BACK

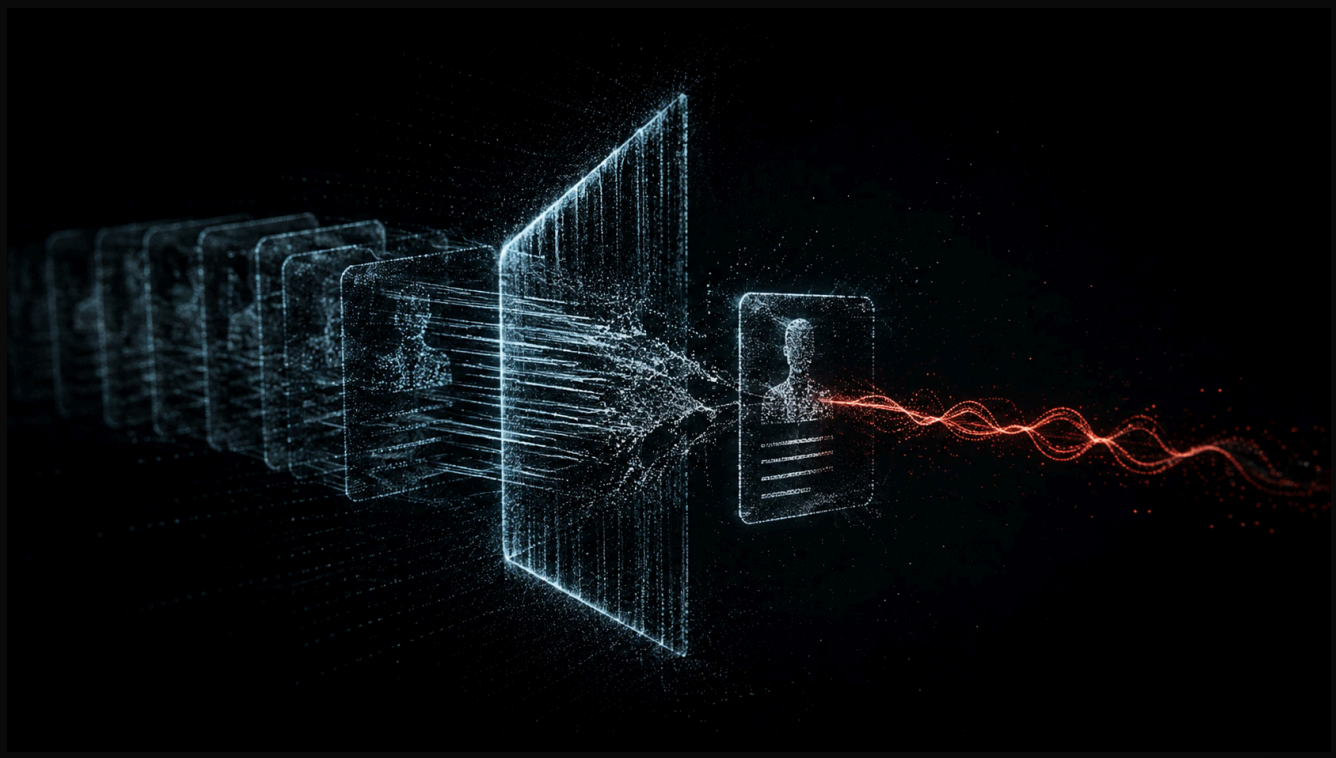
CVE-2026-71511: Dolibarr Members API Exposes Password Hashes

CVE-2026-71511 CVSS 6.5



Amartya Jha

Aug 23, 2026



Research

- The Cleaner Stopped One Field Short
- Where This Started
- How Passwords Are Supposed to Leave the API
- What We Found
  - Affected code
- Proof of Concept
- What Someone Can Actually Do With It
- What This Finding Actually Shows
  - Are You Affected?

BOOK A DEMO

from an API that is supposed to strip passwords out?

That is what CodeAnt AI Security Research found in Dolibarr, the open-source ERP and CRM that clubs, associations, nonprofits, and small businesses use to run their membership and back office.

CVE-2026-71511 is an insufficiently-protected-credentials bug in Dolibarr's Members REST API. Any key that can only read members gets each member's live password hash back in the response, from both the single-member lookup and the bulk member list.

The read is completely legitimate. The problem is what the response contains.

It carries a CVSS score of 6.5. This is a disclosure bug, not an account takeover. What leaks is a scrambled password hash, not a plaintext password, and it still has to be cracked offline before it becomes a login.

