Abraxas Labs
ABRAXAS

DARK LIGHT

Research / forgejo-openid-discovery-ssrf

0-DAY NO CVE HIGH PUBLIC

Forgejo, unauthenticated OpenID discovery SSRF

Forgejo 16.0.5 OpenID sign-in still discovers through yohcop http.DefaultClient with no hostmatcher. Unauth POST /user/login/openid drives a Yadis GET at loopback. Gitea CVE-2026-58314 was not ported. No CVE yet.

NAME Forgejo, unauthenticated OpenID discovery SSRF	TYPE 0-day analysis
CVE n/a	CVE RISK high
DISCLOSURE STATUS public	VENDOR Forgejo
AFFECTED Forgejo through v16.0.5 (8d24b7d) with ENABLE_OPENID_SIGNIN. Web installer checkbox defaults on. Docker pre-locked template without [openid] is off unless the operator sets the key.	PUBLISHED 12 Oct 2026
UPDATED 12 Oct 2026	TAGS 0-day, forgejo, ssrf, openid, cwe-918, unauthenticated

TL;DR

Lab files and proof-of-concept **here**.

01 Webhooks filter. OpenID does not.

I am @abraxas_null. The proof of concept is on GitHub: **abraxas/forgejo-openid-discovery-ssrf** (loopback client). The lab stack is `lab/`: Dockerfile, docker-compose.yml, run.sh, poc.py. Authorized lab only. It talks to loopback.

This is **Forgejo v16.0.5** (`8d24b7d52df327118cd776cf461d34e52e0ec1ce`). No CVE yet. CWE-918. 8.6 High (`CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:N/A:N`) when OpenID sign-in is enabled. Remote SSRF. Unauthenticated.

When `[openid] ENABLE_OPENID_SIGNIN` is on, unauthenticated `POST /user/login/openid` hands the attacker URL to yohcop `openid-go` v1.0.1. That library's package-level helpers use `http.DefaultClient`. There is no hostmatcher on this path. Webhooks and migrations filter outbound HTTP. OpenID does not.

02 Why Forgejo

I had just finished Gitea's OpenID leftover class (CVE-2026-58314 / PR 38406). Forgejo closed the association CSRF on the same surface in v16.0.5 and stopped. Same surface, different bug. Worth sitting on.

I only wanted Critical or High. Local is in-scope.

03 Pinning v16.0.5

Latest GA: **v16.0.5**, commit `8d24b7d` , 2026-09-17 security patches (#14370). Image `codeberg.org/forgejo/forgejo:16.0.5` . Lab turns `FORGEJO__openid__ENABLE_OPENID_SIGNIN=true` because a pre-locked Docker template without `[openid]` is off.

04 The skip list is half the hunt

v16.0.5 / security-announcements#60 closed OpenID identity association CSRF on GET `/user/settings/security`. The callback is bound to the logged-in user's session nonce (`openIDState`). Do not re-file that.

Also closed on this pin:

- ApplyDiffPatch RCE leftover of PR 13705.
- CVE-2026-89094 template `.git`.
- Git clone/push/migrate `http.followRedirects=false`.
- CVE-2026-82556 migrate `LookupIP`.
- v16.0.2 CGNAT on the webhook matcher (the `0.0.0.1` hole is a different post).
- Org-mode `ReadFile` stub.
- OAuth `redirect_uri` skip, `client_secret` leak, SSO auto-admin. Those did not meet the bar.

OpenID login ATO via auto-link-by-email is not here. Existing `user_open_id` row signs in as that user under OP control. Register needs `ENABLE_OPENID_SIGNUP` and refuses a used email. Not High.

CVE-2021-45325 already stopped dumping fetch bodies into the login error. This leftover is the outbound request, not an HTML leak.

05 Twenty surfaces, one unfiltered Discover

Templates, git-apply, Actions, packages, migrate, hooks, tokens, federation, OpenID, attachments, restricted users, hostmatcher, LFS, keys, restore, OAuth, markup, internal API, secrets, visibility.

Migrations and webhooks import `hostmatcher`. `modules/auth/openid/` does not. That is the leftover.

06 The package-level client

GO Copy

```go
func RedirectURL(id, callbackURL, realm string) (string, error) {
    return openid.RedirectURL(id, callbackURL, realm)
}
```

yohcop's package-level helpers use `defaultInstance = NewOpenID(http.DefaultClient)`. Discover does Yadis GET then HTML discovery. `http.DefaultClient` follows redirects. No IP filter.

`SignInOpenIDPost` normalizes, checks the URI whitelist (empty = allow all), then `openid.RedirectURL`. Route group is `reqSignOut`, so CrossOriginProtection is skipped. The sign-in form has no CSRF field.

