Partial password leak over DNS on HTTP redirect

Share:

TIMELINE

mszpl submitted a report to curl. May 15th (3 ye

Summary:

From version 7.62 curl and curllib leaks part of user credentials in the plain text DNS request. This happens if the server makes redirect, both 301 and 302 to a relat path (eg header 'Location: /login'). It is NOT an issue in case of absolute redirection (eg header 'Location: https://domain.tld/login').
I was able to make curl/curlib to send a password that started with @ but I believe that more abuse is possible with this attack.
What makes is worst is that for eg occasionally run/daemon scripts with curl and authorization credentials this can be triggered by a remote server by switching between absolute/relative without any change on client-side.
User secrets are sent in plain text and anybody in the middle can record them. User secrets are sent to the DNS server and can be recorded there.

Steps To Reproduce:

1. Use curl > 7.61 (tested on all from 7.62 to 7.70 and I was able to exploit it)
2. Find a server with relative redirection (eg https://mareksz.gq/301 or https://mareksz.gq/302)
3. Run 'curl https://mareksz.gq/302 -v -L -u saduser:@S3cr3t'

Supporting Material/References:

Logs from running above steps:

/ $ curl -V
curl 7.66.0-DEV (x86_64-pc-linux-gnu) libcurl/7.66.0-DEV OpenSSL/1.1.1d zlib/1.2.11 nghttp2/1.39.2
Release-Date: [unreleased]
Protocols: dict file ftp ftps gopher http https imap imaps pop3 pop3s rtsp smb smbs smtp smtps telnet tftp
Features: AsynchDNS HTTP2 HTTPS-proxy IPv6 Largefile libz NTLM NTLM_WB SSL TLS-SRP UnixSockets
/ $ curl https://mareksz.gq/302 -v -L -u saduser:@S3cr3t

- Trying 194.182.85.202:443...
- TCP_NODELAY set
- Connected to mareksz.gq (194.182.85.202) port 443 (#0)
- ALPN, offering h2
- ALPN, offering http/1.1
- successfully set certificate verify locations:
- CAfile: /etc/ssl/certs/ca-certificates.crt CApath: none
- TLSv1.3 (OUT), TLS handshake, Client hello (1):
- TLSv1.3 (IN), TLS handshake, Server hello (2):
- TLSv1.2 (IN), TLS handshake, Certificate (11):
- TLSv1.2 (IN), TLS handshake, Server key exchange (12):
- TLSv1.2 (IN), TLS handshake, Server finished (14):
- TLSv1.2 (OUT), TLS handshake, Client key exchange (16):
- TLSv1.2 (OUT), TLS change cipher, Change cipher spec (1):
- TLSv1.2 (OUT), TLS handshake, Finished (20):
- TLSv1.2 (IN), TLS handshake, Finished (20):
- SSL connection using TLSv1.2 / ECDHE-RSA-AES256-GCM-SHA384
- ALPN, server accepted to use http/1.1
- Server certificate:
- subject: CN=mareksz.gq
- start date: Apr 27 10:32:33 2020 GMT
- expire date: Jul 26 10:32:33 2020 GMT
- subjectAltName: host "mareksz.gq" matched cert's "mareksz.gq"
- issuer: C=US; O=Let's Encrypt; CN=Let's Encrypt Authority X3
- SSL certificate verify ok.
- Server auth using Basic with user 'saduser'
  > GET /302 HTTP/1.1
  > Host: mareksz.gq
  > Authorization: Basic c2FkdXNlcjpAUzNjcjN0
  > User-Agent: curl/7.66.0-DEV
  > Accept: /
- Mark bundle as not supporting multiuse < HTTP/1.1 302 Moved Temporarily < Server: nginx < Date: Fri, 15 May 2020 08:32:59 GMT < Content-Type: text/html Content-Length: 138 < Connection: keep-alive < Location: /goto302 <
- Ignoring the response-body
- Connection #0 to host mareksz.gq left intact
- Issue another request to this URL: 'https://saduser@S3cr3t@mareksz.gq/goto302'
- Could not resolve host: S3cr3t@mareksz.gq
- Closing connection 1 curl: (6) Could not resolve host: S3cr3t@mareksz.gq

