<> Code   ⊙ Issues 82   Pull requests 21   ▷ Actions   ⚠ Security   ⁓ Insights

New issue

[panic]: slice OOB caused by illegal uri #4775

⊘ Closed   **secsys-go** opened this issue on May 9 · 8 comments

Assignees	
Labels	bug 🐞
Milestone	⬗ v2.5.2

---

secsys-go commented on May 9

It occurs in modules/caddyhttp/rewrite/rewrite.go.
Specifically, this bug locates in `func (rewr Rewrite) rewrite`

```
func (rewr Rewrite) rewrite(r *http.Request, repl *caddy.Replacer, logger *zap.Logger) bool {
        oldMethod := r.Method
        oldURI := r.RequestURI

        // method
        if rewr.Method != "" {
                r.Method = strings.ToUpper(repl.ReplaceAll(rewr.Method, ""))
        }

        // uri (path, query string and... fragment, because why not)
        if uri := rewr.URI; uri != "" {
                // find the bounds of each part of the URI that exist
                pathStart, qsStart, fragStart := -1, -1, -1
                pathEnd, qsEnd := -1, -1
                for i, ch := range uri {
                        switch {
                        case ch == '?' && qsStart < 0:
                                pathEnd, qsStart = i, i+1
                        case ch == '#' && fragStart < 0:
                                qsEnd, fragStart = i, i+1
                        case pathStart < 0 && qsStart < 0 && fragStart < 0:
                                pathStart = i
                        }
                }
                if pathStart >= 0 && pathEnd < 0 {
                        pathEnd = len(uri)
```

```
                }
                if qsStart >= 0 && qsEnd < 0 {
                        qsEnd = len(uri)
                }

                // isolate the three main components of the URI
                var path, query, frag string
                if pathStart > -1 {
                        path = uri[pathStart:pathEnd]
                }
                if qsStart > -1 {
                        query = uri[qsStart:qsEnd]
                }
                if fragStart > -1 {
                        frag = uri[fragStart:]
                }
        ...
```

In this function, it parse the `rewr.URI` and attemps to find the bounds of each part of the URI that exist.
However, this implementation is too simple to handle unexpected scenarios.

If '#' appears in front of the '?' in `rewr.URI`, it makes that `qsStart` is larger than `qsEnd`,
which leads to a crash like *'panic: runtime error: slice bounds out of range'* in slice accessing at `query = uri[qsStart:qsEnd]`

🏷 **francislavoie** added the bug 🐞 label on May 9

---

mholt commented on May 9 • edited ▾   (Member)

Thanks for the report. In the future, please post the precise input and log output to expedite a fix.

In order to add a regression test, can you please share the exact input you had that caused the bug? I can find some too but I need to make sure your case is covered too. What was your config? i.e. what are you rewriting to?

Edit: I'm not able to reproduce the bug with an input of `/foo#a?b=c`.

