

Vulnerability Analysis: CVE-2026-60004 Pre-Auth Remote Code Execution in Gitea

Author: Denizhalil

Site: denizhalil.com

CVSS 9.8 CRITICAL

Date: August 2026

INTRODUCTION

Securing self-hosted version control platforms is critical for maintaining the integrity of modern software development pipelines, source code repositories, and automated CI/CD workflows. Gitea, a lightweight and widely adopted self-hosted Git service, recently addressed a critical security vulnerability identified as **CVE-2026-60004**. Carrying a maximum CVSS v3 score of 9.8 (Critical), this flaw enables unauthenticated remote attackers under default system configurations to achieve arbitrary code execution on the underlying server. Because self-hosted instances often hold high-privilege credentials and sensitive internal intellectual property, compromising the Gitea daemon places entire enterprise environments at severe risk of data exfiltration and lateral movement.

LEARNING OBJECTIVES

By reading this article, you will be able to:

- Understand the scope and impact of CVE-2026-60004 in Gitea instances.
- Identify the technical flaw in patch handling and bare Git repository management that enables arbitrary file writes.
- Recognize how default configuration choices (such as open user registration) escalate user-level privileges to remote code execution.
- Apply effective remediation and temporary mitigation techniques to protect your infrastructure.

WHAT IS GITEA <= 1.27.0 - PRE-AUTH REMOTE CODE EXECUTION CVE-2026-60004

CVE-2026-60004 is a critical security vulnerability affecting Gitea versions 1.17.0 through 1.27.0. The security flaw lies within the repository patch parsing and application mechanism, specifically exposed via the HTTP REST API endpoint responsible for applying diff patches (`/api/v1/repos/{owner}/{repo}/diffpatch`). This vulnerability carries a CVSS v3 score of 9.8 (Critical), reflecting its potential to grant full control over the host operating system executing the Gitea daemon process.

The severe impact of CVE-2026-60004 stems from a combination of improper path resolution in temporary Git operations and standard deployment defaults. While triggering the vulnerability technically requires repository write access, out-of-the-box Gitea installations leave public account creation completely open. Without requiring administrator approval or email confirmation, an

anonymous remote attacker can seamlessly register an account, initialize a new public or private repository, gain immediate write privileges, and exploit the vulnerability. Consequently, the flaw acts as a practical pre-authentication remote code execution vector against default deployments.

Beyond initial system compromise, the vulnerability allows attackers to execute arbitrary shell commands within the security context of the user running Gitea. In enterprise environments, this can lead to total source code exfiltration, exposure of sensitive API keys and database credentials, compromise of automated CI/CD pipelines, and potential lateral movement across the internal corporate network.

- **CVSS Score & Impact:** Rated 9.8 (Critical), granting complete Remote Code Execution (RCE) on the host machine.
- **Vulnerable API Endpoint:** `POST /api/v1/repos/{owner}/{repo}/diffpatch` utilized for applying unified diff patches.
- **Pre-Auth Attack Vector:** Exploitability by unauthenticated external attackers due to default open self-registration settings (`DISABLE_REGISTRATION = false`).
- **Root Cause Mechanism:** Unchecked file writing inside temporary bare Git repositories (`$GIT_DIR`), leading to Git executable hook file overwrites (e.g., `hooks/post-index-change`).

```
// Example Vulnerable Code Structure (Gitea patch handling prior to v1.27.1)
// In services/repository/patch.go

func ApplyDiffPatch(ctx *context.APIContext, opts *api.ApplyDiffPatchOptions) {
    // 1. Temporary bare clone created directly in memory/tmp path
    tmpRepo, err := git.InitBareRepository(tmpPath)
    if err != nil {
        ctx.Error(http.StatusInternalServerError, "InitBareRepository", err)
        return
    }

    // 2. Vulnerable execution: Operating inside a bare repository where $GIT_DIR is the
    root.
    // When git apply utilizes 3-way merge fallback, crafting patch paths causes files
    // to write directly into tmpRepo/hooks/ instead of being isolated.
    cmd := git.NewCommand(ctx, "apply", "--index", "--cached", patchFilePath)
    if err := cmd.RunInDir(tmpRepo.Path); err != nil {
        ctx.Error(http.StatusBadRequest, "GitApplyFailed", err)
        return
    }
}
```

TECHNICAL DETAIL: HOW THE VULNERABILITY WORKS

The technical underlying vulnerability lies in how Gitea manages temporary storage spaces when processing incoming unified diff patches through its API. To validate and index diffs before committing them to the target branch, Gitea creates an isolated temporary Git repository. Prior to version 1.27.1, Gitea initialized these temporary instances as **bare repositories** (`git init --bare`). In a bare repository, there is no separate working tree directory; instead, the root level of the folder functions directly as `$GIT_DIR`, housing internal control directories like `objects/`, `refs/`, and `hooks/`.