Trafic pcap'ed:

/ $ tcpdump 'udp' -vv
X.X.X.X:X IP (tos 0x0, ttl 255, id 57291, offset 0, flags [none], proto UDP (17), length 63)
> : [udp sum ok] 27230+ A? S3cr3t@mareksz.gq. (35)
X.X.X.X:X IP (tos 0x0, ttl 255, id 55686, offset 0, flags [none], proto UDP (17), length 63)

X.X.X.X:X IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto UDP (17), length 136)
> : [udp sum ok] 51727 NXDomain q: AAAA? S3cr3t@mareksz.gq. 0/1/0 ns: gq. SOA a.ns.gq. info.equatorialguineadomains.com. 1589532235 10800 3600 60480(
(108)

- [attachment / reference] Attached Wireshark screenshot with leaked creds.

Impact

I believe it is rather high. Third-party have control over it part of your credentials are being sent over the network in plain text to the DNS server.

1 attachment:
F829171: curl_leaks_secret_dns.png

mszpl posted a comment. May 15th (3 ye
I would like to add to the "Impact" part that libcurl also contains this issue. So ALL projects that are using libcurl for networking will also have this security issue. Wi
libcurl > 7.61 (tested on 7.62 / 7.66 / and latest 7.70).

Example code that compiled with libcurl > 7.61 leaks user credentials via DNS request:

include <curl/curl.h>

int main(int argc, char *argv[])*
{
CURL curl = curl_easy_init();
curl_easy_setopt(curl, CURLOPT_VERBOSE, 1);
curl_easy_setopt(curl, CURLOPT_FOLLOWLOCATION, 1);
curl_easy_setopt(curl, CURLOPT_USERPWD, "saduser:@S3cr3t");
curl_easy_setopt(curl, CURLOPT_URL, "https://mareksz.gq/302");
int result = curl_easy_perform(curl);
return result;
}

bagder curl staff posted a comment. May 15th (3 ye
Thank you. I'm pretty sure this issue has been identified and fixed already in curl:
https://github.com/curl/curl/commit/600a8cded447cd7118ed50142c576567c0cf5158 I'll investigate closer in a bit.

mszpl posted a comment. May 15th (3 ye
Thank you, for a quick response. Retested from the master branch, and (sadly) yes - looks like it is fixed. Creds are not sent any more with DNS resolution.
But shouldn't we issue CVE for 7.61 - 7.70 (if there isn't one already reported/made)?

Have a nice weekend! ;)

bagder curl staff posted a comment. May 15th (3 ye
This is what happened before the fix:

The user credentials set with `CURLOPT_USERPWD` updated the URL object, **forgetting to ask for URL encoding**. When a *relative* HTTP redirect is about to get handle
libcurl will first basically generate a full absolute URL and then "apply" the redirect on that.

It would then make the full URL `http://[user]:[password]@host/path` and for this bug, lets say we have user `user` and password is `@password`:

```
http://user:@password@example.com/path
```

Applying a relative redirect only changes the path part, so it ends up like:

```
http://user:@password@example.com/redir/newpath
```

Then libcurl parses the full new URL back into pieces again, and due to the wrong encoding from before, it now thinks the password is zero bytes long and that `password@example.com` is actually the host name it should use...

Some observations:

1. it doesn't leak the entire password here, just the part on the right sight of the `@` sign, and there's no additional clues how much of the password that is missin
2. It doesn't resolve the same domain anymore so this bad DNS resolve will not end up in the DNS server of `example.com` (which presumably hosted the redirect
wanted the credentials). Not sure it matters much, but it seems hard for a remote "player" to take advantage of this flaw.
3. Since DNS is still usually clear-text, sitting on the same local network as this libcurl-using app will be enough to potentially detect partial passwords.
4. If DoH is used by libcurl, this flaw won't leak a clear-text resolve but will instead ask the DoH server to resolve the wrong name over HTTPS.
5. Is `@` actually legal in host name or should we error in the parser there?
6. I clearly didn't at all consider the security implications when I fixed this bug!
7. Due to the potential leakage of sensitive data I think we should make a CVE out of this.