The web installer checkbox defaults on: `EnableOpenIDSignIn = MustBool(!InstallLock)` while unlocked, then written to `app.ini`. After lock, an omitted key is false. The Docker `app.ini` template has no `[openid]` section, so a pre-locked image without the wizard is off unless the operator sets the key. `custom/conf/app.example.ini` documents `;ENABLE_OPENID_SIGNIN = true`. Empty `WHITELISTED_URIS` allows every URI.

Logged-in `POST /user/settings/security/openid` is the same `RedirectURL` sink and does **not** call `allowedOpenIDURI`. Authenticated SSRF even if a login whitelist exists.

Gitea 1.27.0 (PR 38406 / CVE-2026-58314) put a hostmatcher-filtered `http.Client` into a dedicated `openid-go` instance. That patch is not here.

07 Wrong turns, in the order I made them

Re-filing the association CSRF. The nonce is in tree. Cross-user plant needs the victim's session cookie plus that nonce.

Treating a Docker pre-locked image as the default. The wizard writes the checkbox on. The omitted key after lock is false. Lab sets the env.

Expecting the old OpenID error-string leak. CVE-2021-45325 closed the HTML dump. The request still happens.

Completing an OpenID login. The leftover is Discover, not a session. The lab stops at the Yadis GET.

A reverse shell. Theatre. The witness is `Accept: application/xrds+xml` on loopback.

08 What an attacker can do

No account. POST an OpenID URL. The Forgejo host fetches it.

PRIVILEGE	METHOD / SINK	RECON / INTEGRITY / AVAILABILIT
Drive a Yadis GET at loopback	Unauth `POST /user/login/openid` with `openid=http://127.0.0.1:18080/`	Recon (outbound HTTP from the Forgejo host)

PRIVILEGE	METHOD / SINK	RECON / INTEGRITY / AVAILABILIT
Probe RFC1918, CGNAT, link-local, IMDS	Same `http.DefaultClient` Discover, no IP filter	Recon
Repeat from a signed-in add-OpenID	`POST /user/settings/security/openid` also calls `RedirectURL` and skips `allowedOpenIDURI`	Recon
Read response bodies in the login error	After CVE-2021-45325, errors no longer dump the fetch	Recon

Closed on this pin: OpenID association CSRF (`openIDState` session nonce), webhook/migrate hostmatcher (still used there), git `followRedirects=false`, migrate `LookupIP` CVE-2026-82556. The lab does not complete an OpenID login and does not return a shell.

09 The lab

Source of truth is `lab/`. Image `codeberg.org/forgejo/forgejo:16.0.5` with OpenID sign-in on. Host bind **127.0.0.1:19330**. Oracle shares the Forgejo netns and binds `127.0.0.1:18080`. Unauth POST `openid=http://127.0.0.1:18080/`. Oracle records the Yadis GET.

PLAIN TEXT Copy

```
git clone https://github.com/abraxas/forgejo-openid-discovery-ssrf
cd forgejo-openid-discovery-ssrf
python3 forgejo-openid-discovery-ssrf-Abraxas-Labs.py
```

Last lab run, trimmed:

```
PLAIN TEXT
Copy

1 IOC post-openid status=200
2 IOC xrds-hits count=1
3 SUCCESS forgejo-openid-discovery-ssrf how=live post=/user/login/openid openid=htt
```

I am not printing an OpenID URL you can paste at someone else's login form. The missing hostmatcher is the useful part.

10 The fix

Same as Gitea PR 38406: a hostmatcher-filtered `http.Client` into a dedicated `openid-go` instance. Keep `WHITELISTED_URIS` as a second gate. Block loopback, RFC1918, CGNAT, link-local, and metadata.

Same tag leftovers: keys IDOR, webhook `0.0.0.1` SSRF, restore-repo file open.

11 References

- Proof of concept: abraxas/forgejo-openid-discovery-ssrf · forgejo-openid-discovery-ssrf-Abraxas-Labs.py
- Lab: `lab/` · Dockerfile · docker-compose.yml · run.sh · poc.py
- @abraxas_null · github.com/abraxas · abraxaslabs.tech · abraxas.null@proton.me
- CWE-918

- Tree: forgejo/forgejo v16.0.5 (`8d24b7d52df327118cd776cf461d34e52e0ec1ce`) · `openid.go`
- Nearby patched (not this fetch): v16.0.5 OpenID association CSRF
- Gitea class: CVE-2026-58314 / PR 38406

Abraxas Labs

Independent security research by abraxas.

Analysis of 0-days, n-days, malware, and the

leftovers that still matter.

GitHub **@abraxas_null** **abraxas.null@proton.me** **abraxaslabs.tech**

Independent research. Private analysis by arrangement. No ads. abraxas.null@proton.me · For defensive awareness and authorized testing only.

© 2026 Abraxas Labs