Path Traversal via Malicious Package Metadata.Name — Arbitrary File Write

High **AustinAbro321** published **GHSA-pj97-4p9w-gx3q** 6 hours ago

Package

github.com/zarf-dev/zarf (Go)

Affected versions

>=v0.23.0, <v0.74.2

Patched versions

>=v0.74.2

Description

Impact

This vulnerability impacts users of `zarf package inspect sbom` or `zarf package inspect documentation` on untrusted packages.

Patches

#4793, now fixed in version v0.74.2

Workarounds

Avoid inspecting unsigned packages

Description

The `package inspect sbom` and `package inspect documentation` subcommands construct output file paths by joining a user-controlled output directory with the package's `Metadata.Name` field, which is attacker-controlled data read from the package archive. The `Metadata.Name` field is validated against a regex on create, `^[a-z0-9][a-z0-9\-]*$`, however a malicious user could unarchive a package to change the `.Metadata.Name` field and the files inside the SBOMS.tar. This would lead to arbitrary file write in a location of the attackers choosing.

Neither location sanitizes or validates the package name before using it in the file path.

SBOM inspection:

```
outputPath := filepath.Join(o.outputDir, pkgLayout.Pkg.Metadata.Name)
err = pkgLayout.GetSBOM(ctx, outputPath)
```

Documentation inspection (line 1219):

```
outputPath := filepath.Join(o.outputDir, fmt.Sprintf("%s-documentation", pkgLayout
return pkgLayout.GetDocumentation(ctx, outputPath, o.keys)
```

`pkgLayout.Pkg.Metadata.Name` is read directly from the untrusted package's `zarf.yaml` manifest. An attacker can craft a malicious Zarf package where `Metadata.Name` contains path traversal sequences or root paths such as `../../etc/cron.d/malicious` or `/home/user/.ssh/authorized_keys`.

CVSS Explainations

Attack Vector

Verdict: Network
A malicious package could be published to OCI and inspected directly with `zarf package inspect sbom oci://<bad-package>`

Attack Complexity

Verdict: Low
It is not complicated to make and publish a malicious package. The Attacker only needs to edit the zarf.yaml and sboms.tar then edit the checksums.

Privileges Required

Verdict: None
The attacker is relying on the runner of `zarf package inspect sbom|documentation` and needs no other privileges.

User Interaction

Verdict: Required
The user must run the inspect command

Scope

Verdict: Unchanged
The vulnerability operates entirely within the permissions of the user running zarf package inspect. The file write can't escape the privilege boundary of that user

Confidentiality

Verdict: None
This is an arbitrary file write vulnerability. The attacker can place or overwrite files on the filesystem but the vulnerability does not provide any mechanism to read or exfiltrate data from the target system.

Integrity

Verdict: High
The attacker controls both the file path (via Metadata.Name) and the file content (via the SBOM or documentation files inside the archive). This allows writing attacker-controlled content to arbitrary locations on the filesystem, limited only by the permissions of the user running the inspect command. Realistic exploitation includes writing SSH authorized_keys, cron jobs, or shell profiles.

Availability

Verdict: Low
The vulnerability does not directly target service availability. However, an attacker could overwrite files that cause system disruption.

Severity

High 7.1 / 10

CVSS v3 base metrics

Attack vector	Network
Attack complexity	Low
Privileges required	None
User interaction	Required
Scope	Unchanged
Confidentiality	None
Integrity	High
Availability	Low

Learn more about base metrics

CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:N/I:H/A:L

CVE ID

CVE-2026-40090

Weaknesses

No CWEs

Credits

joonas Finder