mszpl posted a comment. May 16th (3 ye
@1. Yes, additionally using @ in user part (which is most common, as often user names are just emails) was **not** causing DNS resolution, @ needed to be in passwo
part. Probably : (colon) from user-password part that's always after first @ (from mail) fails on internal curl domain parsing.
@2. True, but what is strange both OS and DNS servers did not complain about domain with @ inside. Just tried do register such domain but with no success ;).
Probably being able to inject `password@example.com` somewhere to this chain would cause potentially malicious redirection.
@3. This was my main concern.
@4. Still not 100% happy with the DNS server receiving part of the password.
@5. Hmm. This would prevent the DNS resolution by failing internally before it (as it did with the colon in point @1). But as I believe that url encoding credential pa
cleaner solution, not introducing possible regression for somebody that uses such cases (OS and DNS tried to resolve that!).
@6/7. I spotted it when updating libcurl from 7.51 -> 7.69. First I believed that there was some minor change and it can be somehow fixed by a change in our client
(eg. we need to add a `domain` to 'Set-' cookie format, as they were not sent without it anymore with new lib). Idea was to run `curl -L -u` and copy how it is done
tool, but the issue was the same. Running verbose shows the stuff that looked interesting from security perspective:

bagder curl staff posted a comment. May 18th (3 ye

Suggested plan: I write up a security advisory for this flaw and request a CVE in time so that we can announce the flaw and the advisory in association with the pen
next release. Due to ship on June 24, 2020.

Does anyone (reading this) think we need to handle this with more urgency?

bagder curl staff updated the severity from High (8.6) to Medium (5.5). May 18th (3 ye

bagder curl staff posted a comment. May 18th (3 ye

This is my current Security Advisory draft. Also attached below.

Partial password leak over DNS on HTTP redirect

Project curl Security Advisory, June 24th 2020 -
Permalink

VULNERABILITY

libcurl can be tricked to prepend a part of the password to the host name before it resolves it, potentially leaking the partial password over the network and to the DNS server(s).

libcurl can be given a username and password for HTTP authentication when requesting an HTTP resource - used for HTTP Authentication such as Basic, Digest, NTLM and similar. The credentials are set, either together with `CURLOPT_USERPWD` or separately with `CURLOPT_USERNAME` and `CURLOPT_PASSWORD`. Important detail: these strings are given to libcurl as plain C strings and they are not supposed to be URL encoded.

In addition, libcurl also allows the credentials to be set in the URL, using the standard RFC 3986 format: `http://user:password@host/path`. In this case, the name and password are URL encoded as that's how they appear in URLs.

If the options are set, they override the credentials set in the URL.

Internally, this is handled by storing the credentials in the "URL object" so that there is only a single set of credentials stored associated with this single URL.

When libcurl handles a relative redirect (as opposed to an absolute URL redirect) for an HTTP transfer, the server is only sending a new path to the client and that path is applied on to the existing URL. That "applying" of the relative path on top of an absolute URL is done by libcurl first generating a full absolute URL out of all the components it has, then it applies the redirect and finally it deconstructs the URL again into its separate components.

This security vulnerability originates in the fact that curl did not correctly URL encode the credential data when set using one of the `curl_easy_setopt` options described above. This made curl generate a badly formatted full URL when it would do a redirect and the final re-parsing of the URL would then go bad and wrongly consider a part of the password field to belong to the host name.

The wrong host name would then be used in a name resolve lookup, potentially leaking the host name + partial password in clear text over the network (if plain DNS was used) and in particular to the used DNS server(s).

The password leak is triggered if an at sign (`@`) is used in the password field, like this: `passw@rd123`. If we also consider a user `dan`, curl would generate a full URL like:

`https://dan:passw@rd123@example.com/path`

... while a correct one should have been:

`https://dan:passw%40rd123@example.com/path`

... when parsing the wrongly generated URL, libcurl would end up with user `dan` and password `pass` talking to the host `rd123@example.com`. That bad host name would then be passed on to the name resolver function in use (and for all typical cases return a "cannot resolve host name" error).