👤 🖼 **mholt** self-assigned this on May 9

🏷 🖼 **mholt** added the needs info 🟥 label on May 9

---

secsys-go commented on May 9   (Author)
```

```

In my case, `rewr.URI="/#?"`

As for the precise log output, I made some modification on `TestRewrite` and get that:

```
25  | func TestRewrite1(t *testing.T) {
26  |     repl := caddy.NewReplacer()
27  |
28  |     tc := struct {
29  |         input, expect *http.Request
30  |         rule          Rewrite
31  |     }{
32  |         rule:   Rewrite{URI: "/#?"},
33  |         input: newRequest(t, "GET", "/foo/bar?a=b"),
34  |         expect: newRequest(t, "GET", "/foo?a=b#frag"),
35  |     }
36  |
37  |     // copy the original input just enough so that we can
38  |     // compare it after the rewrite to see if it changed
39  |     originalInput := &http.Request{
40  |         Method:     tc.input.Method,
41  |         RequestURI: tc.input.RequestURI,
42  |         URL:        &*tc.input.URL,
43  |     }
59  |     changed := tc.rule.rewrite(tc.input, repl, nil)
60  |
61  |     if expected, actual := !reqEqual(originalInput, tc.input), changed; expected !=
actual {
62  |         t.Errorf("Expected changed=%t but was %t", expected, actual)
63  |     }
64  |     if expected, actual := tc.expect.Method, tc.input.Method; expected != actual {
65  |         t.Errorf("Expected Method='%s' but got '%s'", expected, actual)
66  |     }
67  |     if expected, actual := tc.expect.RequestURI, tc.input.RequestURI; expected != actual
{
68  |         t.Errorf("Expected RequestURI='%s' but got '%s'", expected, actual)
69  |     }
70  |     if expected, actual := tc.expect.URL.String(), tc.input.URL.String(); expected !=
actual {
71  |         t.Errorf("Expected URL='%s' but got '%s'", expected, actual)
72  |     }
73  |     if expected, actual := tc.expect.URL.RequestURI(), tc.input.URL.RequestURI();
expected != actual {
74  |         t.Errorf("Expected URL.RequestURI()='%s' but got '%s'", expected, actual)
75  |     }
76  |     if expected, actual := tc.expect.URL.Fragment, tc.input.URL.Fragment; expected !=
actual {
77  |         t.Errorf("Expected URL.Fragment='%s' but got '%s'", expected, actual)
78  |     }
79  |
80  |     return
81  | }
```

And it crashed like?:

```
--- FAIL: TestRewrite1 (0.00s)
panic: runtime error: slice bounds out of range [3:1] [recovered]
        panic: runtime error: slice bounds out of range [3:1]

goroutine 25 [running]:
testing.tRunner.func1.2({0x12f70c0, 0xc0001e0a98})
        /home/zjx/.local/go/src/testing/testing.go:1211 +0x24e
testing.tRunner.func1()
        /home/zjx/.local/go/src/testing/testing.go:1214 +0x218
panic({0x12f70c0, 0xc0001e0a98})
        /home/zjx/.local/go/src/runtime/panic.go:1038 +0x215
github.com/caddyserver/caddy/v2/modules/caddyhttp/rewrite.Rewrite.rewrite({{0x0, 0x0}, {0x139e2a5,
0x3}, {0x0, 0x0}, {0x0, 0x0}, {0x0, 0x0, ...}, ...}, ...)
        /home/zjx/workspace/gowork/src/go-fdg-
exmaples/caddy/modules/caddyhttp/rewrite/rewrite.go:161 +0xc65
github.com/caddyserver/caddy/v2/modules/caddyhttp/rewrite.TestRewrite1(0xc000471520)
        /home/zjx/workspace/gowork/src/go-fdg-
exmaples/caddy/modules/caddyhttp/rewrite/tmp_test.go:59 +0x525
testing.tRunner(0xc000471520, 0x1410440)
        /home/zjx/.local/go/src/testing/testing.go:1261 +0x102
created by testing.(*T).Run
        /home/zjx/.local/go/src/testing/testing.go:1308 +0x35a
exit status 2
```

mholt commented on May 9   Member

Thanks! I see it now.

🏷️ **mholt** removed the   **needs info** 📬   label on May 9

mholt closed this as completed in `693e9b5` on May 9

---

mholt commented on May 9   Member

Should be fixed in `693e9b5` . Feel free to double-check my work!

👍 2

🔀 **mholt** added this to the **v2.5.2** milestone on May 9

secsys-go commented on Jul 24   Author

This issue has been assigned as [CVE-2022-34037](#).

😮 2

---

mholt commented on Jul 26 • edited ▾

This bug only affected the client of the request, and to my knowledge cannot DoS anyone other than the attacker; i.e. no attack surface is possible here in the server. (Would be a client bug if the client allows user to make invalid/malformed request when expecting something to work.) The CVE is unwarranted IMO.

👍 3

---

francislavoie commented on Aug 2 • edited ▾

IMO, this CVE is absurd. It was given a score of 7.5. But it requires a bad config, AND it has no real attack surface. The panic does *not* take down the server, it just stops the current request which contained bad input, and emits a log. There's no exploit here.

