A Capital Offence: Read-to-Write Privilege Escalation in Gogs over Git HTTP (CVE-2026-123012)

November 4, 2026

CVE-2026-123012 is a read-to-write privilege escalation in Gogs 0.14.3 caused by a case-sensitivity desync between the authorization classifier and the request dispatcher in `internal/route/repo/http.go`, where the classifier compares the Git action verb case-sensitively and defaults to pull while the dispatcher lowercases the same verb before routing, so capitalising a single letter in the URL, `info/Refs` instead of `info/refs`, lets an authenticated read-only collaborator push to a private repository. I found and reported this vulnerability, it was assigned CVE-2026-123012 (NVD, MITRE) at CVSS 4.0 base 7.1 HIGH, and I confirmed it end-to-end with a differential test against a patched build. The vendor introduced the bug in 0.14.3 while refactoring the same handler to fix a sibling issue, flipping the classifier's default from fail-closed to fail-open in the process.

This post walks through the patch-diff that surfaced it, the root cause across both functions, and the differential confirmation where a read-only collaborator writes a branch on the stock build and gets denied on the patched one.

Scope

Gogs is a lightweight, self-hosted Git forge written in Go that exposes the Git Smart HTTP protocol on the same port as its web UI, which means the authorization code path is directly reachable from the network and in practice from the public internet. All testing was performed locally against two operator-compiled builds (stock 0.14.3 and the same tree with a one-line fix) and no third-party instance was contacted.

> Advisory summary

Field	Value
CVE	CVE-2026-123012
CVSS v4.0	**7.1 HIGH** `AV:N/AC:L/AT:N/PR:L/UI:N/VC:N/VI:H/VA:L/SC:N/SI:L/SA:N`
Product	Gogs (self-hosted Git service)
Affected	**0.14.3** and `main` at time of writing
Introduced	0.14.3, June batch refactor of the Git HTTP handler; 0.14.2 failed closed on this path
Class	CWE-178 (improper handling of case sensitivity), producing CWE-863 (incorrect authorization)

Field	Value
Relationship	Distinct from CVE-2026-52810, a different defect in the same handler area
Root cause	`internal/route/repo/http.go` : pull-vs-push classifier is case-sensitive and defaults to pull, while the request dispatcher lowercases the path before routing
Impact	Authenticated read-only user can push to a private repository: create, update, or delete refs

The `VI:H` reflects total integrity loss on the vulnerable system because a reader can rewrite the repository, and `SI:L` captures the supply-chain consequence since injected commits are trusted by every downstream consumer that pulls from it.

Correction before publication: an early analysis claimed unauthenticated push against public repos, but that claim is wrong, and the real issue is the authenticated read-to-write escalation on private repos, encoded as `PR:L` in the vector.

> Mitigation guidance

Any Gogs 0.14.3 instance, or any newer build that has not normalised the Git action verb, should treat every read-only grant on a private repo as equivalent to write until patched, because the read/write privilege boundary collapses for any

principal that can authenticate with read access, including direct collaborators and org team members.

Detection: behavioural test

Issue two requests as a read-only collaborator against a private repo you control:

```
A (canonical): GET /{owner}/{repo}/info/refs?service=git-receive-pack
B (variant):   GET /{owner}/{repo}/info/Refs?service=git-receive-pack
```

The only difference is the capital `R` in request B.

A	B	Verdict
`403`	`200` (serves `git-receive-pack` advertisement)	**Vulnerable**
`403`	`403`	Patched or not affected

Request A is the control that confirms the account genuinely has no write access, and a `200` on request B means the server misclassified a push request as a read and served the push advertisement to a user who should never have seen it.

Remediation

Primary: upgrade to a build that normalises the action verb before the pull/push decision.

Interim (reverse-proxy): reject any Git Smart HTTP request whose path segment after `{owner}/{repo}` is not fully lowercase, since the canonical Git client never varies case on these verbs, which means this rule blocks the exploit with zero impact on legitimate traffic.

> Target selection

Gogs met the criteria because it is a network-reachable service where the authorization handler faces the internet directly rather than sitting behind a CLI or library boundary, the Go source and both the vulnerable and fixed trees are public with no licence or account required, and the defect sits in the authorization path of a service whose entire purpose is enforcing read vs. write on source code, so a read-to-write escalation in a Git forge is effectively supply-chain write access to everything downstream.

Prior art check

All channels checked 2026-11-05, the CVE publication date; treat the date as part of the claim since this table decays fast.

Channel	Result
GitHub (code + repo search for CVE ID, `gitHTTPIsPull`)	No public exploit or writeup

Channel	Result
nuclei-templates http/cves/2026/	Absent
Metasploit framework master	Nothing
Exploit-DB / Sploitus	No hits
CISA KEV	Not listed
General web search (excluding aggregator noise)	NVD, MITRE, vendor advisory only — no technical analysis

No public exploit was found, which is weak evidence of novelty rather than strong evidence; we found the primary records and nothing explaining the mechanism, and that gap is what this post fills.

> Diff analysis

The bug was introduced by a security fix, which is why the right comparison is 0.14.2 against 0.14.3.

In 0.14.2 the pull-vs-push classifier used a positive test that identified known read operations and required write access for everything else, a fail-closed design where unrecognised or oddly-shaped requests default to requiring write permission. The 0.14.3 refactor rewrote the same decision as a negative test against the push verb, so anything the classifier does not

recognise as the exact canonical push string is treated as a pull that only needs read access — a fail-open default. The refactor did fix a real pre-existing issue on the canonical path, but it opened a case-sensitivity side channel in the process.

Both trees are public on the Gogs GitHub repository, and the CVE record itself points at `internal/route/repo/http.go` around lines 64-68 for the classifier and 415-418 for the dispatcher, but the advisory never explains the interaction between the two functions, which is where the actual vulnerability lives.

