DOYENSEC

Security Advisory
Caddy v2

Created by Adrian Denkiewicz
2022-10-14

Overview

This document summarizes the results of a vulnerability research activity aimed at discovering vulnerabilities in Caddy web-server. While security testing was not meant to be comprehensive in term of attack and code coverage, we have identified access control vulnerability affecting Caddy for Windows. When the `file_server` module is used, the attacker can access hidden files. This could lead to information leakage.

About Us

Doyensec is an independent security research and development company focused on vulnerability discovery and remediation. We work at the intersection of software development and offensive engineering to help companies craft secure code.

Research is one of our founding principles and we invest heavily in it. By discovering new vulnerabilities and attack techniques, we constantly improve our capabilities and contribute to secure the applications we all use.

Broken Access Control In The file_server Module

Vendor	Caddy (https://github.com/caddyserver/caddy)
Severity	Medium
Vulnerability Class	Insecure Design
Component	The file_server Module
Status	Open
CVE	N/a
Credits	Adrian Denkiewicz

Summary

The Caddy web-server can be configured to serve local files. This requires the user to explicitly turn on the `file_server` module. Additionally, the user can configure a list of hidden files using the `hide` directive. Per the documentation[1]:

> *A list of files or folders to hide; the file server will pretend as if they don't exist. Accepts globular patterns like *.ext or /foo/*/bar as well as placeholders.*

By default, the hide list also contains a full path to a configuration file used by the web-server. This suggests that it is common to start a web-server hosting the files inside the current working directory and the config file is not exposed only due to this mechanism.

The implemented logic does not take into consideration alternate file streams or short file names, both available on Windows' file systems. By using the streams or short names, the attacker can use alternate names to access the hidden files, thus bypassing the builtin protection.

In particular, attacker could also use this technique to read the content of the `Caddyfile` if available in the exposed directory.

[1] https://caddyserver.com/docs/json/apps/http/servers/routes/handle/file_server/hide/

Technical Description

We reviewed the v2.6.1 release of Caddy. The `fileHidden` function is implemented in the `modules/caddyhttp/fileserver/staticfiles.go` file. The function is not aware of alternate data streams[2] (ADS) or short names[3], both available on Windows systems.

Consider a file: `secret_file.txt` or its absolute path: `C:/caddy/secret_file.txt` and the following configuration:

```
example.com {
        root * C:/caddy
        file_server {
                hide secret_file.txt
        }
}
```

The standard HTTP request to the `/secret_file.txt` endpoint would result in a HTTP 404 error.

Bypass using ADS

The alternate method to access the same file is to append the name of its default stream (an empty string) and `$DATA` as its stream type. The additional components are separated using colon characters. The HTTP request would look like this:

```
GET /secret_file.txt::$DATA HTTP/1.1
Host: example.com
```

By using this method, the attacker will bypass the check against the configured `hide` list and gain access to the file content.

To access a hidden directory, the attacker would append `::$INDEX_ALLOCATION` or `:$I30:$INDEX_ALLOCATION` instead.

Accessing other ADSs could be seen as unintended data exposure. For instance, the `Zone.Identifier` stream could leak the information about file's origin.

[2] https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-fscc/c54dec26-1551-4d3a-a0ea-4fa40f848eb3

[3] https://learn.microsoft.com/en-us/windows/win32/fileio/naming-a-file#short-vs-long-names

This would not work against glob matching. E.g.: `test.txt*` would still hide the file.

Bypass using short names

Additionally, the attacker could use a short name to access the same file. The following HTTP request demonstrates this attack:

```
GET /SECRET~1.txt HTTP/1.1
Host: example.com
```

Windows automatically creates short filenames only for the files whose name exceeds the MS-DOS style naming convention. Per Wikipedia[4], the convention used by modern Windows systems is the following:

> *8.3 filenames are limited to at most eight characters (after any directory specifier), followed optionally by a filename extension consisting of a period . and at most three further characters.*

Remediation

The default configuration should not expose alternate data streams or short names.

We recommend returning HTTP 404 error for the aforementioned modifications or rejecting any file request with a colon character in its file name. To block access to short names, reject files with a tilde character. Note however that it may break compatibility with existing software hence additional flag could be introduced to control this behavior on Windows.

Apache httpd rejects paths with colon or tilde characters if used in the context of short names.

Nginx performs more granular validation and rejects only particular sequences of characters[5] [6].

[4] https://en.wikipedia.org/wiki/8.3_filename

[5] https://github.com/nginx/nginx/commit/0d7720ddc052d5bf8aa09485a2a6bb9004bb943d

[6] https://github.com/nginx/nginx/commit/db80a7adfc1636ef146006b652c1892d42847ccd

Disclosure Timeline

10/14/2022	Issue identified and reported to the vendor
10/19/2022	Issue fixed on the master branch
02/08/2023	Caddy v2.6.3 released with the fix
02/20/2023	Official Security Advisory published