Example Caddyfile config of the reported issue:

```
{
        debug
}

:8888 {
        rewrite * /#?
        respond "Hello"
}
```

Run Caddy v2.5.1, making a request with `curl -v http://localhost:8888` twice, notice the server never shuts down, it just prints two `DEBUG` level logs (newlines for emphasis), because the `debug` global option is enabled, which shows the panic message.

```
$ ./caddy run
2022/08/03 01:53:29.381 INFO    using adjacent Caddyfile
2022/08/03 01:53:29.381 WARN    admin    admin endpoint disabled
2022/08/03 01:53:29.381 INFO    tls.cache.maintenance    started background certificate maintenance
{"cache": "0xc00034fea0"}
2022/08/03 01:53:29.382 INFO    tls    cleaning storage unit    {"description":
"FileStorage:/var/lib/caddy/.local/share/caddy"}
2022/08/03 01:53:29.382 DEBUG    http    starting server loop    {"address": "[::]:8888", "http3":
false, "tls": false}
2022/08/03 01:53:29.385 INFO    autosaved config (load with --resume flag)    {"file":
"/var/lib/caddy/.config/caddy/autosave.json"}
2022/08/03 01:53:29.385 INFO    serving initial configuration
```

```
2022/08/03 01:53:29.385 INFO    tls     finished cleaning storage units

2022/08/03 01:53:40.694 DEBUG   http.stdlib     http: panic serving 127.0.0.1:43806: runtime
error: slice bounds out of range [3:1]
goroutine 36 [running]:
net/http.(*conn).serve.func1()
        net/http/server.go:1825 +0xbf
panic({0x186a1a0, 0xc0000422d0})
        runtime/panic.go:844 +0x258
github.com/caddyserver/caddy/v2/modules/caddyhttp/rewrite.Rewrite.Rewrite({{0x0, 0x0},
{0xc00009c008, 0x3}, {0x0, 0x0}, {0x0, 0x0}, {0x0, 0x0, ...}, ...}, ...)
        github.com/caddyserver/caddy/v2@v2.5.1/modules/caddyhttp/rewrite/rewrite.go:161 +0xc65
github.com/caddyserver/caddy/v2/modules/caddyhttp/rewrite.Rewrite.ServeHTTP({{0x0, 0x0},
{0xc00009c008, 0x3}, {0x0, 0x0}, {0x0, 0x0}, {0x0, 0x0, ...}, ...}, ...)
        github.com/caddyserver/caddy/v2@v2.5.1/modules/caddyhttp/rewrite/rewrite.go:109 +0x20e
github.com/caddyserver/caddy/v2/modules/caddyhttp.
(*metricsInstrumentedHandler).ServeHTTP(0xc00053afe0, {0x1ca3ea0?, 0xc000158000}, 0xc0002fc200,
{0x1c9a3c0, 0xc00053b080})
        github.com/caddyserver/caddy/v2@v2.5.1/modules/caddyhttp/metrics.go:132 +0x53b
github.com/caddyserver/caddy/v2/modules/caddyhttp.wrapMiddleware.func1.1({0x1ca3ea0?,
0xc000158000?}, 0x1c9a3c0?)
        github.com/caddyserver/caddy/v2@v2.5.1/modules/caddyhttp/routes.go:272 +0x3b
github.com/caddyserver/caddy/v2@v2.5.1/modules/caddyhttp.HandlerFunc.ServeHTTP(0x1c9a3c0?, {0x1ca3ea0?,
0xc000158000?}, 0x6?)
        github.com/caddyserver/caddy/v2@v2.5.1/modules/caddyhttp/caddyhttp.go:57 +0x2f
github.com/caddyserver/caddy/v2/modules/caddyhttp.wrapRoute.func1.1({0x1ca3ea0, 0xc000158000},
0xc0002fc200)
        github.com/caddyserver/caddy/v2@v2.5.1/modules/caddyhttp/routes.go:244 +0x322
github.com/caddyserver/caddy/v2/modules/caddyhttp.HandlerFunc.ServeHTTP(0xc0002c2c00?,
{0x1ca3ea0?, 0xc000158000?}, 0x18a8200?)
        github.com/caddyserver/caddy/v2@v2.5.1/modules/caddyhttp/caddyhttp.go:57 +0x2f
github.com/caddyserver/caddy/v2/modules/caddyhttp.(*Server).enforcementHandler(0x0?, {0x1ca3ea0?,
0xc000158000?}, 0xc00069c6e0?, {0x1c9a3c0?, 0xc00053b0a0?})
        github.com/caddyserver/caddy/v2@v2.5.1/modules/caddyhttp/server.go:318 +0x252
github.com/caddyserver/caddy/v2/modules/caddyhttp.(*Server).wrapPrimaryRoute.func1({0x1ca3ea0?,
0xc000158000?}, 0x49b737?)
        github.com/caddyserver/caddy/v2@v2.5.1/modules/caddyhttp/server.go:294 +0x3b
github.com/caddyserver/caddy/v2/modules/caddyhttp.HandlerFunc.ServeHTTP(0xc0002b6bb0?,
{0x1ca3ea0?, 0xc000158000?}, 0xc0002fc200?)
        github.com/caddyserver/caddy/v2@v2.5.1/modules/caddyhttp/caddyhttp.go:57 +0x2f
