

Security Advisory
Fider

Created by: Viktor Chuchurski
06/11/2026

© 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.

SSRF via OAuth Profile URL	
Vendor	Fider
Severity	**High**
Vulnerability Class	Server-Side Request Forgery (SSRF)
Component	app/services/oauth/oauth.go#L345
Status	Closed
Credits	Viktor Chuchurski

Summary

A Server Side Request Forgery (SSRF) attack describes the ability of an attacker to create network connections from a vulnerable web application to the internal network and other Internet hosts. Frequently, an SSRF vulnerability is used to attack internal services located behind a firewall and not directly accessible from the Internet.

In the Fider application, the custom OAuth provider configuration functionality can be leveraged to initiate an HTTP(S) connection targeting internal hosts.

More specifically, the user-configurable `ProfileURL` value is not properly validated. This allows malicious tenant administrators to specify URLs resolving to internal IP addresses. Later, when an OAuth flow is initiated, an HTTP `GET` request will be made to the internal host. The full response of the request can be then obtained by the administrator by using the OAuth testing functionality.

Technical Description

The reproduction steps outlined below require a working 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_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
```

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 correctly by navigating to `http://login.fider.local:3000/signup`
6. Using the UI rendered on the page from step 5), create a new tenant with the admin user `admin@alice.io`
7. Log in with the admin user and verify that the tenant is correctly set up

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

1. Log in using `admin@alice.io`
2. From the top-right navigate to "Site Settings" > "Authentication"
3. Configure a custom OAuth provider. You can use any provider. For ease of setup, the reproduction steps make use of maSSO[1], running on `localhost:8081`
4. Set the required fields to the following values:

   • Client ID: `web`
   • Client Secret: `secret`
   • Authorize URL: `http://localhost:8081/authorize`
   • Token URL: `http://localhost:8081/oauth/token`
   • Scope: `*`
   • Profile URL: `http://localhost:8025/api/v2/messages` (**MailHog URL**)

5. Save the configuration
6. From the "Site Settings" > "Authentication" perform a test of the new OAuth provider and log in with any user
7. Verify that after completing the login, the result page shows the messages from the MailHog server

Impact

High. An attacker can leverage this issue to access information hosted on internal hosts. The ultimate impact depends on which host are available on the network. For self-hosted instances, the issue will allow access to unauthenticated MailHog APIs, as demonstrated by the reproduction steps. In extreme cases, the issue can allow access to cloud metadata instances, which in turn will disclose valid cloud credentials.

Complexity

Medium to **Low**. Access to an administrative account is required to exploit this issue. However, the self-signup for the Fider SaaS will allow creation of such accounts, and also provide a path to potentially exploit any misconfigurations.

Remediation

Attempts to guard against Server Side Request Forgery are often implemented incorrectly, by either blocking all IP addresses, not handling IPv6, following HTTP redirects, not accounting for differences between various parsers or having *Time of Check to Time of Use (TOCTTOU)* issues, among other issues.

1. https://github.com/doyensec/maSSO

Because of the complicated nature of this issue, we recommend a defense-in-depth strategy, including as many of these steps as possible:

1. Wherever possible, implement network segmentation (e.g., firewalls, `iptables`) to prevent internal requests from services intended to make external outbound requests with user-controlled data,
2. If that is not possible everywhere and/or as an additional layer, use an outbound proxy for all requests to prevent internal connections
3. Implement authentication on **all** internal services/endpoints, so that services making connections based on user-controlled data must have a secret token for each request intended to connect internally (e.g., how AWS's Instance Metadata Service Version 2 (IMDSv2) works). Logically, all internal requests would be blocked and require a specific library be used to pass the authentication, so that the code is "safe by default"
4. Use a well-vetted library to process the requests and reject ones targeting internal resources.

Although use of a well vetted library in the application layer can help to reduce the risk of these vulnerabilities, they are often vulnerable to bypasses due to the complexities involved. As a result, many of the maintainers have ceased support for their open source projects as discussed here.

One example of such a library for Go is safeurl[2]. It protects against SSRF by resolving a hostname to an IP address and then checking whether the IP address belongs to a private network (RFC 1918).

Resources

- OWASP, “Server Side Request Forgery”
  https://www.owasp.org/index.php/Server_Side_Request_Forgery

Disclosure Timeline

06/11/2026	Issue responsibly disclosed to Fider maintainers
06/12/2026	A patch was introduced with PR#1567
07/01/2026	Advisory published

2. https://github.com/doyensec/safeurl