> Root cause

The classifier: case-sensitive, defaults to pull

The `gitHTTPIsPull` function in `internal/route/repo/http.go` receives the raw URL suffix after `/{owner}/{repo}/` with its original capitalisation preserved, and both of its return paths use case-sensitive string comparison with a default of `true` (pull), which means any action string that is not byte-for-byte equal to the canonical push verb gets classified as a read:

```
func gitHTTPIsPull(c *macaron.Context, action string) bool {
    if action == "info/refs" {
        return c.Query("service") != "git-receive-pack"
    }
    return action != "git-receive-pack"
}
```

The authorization handler `HTTPContexter` consumes that boolean directly, and when `gitHTTPIsPull` returns `true` it enforces `AccessModeRead`, which a read-only collaborator passes cleanly:

```
mode := database.AccessModeWrite
if isPull {
    mode = database.AccessModeRead
}
if !database.Handle.Permissions().Authorize(
    c.Req.Context(), authUser.ID, repo.ID, mode, ...) {
    askCredentials(c, http.StatusForbidden, "User permission denied")
    return
}
```

The dispatcher: case-insensitive

The router matches on `strings.ToLower(c.Req.URL.Path)`, so a POST to `git-receive-Pack` with a capital `P` lowercases to `git-receive-pack` and routes to the receive-pack handler that accepts a packfile and updates refs. The authorization layer, which looked at the same request a moment earlier, saw a string that was not equal to the canonical verb and authorized it as a read, which is the entire desync.

The third guard: why the exploit keeps most of the verb lowercase

A pre-authentication guard rejects actions that do not look like Git operations, and it requires the lowercase substring `git-` or `info/` to be present via case-sensitive `strings.Contains`:

```
if !strings.Contains(action, "git-") &&
    !strings.Contains(action, "info/") &&
    !strings.Contains(action, "HEAD") &&
    !strings.Contains(action, "objects/") {
    c.Error(http.StatusBadRequest,
        fmt.Sprintf("Unrecognized action %q", action))
    return
}
```

A fully uppercased verb like `GIT-RECEIVE-PACK` fails this guard with a 400 before authorization even runs, so the naive approach of capitalising the whole verb is a dead end. The working request has to thread three constraints simultaneously: it must contain the lowercase substring `git-` or `info/` to pass the guard, it must differ from the canonical verb by at least one character so the classifier treats it as a pull, and it must lowercase to the real route so the dispatcher sends it to receive-pack. The way to satisfy all three is to keep the prefix lowercase and capitalise only a later character:

```
GET  /{owner}/{repo}/info/Refs?service=git-receive-pack
POST /{owner}/{repo}/git-receive-Pack
```

The canonical lowercase forms are correctly denied to a read-only user, while the variants above pass the guard, get misclassified as reads, and dispatch as writes.

> Confirmation: differential test

The confirmation was a differential against two local builds from the same source tree, because the claim is not “this code looks wrong” but “this exact request succeeds on the stock build and fails on the patched one”:

```
gogs          stock 0.14.3                              vulnerable   -
gogs-patched  0.14.3 + one-line case normalisation      patched      -
```

The one-line fix adds `strings.ToLower(action)` at the top of `gitHTTPIsPull`:

```
func gitHTTPIsPull(c *macaron.Context, action string) bool {
    action = strings.ToLower(action)
    if action == "info/refs" {
        return c.Query("service") != "git-receive-pack"
    }
    return action != "git-receive-pack"
}
```

The scenario uses a private repo `alice/proj` with a second account `mallory` added as a read-only collaborator.

Results

Request (as `mallory`)	Stock :3000	Patched :3001
`GET info/refs?service=git-receive-pack` (canonical)	`403`	`403`
`GET info/Refs?service=git-receive-pack` (variant)	**`200`**, push advertisement served	`403`
`POST git-receive-Pack` → create `refs/heads/x`	**`200`**, `unpack ok`, `ok refs/heads/x`, ref written to disk	Denied (no advertisement)

The canonical `403` on both builds proves `mallory` has no write access, and the `200` on the variant against stock 0.14.3 is the vulnerability: the server handed a read-only user the push advertisement, accepted the push, and the new ref was present on disk afterwards. The patched build refuses the same request with `403` because after normalisation the classifier and dispatcher agree it is a write. Same request, two builds, opposite outcome, one line of difference.

> Proof of concept

The verification tooling is a single Python script that drives the Smart HTTP protocol directly rather than shelling out to `git`, giving byte-level control over the request so the case variation is applied exactly where it matters.

Check mode (safe, read-only)

Issues both GET advertisements and compares status codes, writes nothing:

```
RECV = "git-receive-pack"
VARIANT_INFO = "info/Refs"  # canonical: "info/refs"

def advert_url(base, repo, segment):
    return f"{base}/{repo}/{segment}?service={RECV}"

def do_check(base, repo):
    canon_code, _      = http(advert_url(base, repo, "info/refs"))
    var_code, var_body  = http(advert_url(base, repo, VARIANT_INFO))
    served = var_code == 200 and b"service=git-receive-pack" in var_bo
    if canon_code in (401, 403) and served:
        print("VULNERABLE: read-only user reached the push advertiseme
```

Push mode

The push constructs a POST to the case-variant endpoint with a well-formed receive-pack command; the ref-update framing and packfile construction are withheld since publishing them saves an attacker the only non-trivial step and gives a defender nothing the check mode does not already provide.

> References

- NVD: CVE-2026-123012

- MITRE: CVE-2026-123012
- Rapid7: CVE-2026-123012
- Affected source: internal/route/repo/http.go

Author: Khalid Halloumi

< Back to Knowledge Base