github.com/caddyserver/caddy/v2/modules/caddyhttp.(*Server).ServeHTTP(0xc000024360, {0x1ca3ea0,
0xc000158000}, 0xc0002fc200)
        github.com/caddyserver/caddy/v2@v2.5.1/modules/caddyhttp/server.go:230 +0xac6
net/http.serverHandler.ServeHTTP({0xc000314120?}, {0x1ca3ea0, 0xc000158000}, 0xc0002fc000)
        net/http/server.go:2916 +0x43b
net/http.(*conn).serve(0xc0000baaa0, {0x1ca5790, 0xc0001aa630})
        net/http/server.go:1966 +0x5d7
created by net/http.(*Server).Serve
        net/http/server.go:3071 +0x4db

2022/08/03 01:53:46.882 DEBUG   http.stdlib     http: panic serving 127.0.0.1:43808: runtime
error: slice bounds out of range [3:1]
goroutine 38 [running]:
net/http.(*conn).serve.func1()
        net/http/server.go:1825 +0xbf
panic({0x186a1a0, 0xc0000423c0})
```

```
            runtime/panic.go:844 +0x258
github.com/caddyserver/caddy/v2/modules/caddyhttp/rewrite.Rewrite.Rewrite({{0x0, 0x0},
{0xc00009c008, 0x3}, {0x0, 0x0}, {0x0, 0x0}, {0x0, 0x0, ...}, ...}, ...)
            github.com/caddyserver/caddy/v2@v2.5.1/modules/caddyhttp/rewrite/rewrite.go:161 +0xc65
github.com/caddyserver/caddy/v2/modules/caddyhttp/rewrite.Rewrite.ServeHTTP({{0x0, 0x0},
{0xc00009c008, 0x3}, {0x0, 0x0}, {0x0, 0x0}, {0x0, 0x0, ...}, ...}, ...)
            github.com/caddyserver/caddy/v2@v2.5.1/modules/caddyhttp/rewrite/rewrite.go:109 +0x20e
github.com/caddyserver/caddy/v2/modules/caddyhttp.
(*metricsInstrumentedHandler).ServeHTTP(0xc00053afe0, {0x1ca3ea0?, 0xc0001580e0}, 0xc0002fc500,
{0x1c9a3c0, 0xc00053b080})
            github.com/caddyserver/caddy/v2@v2.5.1/modules/caddyhttp/metrics.go:132 +0x53b
github.com/caddyserver/caddy/v2/modules/caddyhttp.wrapMiddleware.func1.1({0x1ca3ea0?,
0xc0001580e0?}, 0x1c9a3c0?)
            github.com/caddyserver/caddy/v2@v2.5.1/modules/caddyhttp/routes.go:272 +0x3b
github.com/caddyserver/caddy/v2/modules/caddyhttp.HandlerFunc.ServeHTTP(0x1c9a3c0?, {0x1ca3ea0?,
0xc0001580e0?}, 0x6?)
            github.com/caddyserver/caddy/v2@v2.5.1/modules/caddyhttp/caddyhttp.go:57 +0x2f
github.com/caddyserver/caddy/v2/modules/caddyhttp.wrapRoute.func1.1({0x1ca3ea0, 0xc0001580e0},
0xc0002fc500)
            github.com/caddyserver/caddy/v2@v2.5.1/modules/caddyhttp/routes.go:244 +0x322
github.com/caddyserver/caddy/v2/modules/caddyhttp.HandlerFunc.ServeHTTP(0xc0000dd400?,
{0x1ca3ea0?, 0xc0001580e0?}, 0x18a8200?)
            github.com/caddyserver/caddy/v2@v2.5.1/modules/caddyhttp/caddyhttp.go:57 +0x2f
github.com/caddyserver/caddy/v2/modules/caddyhttp.(*Server).enforcementHandler(0x0?, {0x1ca3ea0?,
0xc0001580e0?}, 0xc00069c850?, {0x1c9a3c0?, 0xc00053b0a0?})
            github.com/caddyserver/caddy/v2@v2.5.1/modules/caddyhttp/server.go:318 +0x252
github.com/caddyserver/caddy/v2/modules/caddyhttp.(*Server).wrapPrimaryRoute.func1({0x1ca3ea0?,
0xc0001580e0?}, 0x49b737?)
            github.com/caddyserver/caddy/v2@v2.5.1/modules/caddyhttp/server.go:294 +0x3b
github.com/caddyserver/caddy/v2/modules/caddyhttp.HandlerFunc.ServeHTTP(0xc0002b6bb0?,
{0x1ca3ea0?, 0xc0001580e0?}, 0xc0002fc500?)
            github.com/caddyserver/caddy/v2@v2.5.1/modules/caddyhttp/caddyhttp.go:57 +0x2f
github.com/caddyserver/caddy/v2/modules/caddyhttp.(*Server).ServeHTTP(0xc000024360, {0x1ca3ea0,
0xc0001580e0}, 0xc0002fc500)
            github.com/caddyserver/caddy/v2@v2.5.1/modules/caddyhttp/server.go:230 +0xac6
net/http.serverHandler.ServeHTTP({0xc0003145d0?}, {0x1ca3ea0, 0xc0001580e0}, 0xc0002fc300)
            net/http/server.go:2916 +0x43b
net/http.(*conn).serve(0xc0000babe0, {0x1ca5790, 0xc0001aa630})
            net/http/server.go:1966 +0x5d7
created by net/http.(*Server).Serve
            net/http/server.go:3071 +0x4db
```