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 Cleaner Stopped One Field Short

The Members API is supposed to strip the password fields out of every member record it hands back, the same way the User API does. It strips two of the three. The one it misses is the box holding the real, live, scrambled password. It rides out next to the member's login and email in an otherwise ordinary, fully authorized read.

The key that gets it holds nothing but the read-members right, the grant you'd hand a membership secretary, a front-desk clerk, a reporting dashboard, or a CRM sync job.

The contrast sits one API over. The User API blacks out the entire password family for everyone, administrators included. The Members API, serving a near-identical record, lets one of those same fields through.

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

Attribute	Details
CVE	CVE-2026-71511
Affected component	Dolibarr Members REST API (member read responses)
Issue	Insufficiently Protected Credentials (CWE-522)
CVSS	6.5 Medium (`CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N`)
Required permission	The read-members right (`adherent.lire`)
What leaks	The member's stored bcrypt password hash
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 not just a company's data but its people's credentials, and how carefully those are handled tells you a lot.

Dolibarr is exactly that kind of platform, an open-source ERP and CRM that organizations around the world use to run accounting, membership, and operations.

Dolibarr's User API is strict about passwords. When it returns a user, it blacks out the whole password family before the response goes out, and it does that for everyone, administrators included. It even refuses to let anyone write those

The Members API serves up a very similar record, a member with a login and a stored password. So we asked whether it followed the same rule. Does the Members API strip the password fields the way the User API does?

It strips two of the three. The third one, the box that holds the real scrambled password, sails straight through. So we pointed a key that could do nothing but read members at a single member, and the response came back with that member's password hash sitting right next to their login and email.

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

How Passwords Are Supposed to Leave the API

Imagine a member's record as a form that happens to store the password in three different boxes. One is empty, one holds an old format, and one holds the real scrambled password. Before the API hands that form to anyone, a cleaner is supposed to black out the password boxes.

The User side of Dolibarr does this thoroughly. It blacks out the whole password family for everyone, even administrators, and it refuses to let anyone write those boxes either. That is the policy, written by the project, in the project's own words.

The Members side runs a cleaner too, and the permission check in front of it is correct. You genuinely need the read-members right to reach a member at all. But the Members cleaner's redaction list names only two of the three boxes.

The third, the one holding the real scrambled password, goes out uncensored. So the disagreement here is not about who is allowed to read a member. It is about what a legitimate read is allowed to contain, and on this path, nobody remembered to redact the one box that mattered.

What We Found

The hash is loaded early and sits on the object as a plain, readable field long before any response is built. By the time a route reaches the cleaner, the live verifier is already there waiting.

```
// htdocs/adherents/class/api_members.class.p
$object = parent::_cleanObjectDatas($object);
if ($object instanceof Adherent) {
    unset($object->subscriptions);
    // ...
}
```

And the shared base cleaner it hands off to stops one line too early:

```
// htdocs/api/class/api.class.php:299-310
unset($object->pass);                // line 30
unset($object->pass_indatabase);     // line 30
// pass_indatabase_crypted, pass_crypted, pas
```

Two of the three password fields are removed. The one holding the real hash survives.

Both routes that reach this cleaner authorize correctly first: the single lookup and the bulk list each check the read-members right before returning anything.

Four alternate member lookups all funnel through the same cleaner, so they leak the same way.

Affected code

- Vulnerable path: `htdocs/adherents/class/api_members.class.php:554-573`, the Members API's `_cleanObjectDatas()`, which delegates password redaction to the shared base cleaner instead of repeating it
- Where the redaction actually falls short: `htdocs/api/class/api.class.php:299-310`, the shared base cleaner, which removes `pass` and `pass_indatabase` but leaves

every caller, including administrators, which is the rule this path was supposed to inherit and did not

Proof of Concept

Everything below ran against a self-contained Dolibarr in local Docker, seeded with fixtures we created. No external or shared system was in the loop at any point.

The instance ran a hardened production profile, install locked, production mode on, HTTPS forced, and CSRF at its strictest, and none of that hardening touched this.

We provisioned one non-administrator user whose entire effective right set was a single row, the read-members right. No group membership, admin off, account enabled.

We gave a fixture member a known password through the ordinary admin path, so the stored password column held a fresh bcrypt string we controlled, and the plaintext column stayed empty, which meant the hash was the only credential material the member row carried.