There's no hint in the name resolve as to how large portion of the password that is actually prepended to the host name (ie an observer won't know how much data there was on the left side of the `@`), but it can of course be a significant enough clue for an attacker to figure out the rest.

We are not aware of any exploit of this flaw.

INFO

Requirements to trigger this flaw.

This bug was brought in commit
46e164069d, first shipped in
curl 7.62.0.

This flaw can happen to users of the curl tool as well as for applications
using libcurl.

This bug was reported and inadvertently fixed and pushed to the public source
respository before anyone realized its security impact.

The effects of this flaw is somewhat reduced if DNS-over-HTTPS is used, since
then at least the name won't be observable on the network by a passive
by-stander but only by the DoH server.

The Common Vulnerabilities and Exposures (CVE) project has assigned the name
CVE-2020-RRRR to this issue.

CWE-200: Exposure of Sensitive Information to an Unauthorized Actor

Severity: 5.5 (Medium)

AFFECTED VERSIONS

- Affected versions: libcurl 7.62.0 to and including 7.70.0
- Not affected versions: libcurl < 7.62.0

libcurl is used by many applications, but not always advertised as such.

THE SOLUTION

A fix for CVE-2020-RRRR

RECOMMENDATIONS

We suggest you take one of the following actions immediately, in order of
preference:

A - Upgrade curl to version 7.71.0

B - Apply the patch on your libcurl version and rebuild

C - Disable `CURLOPT_FOLLOWLOCATION` or redirects to HTTP(S).

TIMELINE

This issue was first reported to the curl project on May 14, 2020. The initial
fix was done, verified and pushed to git on the same day. (As a regular
non-security related fix.)

On May 15, 2020, the bug was reported again but then with the security impact
highlighted.

This advisory was posted on June 24th 2020.

CREDITS

The security issue was reported by Marek Szlagor. The initial bug report was
done by Gregory Jefferis and Jeroen Ooms. Patched by Daniel Stenberg.

Thanks a lot!

1 attachment:
F832846: CVE-2020-RRRR.md

mszpl posted a comment. May 19th (3 ye
It looks good, thank you!

bagder curl staff added weakness "Information Disclosure" and removed weakness "Insufficiently Protected Credentials". May 19th (3 ye

bagder curl staff updated CVE reference to **CVE-2020-8169**. May 19th (3 ye

bagder curl staff changed the report title from **curl and curllib send user credentials in dns request on relative redirect** to **Partial password leak over DNS on HTTP redirect**. May 19th (3 ye

bagder curl staff changed the status to **Triaged**. Jun 2nd (3 ye

url rewarded mszpl with a **$400** bounty. Jun 4th (3 ye
The curl security team has decided to reward hacker @mszpl with the amount of 400 USD for finding and reporting this issue. Many thanks for your great work!

This report was technically not within policy, but the curl security team has decided that it was still a valuable report that highlighted a part of the issue that was previously missed. Thus still rewarding the reporter as a sign of our gratitude, just maybe a little lower than it could otherwise have been.

Thank you @mszpl!

mszpl posted a comment. Jun 4th (3 ye

- Attack Complexity - low
- Privileges Required - none
- User Interaction - none (user is not doing anything special, remote server have control over this issue)
- Scope - unchanged
- Confidentiality / Integrity / Availability - low

Eg.
https://www.first.org/cvss/calculator/3.0#CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:L

bagder curl staff posted a comment. Jun 4th (3 ye

Because I consider it to be attack complexity (AC) high. It's really hard for an attacker to make the user to pick a password with an @-letter in it for a site that the attacker controls and can make a redirect for and then also be able to actually get the leaked information.

mszpl posted a comment. Jun 4th (3 ye

I understand you point and need to agree, thanks for explanation!

bagder curl staff closed the report and changed the status to **Resolved**. Jun 24th (2 ye

Published!

bagder curl staff requested to disclose this report. Nov 5th (2 ye

Let's disclose the details for the world

This report has been disclosed. Dec 5th (2 ye