If `debug` was not enabled, there wouldn't even be a log (the failing requests should be an indication to the user that they should turn on `debug` mode to see what's going on).

Curl output:

```
$ curl -v http://localhost:8888/
*   Trying 127.0.0.1:8888...
* Connected to localhost (127.0.0.1) port 8888 (#0)
> GET / HTTP/1.1
> Host: localhost:8888
> User-Agent: curl/7.74.0
> Accept: */*
```

```
>
* Empty reply from server
* Connection #0 to host localhost left intact
curl: (52) Empty reply from server
```

This is *very clearly* just a regular bug, there is *no* security implications here at all. The CVE should be revised or dismissed.

👍 3

---

mholt commented on Aug 3   Member

I've officially disputed the CVE with this letter:

> CVE-2022-34037 was assigned to an ordinary bug, not a security vulnerability, in the Caddy web server that emerged when an administrator's bad configuration containing a malformed request URI caused the server to return an empty reply instead of a valid HTTP response to the client.
>
> This CVE entry claims a Denial-of-Service (DoS) attack on the server, but this claim is rejected on the grounds that:
>
> 1. Only a faulty/nonsensical configuration causes the bug. It attempts to transform the request URI into an invalid syntax that would result in HTTP errors anyway. There is no practical use for such a configuration.
>
> 2. Only a trusted, authorized user is allowed to set the server configuration. This CVE entry does not demonstrate any unauthorized tampering or disabling of the server.
>
> 3. Only matched requests are affected (again, requires a trusted administrator's certain configuration). The CVE claims that the attacker can cause DoS "via a crafted URI," however that is not the case.
>
> 4. The server's availability is not impacted by this bug. The "panic" (which is just an error message with a stack trace) is recovered successfully, and it continues to serve requests as configured. It does not even yield a log entry unless verbose/debug logging is enabled.
>
> 5. This bug does not impact service availability. Denial-of-Service is defined by OWASP as to "make a service unavailable for legitimate users", further explaining that these attacks "significantly degrade the service quality experienced by legitimate users ... large response delays, excessive losses, and service interruptions, resulting in direct impact on availability." This bug does not fit that definition.
>
> 6. It is actually compelling to return an HTTP error response, since the faulty configuration attempts to rewrite the request URI to an invalid syntax. While empty HTTP responses are less than ideal, they are no more a security exploit than returning an HTTP 400 response. The fix we chose involves silently ignoring the invalid syntax, but returning an error is also a valid (and less confusing) way to resolve this.