Control. Confirm the right is actually the gate. An anonymous request returned `401`, so authentication is required. A second valid key that authenticated but lacked the read-members right was refused with `403`, confirming that one right is the only gate.

Exploit. Read a member with the read-only key.

```
GET /api/index.php/members/1
DOLAPIKEY: <reader key, read-members only>

{ "id": "1", "login": "audit_victim",
  "pass_indatabase_crypted": "2y10$N0hvUclTv5
  "email": "audit_member@audit.invalid" }
```

Confirm. Prove the hash is real, and that it isn't a one-off. We compared the string byte for byte against the stored password column, identical, and ran a verify against it in PHP, which returned true for the member's real current password and false for a wrong one.

It is a working verifier, retrieved with a read-only key. Asking the bulk list route returned the same field per member, so on a populated deployment that is one hash per member, per page.

What Someone Can Actually Do With It

The read-members right is not a privileged grant. It is what you give a membership secretary, a front-desk clerk, a reporting dashboard, or a CRM sync job, anyone who needs to list members but should never touch a password.

Nothing in the right's name or description hints at credential access, so an administrator handing it out has no reason to think they are also handing out the member password store.

Once the hashes are off the server, every defense that protects the live login is gone. There is no rate limit, no lockout, no failed-attempt counter between an attacker and an offline cracking rig.

Bcrypt raises the cost of each guess, so this is not an instant break, but weak, common, and reused passwords fall to an offline wordlist run regardless, and the attacker gets to try forever, quietly, with no live system to notice.

Every member who reused that password somewhere else is now exposed there too, with their leaked login and email as ready-made identifiers.

There is also a quieter escalation, which we confirmed by reading the source but did not run. Dolibarr can provision a back-office staff user from a member and copies the member's password hash across when it does.

Where a staff login was created that way and neither password has changed since, the leaked member hash is also that staff account's hash, an account that may carry far broader rights than the member ever had.

after the fact.

What This Finding Actually Shows

CVE-2026-71511 wasn't caused by a missing permission check. Both routes that expose this field authorize correctly, checking the read-members right before returning anything.

The failure was that the redaction list one cleaner needed didn't get repeated in the one that mattered. The Members API delegated to a shared base cleaner and trusted it to finish the job, and the shared cleaner stopped one field short of the User API's own rule.

The lesson this whole class keeps teaching is that a redaction list living inside one cleaner is a rule only that cleaner obeys. The next API class that serializes the same record starts from zero and has to re-earn every omission by hand, and eventually one gets missed.

The durable fix is to make the safe thing structural: put the credential stripping in the shared base cleaner so no subclass can forget it, mark the credential fields on the model so they never reach a serializer at all, and add a test that walks every REST class and fails the build if any password field survives its cleaner.

Authorization does not end when access is granted. What a response contains is part of the security boundary too, so every endpoint that returns a shared user or member object should deny credentials, tokens, and secrets by default.

We found this not by asking whether the request was allowed, but by checking what an allowed request actually returned.

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

Are You Affected?

Upgrade to Dolibarr 24.0.0. Treat any member passwords exposed by a vulnerable build as compromised, and revoke the read-members right from any key that does not strictly need it.

Dolibarr 24.0.0 extends the redaction so the whole password family is stripped before a member response goes out, bringing the Members API in line with the never-return-passwords rule the User API already enforced.

Related research. This finding is one node in the split-brain authorization pattern, the pillar behind all nine findings, two doors into the same rows that disagree about the rules. Its mirror image lives in this very same Members API. Where this bug lets a low-privilege caller read a credential out, CVE-2026-71504 lets one write an admin credential in through mass assignment. Same file, opposite direction.

[FAQ]

