

Security Advisory
Fider

Created by: Viktor Chuchurski
06/02/2026

© DOYENSEC

Overview

This document summarizes the results of a vulnerability research activity aimed at discovering authorization implementation issue. While security testing was not meant to be comprehensive in terms of attack and code coverage, we have identified onevulnerability that could lead to unauthorized cross-tenant data access.

About Us

Doyensec is an independent security research and development company focused on vulnerability discovery and remediation. We work at the intersection of software development and offensive engineering to help companies craft secure code.

Research is one of our founding principles and we invest heavily in it. By discovering new vulnerabilities and attack techniques, we constantly improve our capabilities and contribute to secure the applications we all use.

Copyright 2026. Doyensec LLC. All rights reserved.

Permission is hereby granted for the redistribution of this advisory, provided that it is not altered except by reformatting it, and that due credit is given. Permission is explicitly given for insertion in vulnerability databases and similar, provided that due credit is given. The information in the advisory is believed to be accurate at the time of publishing based on currently available information, and it is provided as-is, as a free service to the community by Doyensec LLC. There are no warranties with regard to this information, and Doyensec LLC does not accept any liability for any direct, indirect, or consequential loss or damage arising from use of, or reliance on, this information.

Cross-Tenant Invitation Acceptance	
Vendor	Fider
Severity	**Critical**
Vulnerability Class	Insecure Design
Component	https://github.com/getfider/fider/blob/main/app/services/sqlstore/postgres/tenant.go#L129
Status	Closed
Credits	Viktor Chuchurski

Summary

Fider allows its users to create a multi-tenant setup. Each tenant will be logically separated and accessible on a separate subdomain. While reviewing the application's implementation Doyensec engineers identified an issue in how invitation tokens are handled. When a user gets invited to a tenant, the application will send an email containing an invitation token. Using that token the user can accept the invitation, create a profile and gain access to the tenant.

When accepting the invitation, the application will check its validity using the following code:

```
func getVerificationByKey(ctx context.Context, q *query.GetVerificationByKey)
error {
        return using(ctx, func(trx *dbx.Trx, tenant *entity.Tenant, user
*entity.User) error {
                verification := dbEntities.EmailVerification{}

                query := "SELECT id, email, name, key, code, created_at,
verified_at, expires_at, kind, user_id, attempts FROM email_verifications WHERE key
= $1 AND kind = $2 LIMIT 1"
                err := trx.Get(&verification, query, q.Key, q.Kind)
                if err != nil {
                        return errors.Wrap(err, "failed to get email verification
by its key")
                }

                q.Result = verification.ToModel()
                return nil
        })
}
```

The `getVerificationByKey` function fails to check whether the token is associated with the current tenant represented by the `tenant *entity.Tenant` object. As a result, invites for tenant A can be accepted on tenant B allowing access to that tenant.

Technical Description

The reproduction steps outlined below require a working multi-tenant Fider setup. A local deployment can be configured by following these steps:

1. Create a `.env` file with the following contents:

```
GO_ENV=development
HOST_MODE=multi
HOST_DOMAIN=fider.local
DATABASE_URL=postgres://fider:fider_pw@localhost:5555/fider?sslmode=disable
EMAIL_SMTP_HOST=localhost
EMAIL_SMTP_PORT=1025
EMAIL_NOREPLY=noreply@fider.local
JWT_SECRET=super-secret-123
```

2. Configure hostnames for the tenants in the `/etc/hosts` file:

```
127.0.0.1    login.fider.local
127.0.0.1    alice.fider.local
127.0.0.1    bob.fider.local
```

3. Run the Docker containers using `docker compose up -d`
4. Run the application:

```
make migrate
make watch
```

5. Verify that the application is running correct by navigating to `http://login.fider.local:3000/signup`
6. Using the UI rendered on the page from step 5), create two tenants. The reproduction steps assume that the admin users for their respective tenants have the emails `admin@alice.io` and `admin@bob.io`
7. Log in with both admin users and from the top-right navigate to "Site Settings" > "Private" > enable "Private Board"
8. Navigate to both tenants' domains and verify there's no option to sign up a new account

With the setup completed, we take the following steps to reproduce the issue:

1. [Alice] Log in using `admin@alice.io`
2. [Alice] From the top-right navigate to "Site Settings" > "Invitation"
3. [Alice] Invite a new user `user1@alice.io`
4. [Alice] Navigate to the MailHog instance (`http://localhost:8025`) and verify that the newly invited user received an invitation token. The link has the following format:

```
http://alice.fider.local:3000/invite/verify?k=<TOKEN>
```

5. [Alice] Copy the link and modify it to point to Bob's tenant's domain:
6. [Alice] Navigate to the modified link
7. [Alice] Verify that token was accepted and allowed for a new account to be created
8. [Alice] Verify that you are now able to access Bob's tenant

9. [Bob] Log in using `admin@bob.io`
10. [Bob] From the top-right navigate to "Site Settings" > "Users"
11. [Bob] Verify that the user `user1@alice.io` is shown on the list of active users

Impact

High. If exploited, this issue will allow users access to other tenants configured on a particular deployment of the application. This in turn leads to unauthorized access and modification of data belonging to the victim tenant. The impact is somewhat lowered by the fact that users cannot be invited with higher privileges, restricting the access to the level of a "Member" level user.

Complexity

Low. Finding the issue is trivial given the open-source nature of the application. However, successful exploitation requires attackers to have access to, or the ability to generate invitation tokens. The former will require someone to invite the attacker. The later would require the attacker to be an administrator of their own tenant. For on-prem deployments, this should require the signup to be publically accessible (`SIGNUP_DISABLED=false`). Fider's SaaS offering (`https://login.fider.io/signup`) does have tenant signups enabled, making exploitation possible.

It is worth noting that `SIGNUP_DISABLED` is disabled by default[1], increasing the possibility of misconfigurations.

Remediation

Anchor the invite token verification to the tenant. Make sure that the token being validated is associated with the tenant where it's being redeemed.

One potential solution is to update the verification query to also check for the tenant ID:

```
query := "SELECT id, /* ---snip--- */, attempts FROM email_verifications WHERE key
= $1 AND kind = $2 AND tenant_id = $3 LIMIT 1"
err := trx.Get(&verification, query, q.Key, q.Kind, tenant.ID)
```

Any deviation from that requirement should result in the request being fully rejected.

Disclosure Timeline

06/02/2026	Issue responsibly disclosed to Fider maintainers
06/02/2026	A patch was introduced with PR#1561
07/01/2026	Advisory published

1. https://github.com/getfider/fider/blob/main/app/pkg/env/env.go#L39