7. A Caddy instance being given a bad configuration is not a security exploit in Caddy. It would be, however, if an untrusted user was able to cause Caddy to run an unauthorized configuration. But this CVE entry does not demonstrate any such vulnerability.

8. The severity rating is based on metrics that simply do not apply to this bug.

Our case can be objectively verified with this configuration:

```
{
        debug
}

:8888 {
        rewrite /bad /#?
        respond "Hello"
}
```

With Caddy running that config, first execute `curl -v http://localhost:8888/bad`, and then `curl -v http://localhost:8888`.

Server log:

```
$ ./caddy run
INFO    using adjacent Caddyfile
WARN    admin   admin endpoint disabled
INFO    tls.cache.maintenance   started background certificate maintenance      {"cache":
"0xc00034fea0"}
INFO    tls     cleaning storage unit   {"description":
"FileStorage:/var/lib/caddy/.local/share/caddy"}
DEBUG   http    starting server loop    {"address": "[::]:8888", "http3": false, "tls":
false}
INFO    autosaved config (load with --resume flag)      {"file":
"/var/lib/caddy/.config/caddy/autosave.json"}
INFO    serving initial configuration
INFO    tls     finished cleaning storage units

DEBUG   http.stdlib     http: panic serving 127.0.0.1:43806: runtime error: slice bounds out
of range [3:1]
goroutine 36 [running]:
net/http.(*conn).serve.func1()
        net/http/server.go:1825 +0xbf
panic({0x186a1a0, 0xc0000422d0})
        runtime/panic.go:844 +0x258
github.com/caddyserver/caddy/v2/modules/caddyhttp/rewrite.Rewrite.Rewrite({{0x0, 0x0},
{0xc00009c008, 0x3}, {0x0, 0x0}, {0x0, 0x0}, {0x0, 0x0, ...}, ...}, ...)
        github.com/caddyserver/caddy/v2@v2.5.1/modules/caddyhttp/rewrite/rewrite.go:161
+0xc65
github.com/caddyserver/caddy/v2/modules/caddyhttp/rewrite.Rewrite.ServeHTTP({{0x0, 0x0},
{0xc00009c008, 0x3}, {0x0, 0x0}, {0x0, 0x0}, {0x0, 0x0, ...}, ...}, ...)
        github.com/caddyserver/caddy/v2@v2.5.1/modules/caddyhttp/rewrite/rewrite.go:109
+0x20e
github.com/caddyserver/caddy/v2/modules/caddyhttp.
```

```
    (*metricsInstrumentedHandler).ServeHTTP(0xc00053afe0, {0x1ca3ea0?, 0xc000158000},
0xc0002fc200, {0x1c9a3c0, 0xc00053b080})
        github.com/caddyserver/caddy/v2@v2.5.1/modules/caddyhttp/metrics.go:132 +0x53b
github.com/caddyserver/caddy/v2/modules/caddyhttp.wrapMiddleware.func1.1({0x1ca3ea0?,
0xc000158000?}, 0x1c9a3c0?)
        github.com/caddyserver/caddy/v2@v2.5.1/modules/caddyhttp/routes.go:272 +0x3b
github.com/caddyserver/caddy/v2/modules/caddyhttp.HandlerFunc.ServeHTTP(0x1c9a3c0?,
{0x1ca3ea0?, 0xc000158000?}, 0x6?)
        github.com/caddyserver/caddy/v2@v2.5.1/modules/caddyhttp/caddyhttp.go:57 +0x2f
github.com/caddyserver/caddy/v2/modules/caddyhttp.wrapRoute.func1.1({0x1ca3ea0,
0xc000158000}, 0xc0002fc200)
        github.com/caddyserver/caddy/v2@v2.5.1/modules/caddyhttp/routes.go:244 +0x322
github.com/caddyserver/caddy/v2/modules/caddyhttp.HandlerFunc.ServeHTTP(0xc0002c2c00?,
{0x1ca3ea0?, 0xc000158000?}, 0x18a8200?)
        github.com/caddyserver/caddy/v2@v2.5.1/modules/caddyhttp/caddyhttp.go:57 +0x2f
github.com/caddyserver/caddy/v2/modules/caddyhttp.(*Server).enforcementHandler(0x0?,
{0x1ca3ea0?, 0xc000158000?}, 0xc00069c6e0?, {0x1c9a3c0?, 0xc00053b0a0?})
        github.com/caddyserver/caddy/v2@v2.5.1/modules/caddyhttp/server.go:318 +0x252
github.com/caddyserver/caddy/v2/modules/caddyhttp.
(*Server).wrapPrimaryRoute.func1({0x1ca3ea0?, 0xc000158000?}, 0x49b737?)
        github.com/caddyserver/caddy/v2@v2.5.1/modules/caddyhttp/server.go:294 +0x3b
github.com/caddyserver/caddy/v2/modules/caddyhttp.HandlerFunc.ServeHTTP(0xc0002b6bb0?,
{0x1ca3ea0?, 0xc000158000?}, 0xc0002fc200?)
        github.com/caddyserver/caddy/v2@v2.5.1/modules/caddyhttp/caddyhttp.go:57 +0x2f
github.com/caddyserver/caddy/v2/modules/caddyhttp.(*Server).ServeHTTP(0xc000024360,
{0x1ca3ea0, 0xc000158000}, 0xc0002fc200)
        github.com/caddyserver/caddy/v2@v2.5.1/modules/caddyhttp/server.go:230 +0xac6
net/http.serverHandler.ServeHTTP({0xc000314120?}, {0x1ca3ea0, 0xc000158000}, 0xc0002fc000)
        net/http/server.go:2916 +0x43b
net/http.(*conn).serve(0xc0000baaa0, {0x1ca5790, 0xc0001aa630})
        net/http/server.go:1966 +0x5d7
created by net/http.(*Server).Serve
        net/http/server.go:3071 +0x4db
```