Frequently Asked Questions

What is CVE-2026-71511, and what does it leak? +

Can the bcrypt hash leaked by CVE-2026-71511 be used to log in directly? +

How does CVE-2026-71511 expose a back-office staff account? +

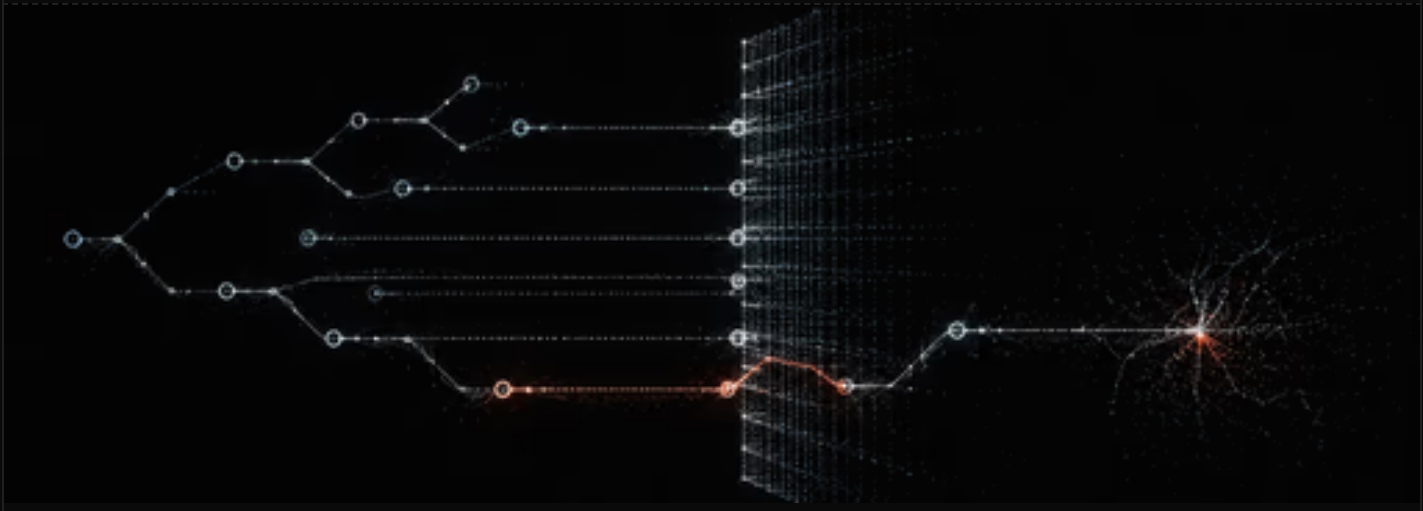
Which Dolibarr versions are affected, and how is CVE-2026-71511 fixed? +