When applying the patch, Gitea invokes standard Git CLI subcommands such as `git apply --index --cached`. In modern Git releases (version 2.32 and above), when `git apply` encounters ambiguous patches or merge collisions, it automatically attempts a three-way merge fallback strategy. Under specific conflict conditions, Git bypasses the in-memory index restriction and writes out intermediate file blobs directly to disk using paths resolved relative to the repository root. Because the temporary clone is bare, writing a file with a targeted relative path like `hooks/post-index-change` places a executable shell script directly into Git's active hook execution directory.

Once the malicious payload script is written to `$GIT_DIR/hooks/post-index-change`, the attacker only needs Git to execute a subsequent operation that modifies or reads the repository index. As soon as Gitea runs the next internal Git command on that temporary repository, Git automatically invokes the `post-index-change` hook script. The OS executes the script's instructions immediately under the user context of the running Gitea process (typically `git` or `gitea`), completing the remote code execution chain.

- **Bare Repository Misconfiguration:** Temporary repositories were created without working trees, causing `$GIT_DIR` to coincide directly with the temporary working root folder.
- **Git Subprocess Fallback Side-Effects:** The reliance on `git apply` three-way merge fallbacks allows file creation outside the intended staging index upon patch conflict.
- **Arbitrary Hook Injection:** Attackers exploit relative path resolution to plant malicious shell scripts inside `hooks/post-index-change`.
- **Automatic Script Triggering:** Subsequent index-updating Git subcommands executed by Gitea automatically launch the injected hook binary with host system privileges.

```
# Example Proof-of-Concept Exploit Execution (CVE-2026-60004)

$ python3 gitea_rce_poc.py --url http://git.internal.corp --cmd "id && uname -a"

[*] Target: http://git.internal.corp
[*] Authenticated as 'attacker'. Creating temporary repository 'stage-repo'...
[+] Repository 'attacker/stage-repo' created successfully.
[*] Crafting malicious patch payload target -> hooks/post-index-change ...
[*] Submitting raw patch to /api/v1/repos/attacker/stage-repo/diffpatch ...
[+] Patch accepted. Three-way merge fallback forced file write to $GIT_DIR/hooks/post-index-change.
[*] Triggering Git hook execution via index refresh...

[+] SUCCESS! Command output:
-----
uid=1000(gitea) gid=1000(gitea) groups=1000(gitea),27(sudo)
Linux gitea-prod-server 6.8.0-45-generic #45-Ubuntu SMP PREEMPT_DYNAMIC x86_64 GNU/Linux
-----
```

URGENT MEASURES YOU NEED TO TAKE

Securing your infrastructure against CVE-2026-60004 requires prompt intervention to eliminate both the underlying root cause and the zero-barrier attack vectors. Because default installations expose open account registration alongside vulnerable patch handling routines, unauthenticated actors can quickly achieve remote code execution on vulnerable instances. System administrators must prioritize applying official vendor updates to ensure that temporary Git repositories are initialized with proper working tree isolations, preventing arbitrary file writes into system hook directories.

If an immediate system update cannot be scheduled right away, administrators should implement temporary surface-reduction mitigations to block unauthorized entry points. Restricting self-registration prevents external attackers from establishing the repository write privileges required to trigger the exploit chain. However, because temporary mitigations do not protect against compromised or existing malicious user accounts, upgrading to the patched release remains the only permanent fix.

- **Immediate Upgrade to Fixed Releases:** Update all self-hosted Gitea instances to version **1.27.1** or higher immediately to enforce non-bare temporary clone creation.
- **Disable Self-Registration (`app.ini`):** Set `DISABLE_REGISTRATION = true` under the `[service]` section in your `custom/conf/app.ini` configuration file to cut off the unauthenticated attack vector.
- **Restart Gitea Service:** Reload or restart the Gitea daemon process immediately after editing configuration files to ensure the registration restriction takes effect.
- **Audit User Accounts & Active Hooks:** Review recently registered user accounts and inspect existing repository `$GIT_DIR/hooks` directories for unauthorized or suspicious executable scripts.

CONCLUSION

CVE-2026-60004 serves as a stark reminder of the architectural security risks inherent in handling complex binary operations and untrusted file parsing within version control systems. By combining standard Git subprocess fallbacks with bare repository structures, the vulnerability transforms a seemingly constrained patch application into an arbitrary file write attack. When deployed under default configurations, this flaw completely bridges the gap between an unauthenticated web user and host-level system administration.

The severity of this vulnerability underscores the crucial role that defense-in-depth principles play in securing self-hosted developer infrastructure. Relying solely on endpoint authentication is insufficient when default system configurations grant write access to anonymous actors. Organizations must maintain strict baseline configurations, restrict open registration on core internal tools, and continuously monitor exposed API endpoints against unexpected subprocess behaviors.

System administrators running self-hosted Gitea instances should treat remediation of CVE-2026-60004 as an urgent priority. Upgrading immediately to version 1.27.1 or higher permanently closes the file-traversal path within temporary repository handling. Taking immediate action to update software, restrict public account access, and audit existing Git hooks remains the only reliable path to ensuring software pipeline integrity.