You can see the server continues to handle and serve requests properly, according to its configuration:

```
$ curl -v "http://localhost:8888/"
*   Trying 127.0.0.1:8888...
* Connected to localhost (127.0.0.1) port 8888 (#0)
> GET / HTTP/1.1
> Host: localhost:8888
> User-Agent: curl/7.81.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Server: Caddy
< Date: Wed, 03 Aug 2022 05:00:39 GMT
< Content-Length: 5
<
* Connection #0 to host localhost left intact
Hello
```

Additionally, the assigned severity base score of 7.5 is absurdly high.

The CVE entry claims the attack vector (AV) is Network (AV:N), but no network exploit has been demonstrated. As we have shown, the bug is caused by a faulty server configuration by an administrator; no exploit where an attacker successfully loads a faulty configuration over a network has been demonstrated in this CVE entry. It should be AV:L or AV:P.

The attack complexity (AC) metric was assigned Low (AC:L), however, this bug requires external factors outside an attacker's control to be put into force. It should have a complexity of High (AC:H).

The privileges required (PR) metric was assigned None (PR:N) but an attacker needs sufficient (usually administrator/root) privileges to load an unauthorized configuration. It should be High (PR:H).

The user interaction (UI) metric was assigned None (UI:N) but a user must send a certain kind of matching request for the error to be revealed; by clicking a link or otherwise. It should be Required (UI:R).

The availability impact (A) metric was assigned High (A:H), however we have demonstrated why that is wrong. It should be None (A:N).

These corrections take the base score down to 0.3.

The most severe bug-like behavior in the report is the unpleasant panic in the logs and an empty HTTP response instead of a pleasant message in the logs and an error HTTP response. This is an "ordinary" bug that is not a security vulnerability at all: the web server is properly executing its configuration from an authorized user.

We recommend that this irrelevant CVE entry be withdrawn to keep the database focused on legitimate security bugs and meaningful for other users.

🟢 **caddyserver** locked as **resolved** and limited conversation to collaborators on Aug 3

Assignees

👤 mholt

Labels

bug 🐞

Projects

None yet

Milestone

v2.5.2

Development

No branches or pull requests

---

3 participants