What data can be exposed through CVE-2026-71511? +

Does CVE-2026-71511 require administrator privileges? +

More from CodeAnt

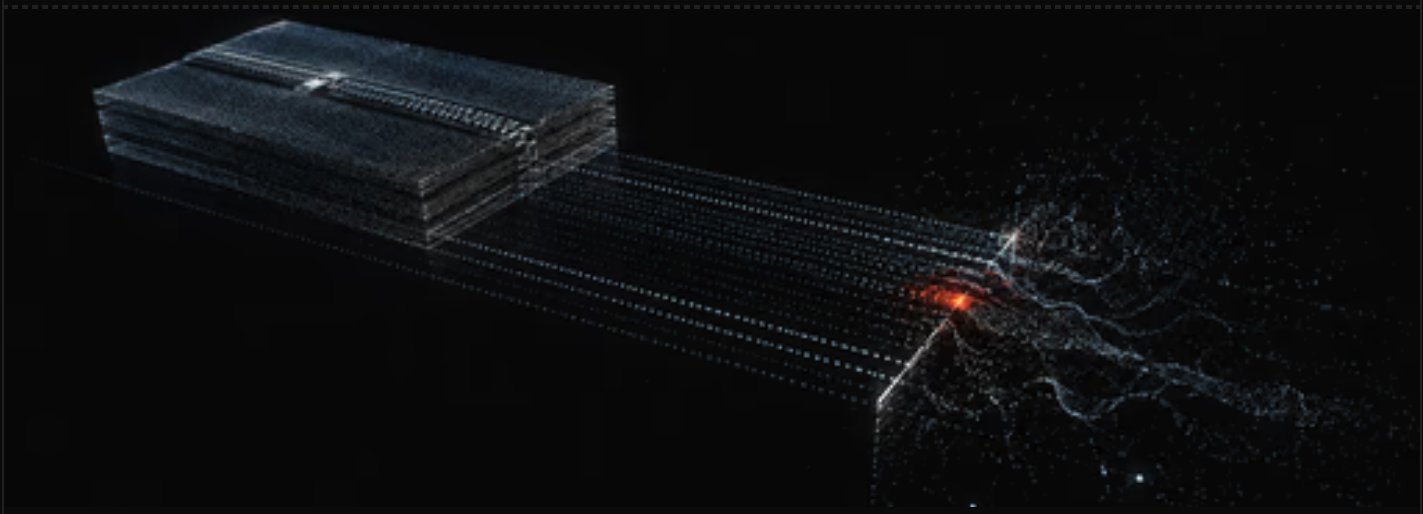
VIEW ALL



Mar 3, 2026

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

CVE-2026-29000 CVSS 10



Mar 9, 2026

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

CVE-2026-28292 CVSS 9.8



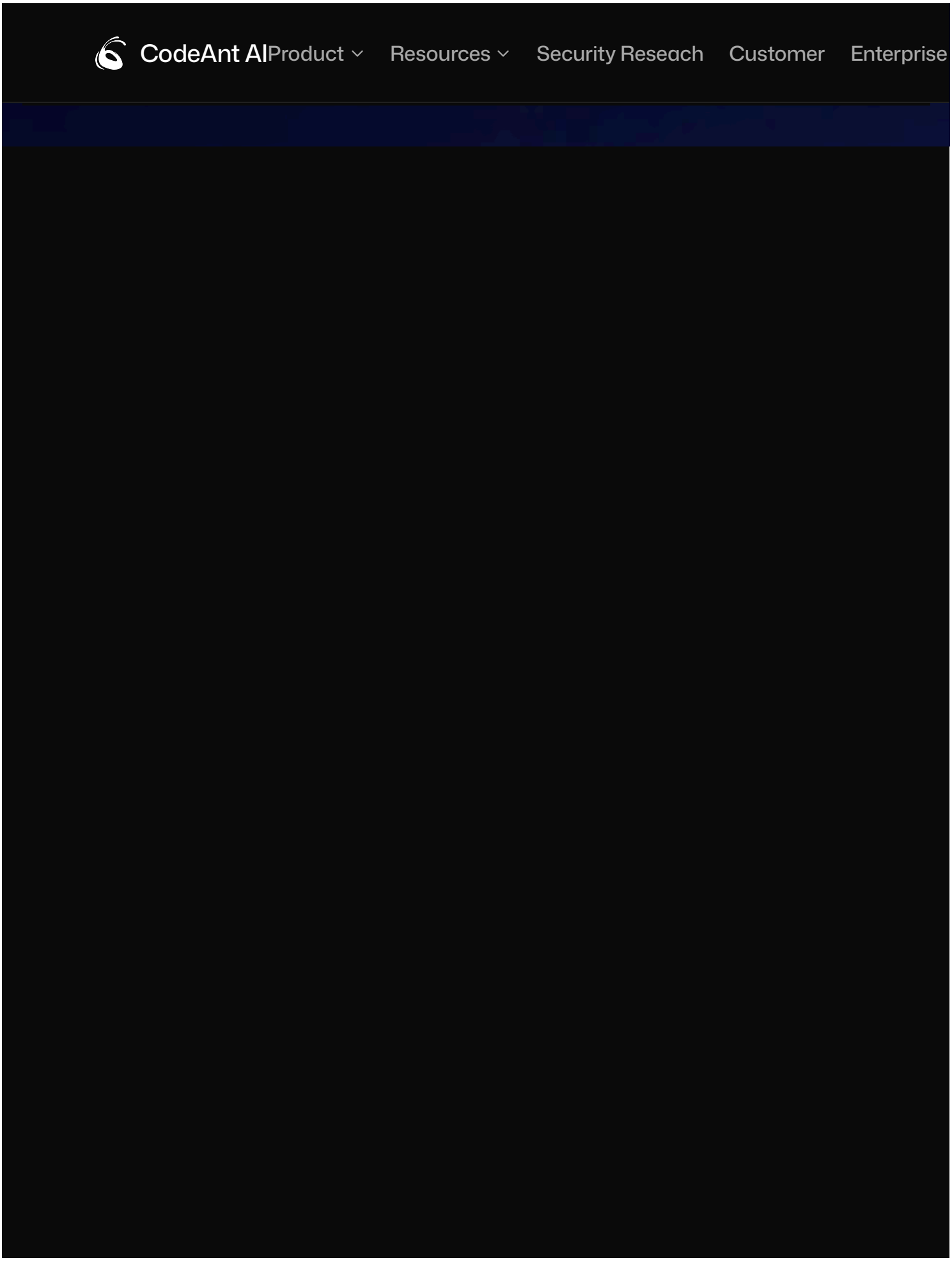
Mar 11, 2026

CVE-2026-31988: jszip DoS via Malformed ZIP File

[GET STARTED]

Find out what's already exploitable in your codebase.

START PENTEST BOOK A DEMO


CodeAnt AI
Product
Resources
Security Reseach
Customer
Enterprise