customer, no CSRF token, and no reset flow to intercept.

They just set the password. With the session in hand they read another company's commercial records, and the same 200 response leaks stored password hashes for accounts they cannot read.

Be clear about the ceiling. Those leaked hashes are disclosed, not cracked, so they are offline-cracking and credential-reuse candidates, an aggravating factor rather than the basis of the confidentiality rating, which rests on the portal invoice read we demonstrated.

The same unguarded path also forges and deletes portal accounts on other people's companies, and because Dolibarr ships no self-service portal reset and fires no notification on the change, a locked-out customer has no signal and no recovery short of a back-office admin.

What This Finding Actually Shows

CVE-2026-71505 wasn't caused by a missing check on the read side. The route that reads a company's portal accounts checks ownership correctly, every time.

The failure was that the sibling write route never repeated it. It checked the role, went straight to the database, and copied the request body onto the record with no ownership check and no field allowlist in the way.

The lesson generalizes past this one function. When two routes write the same rows and each author has to remember to re-type the ownership check, one of them eventually will not.

The durable fix is structural: resolve and authorize the owning company once, in the API base class, for every nested company route before the handler runs, so a brand-new write route fails closed by default.

And a credential change should be its own permissioned operation, not a side effect a generic copy-the-body loop can trip into. Reject the stored password hash as an input, and never serialize it as an output.

We found this by testing what the attacker could read against what they could modify, then validating the takeover end to end, from the refused read to the invoice on the screen.

If you want to see where your own APIs disagree with themselves, start with a free CodeAnt pentest.

Are You Affected?

If you run Dolibarr through the affected builds and any account holds the create-companies right, that account can take over any customer's portal today. Upgrade to Dolibarr 24.0.0.

The 24.0.0 patch is the missing line put back. The ownership check now runs immediately after the role check in the write route, matching the read route, and the same guard was added to the four sibling write routes, create, replace-or-create, and both delete paths, that shared the omission.

Related research. This finding is one node in the split-brain authorization pattern, the pillar that walks through why the same read-guarded, write-unguarded asymmetry recurs across Dolibarr's REST surface. For the sibling that follows the identical missing ownership check on a Third Parties write route but ends with money rather than data, see CVE-2026-71507, where the unchecked write redirects an outbound SEPA payment file.

[FAQ]

Frequently Asked Questions

What is CVE-2026-71505? +

What privileges are required to exploit CVE-2026-71505? +

Which Dolibarr versions are affected, and how is CVE-2026-71505 fixed? +

What is the difference between IDOR and BOLA in this finding? +

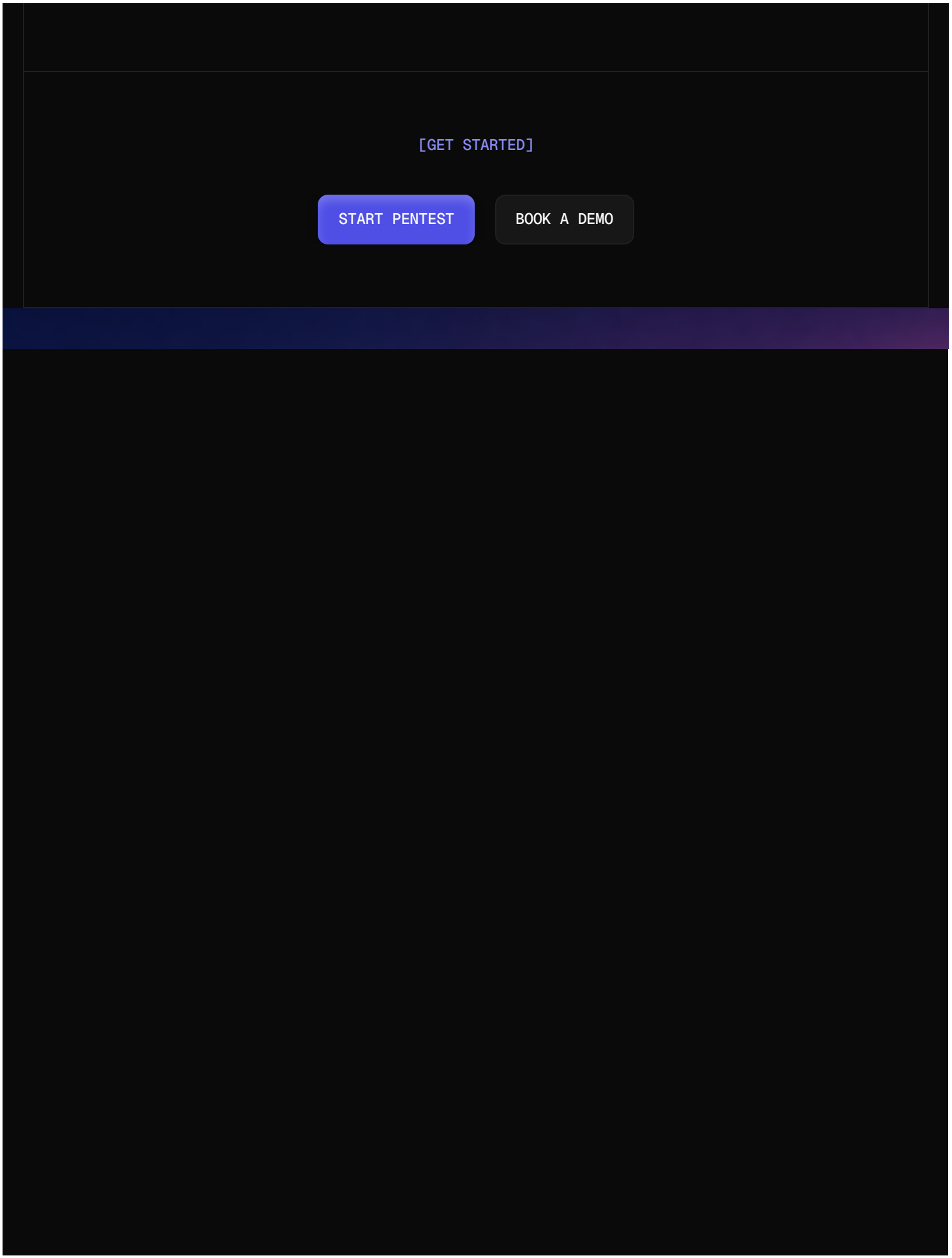
What is a BOLA vulnerability, and how does CVE-2026-71505 illustrate it? +

Are the password hashes leaked by CVE-2026-71505 crackable? +

What privileges does an attacker need to exploit CVE-2026-71505? +

More from CodeAnt

VIEW ALL


[GET STARTED]
START PENTEST
BOOK A DEMO