7 CVE-2021-22890: TLS 1.3 session ticket proxy host mixup

Share:

TIMELINE

mingtao submitted a report to curl. Mar 17th (2 ye

Summary:

(I don't think that this can be easily exploitable, but I am submitting it as a security issue for precaution. I am not looking for a bounty.)

Commit 549310e907e82e44c59548351d4c6ac4aaada114 enables session resumption with TLS 1.3. Curl connections maintain two SSL contexts, one for the pro and one for the destination. However, curl incorrectly stores session tickets issued by an TLS 1.3 HTTPS proxy under the non proxy context.

The issue is that the logic inside `Curl_ssl_addsessionid` that chooses which context to store the tickets under is incorrect under TLS 1.3.

Code 228 Bytes Wrap lines Copy Dow

```
1 const bool isProxy = CONNECT_PROXY_SSL();
2 struct ssl_primary_config * const ssl_config = isProxy ?
3   &conn->proxy_ssl_config :
4   &conn->ssl_config;
5 const char *hostname = isProxy ? conn->http_proxy.host.name :
6   conn->host.name;
```

Code 127 Bytes Wrap lines Copy Dow

```
1 #define CONNECT_PROXY_SSL()\
2   (conn->http_proxy.proxytype == CURLPROXY_HTTPS &&\
3   !conn->bits.proxy_ssl_connected[sockindex])
```

One of the major differences between how TLS session tickets are issued between TLS 1.3 and prior versions of TLS is that TLS 1.3 issues session tickets in a *pos* handshake message. What this means in practice is that TLS 1.3 tickets are delivered in the first call to `SSL_read()`, rather than being issued as part of `SSL_conne` Consequently, `CONNECT_PROXY_SSL()` will see that the proxy has already been connected (since the call to `SSL_connect()` to the proxy was completed), so the call `Curl_ssl_addsessionid` believes the `isProxy` is `false`, and it stores the ticket under the non proxy context.

After the `CONNECT` call returns successfully, a connection to the original destination will be made through the established TCP tunnel. If the original destination us https, another TLS handshake will be made. During this TLS handshake, the curl client offers the session ticket of the *proxy* to the destination.

If the proxy is malicious, at this point it could decide to terminate the TLS handshake to the upstream. Since the proxy has the corresponding session ticket key (it the entity that issued the ticket, after all), it can complete the client -> destination TLS handshake through a resumption. Normally, this would result in a full man ir middle, as TLS certificates are not exchanged as part of a resumed connection. However, curl already performs some of its own certificate validation outside of OpenSSL in `ossl_connect_step3`, which largely mitigates this vulnerability.

The certificate validation that curl performs includes steps such as (1) checking if the certificate was self signed and (2) ensuring that the certificate contains a sub that matches the destination. The certificate of the proxy is stored in the `SSL_SESSION` that was used for resumption, so curl will attempt to perform these validat against the proxy certificate.

Steps To Reproduce:

I've attached a reproducer in this report.

- `server_that_fails_on_ticket.c` is a simple TLS server (listening on port 12345) that will send an alert if it receives a session resumption attempt. Under norm circumstances, curl should never be sending a ticket when connecting through a proxy, since it has never connected to this destination before. With this bug, y should be able to observe that the server receives a ticket on the first connection regardless.
- `https_proxy.c` is a extremely rudimentary implementation of a HTTPS proxy (listening on port 12346), that only uses TLS 1.3. If a special proxy header `Mitm:` passed, then the proxy will attempt to terminate the TLS connection itself, acting as a man in the middle.
- `proxy_ca.pem` is the CA file that signs the proxy cert, `haxx.se.pem`
- `haxx.se.pem` is the TLS certificate that the proxy uses. Notice that it has the identities: `localhost` and `haxx.se`.

Demonstrating that curl sends the proxy ticket to the original destination.

1. Run `server_that_fails_on_ticket`. This will listen on port 12345
2. Run `https_proxy`. This will listen on port 12346
3. Run `curl --proxy-cacert proxy_ca.pem -x 'https://localhost:12346' 'https://localhost:12345'`
4. Notice that the curl client receives a TLS alert, and that "Received a TLS 1.3 ticket resumption attempt" is printed on the server.

Demonstrating the very limited MiTM possibility.

1. Run `https_proxy`. This will listen on port 12346
2. Run `curl --proxy-cacert proxy_ca.pem --proxy-header 'Mitm: 1' -x 'https://localhost:12346' 'https://haxx.se'`
3. Notice that "MITM" is returned, and no certificate error is thrown.

The MITM is only possible because `haxx.se` is listed as one of the subjects in the proxy certificate. Curl's certificate validation passes: (1) the proxy cert is not self signed and (2) the name haxx.se is present in the certificate is "presented" by the original destination.

Impact

In a very specific environment (perhaps a corporate environment where all access to the internet requires going through an HTTPS proxy), an attacker that can iss trusted proxy certificate may be able to man in the middle connections established with libcurl, even if curl explicitly does not include the proxy CA in the trust sto normal destinations.

1 attachment:

Thanks, we'll investigate!

badger curl staff posted a comment. Mar 18th (2 ye

@mingtao did you by any chance look into how a patch for this problem could look like?

badger curl staff posted a comment. Mar 18th (2 ye

I'm very impressed by the report, the level of detail and the source code provided to reproduce this. Thanks @mingtao for submitting a first-class issue.

I believe I've wrapped my head around the problem now but I have not yet worked out a fix for it. This thing with the ticket coming after the handshake requires something proper thinking of how to deal with it correctly. I think we also need to check other TLS backends that support HTTPS proxies to see if they possibly ex the same problem.

To assess if this is a security problem I think this is a crucial detail:

> The MITM is only possible because `haxx.se` is listed as one of the subjects in the proxy certificate.

Only a malicious proxy would do this and curl would then be allowed to use and accept the TLS connection to this to begin with. This made me think of the problem more general terms.

Can there be a legitimate expectation of security when doing https over a malicious (https) proxy?

If there can't, then this isn't actually a security problem as long as it relies on the proxy doing rogue things.

If there can, and since TLS is end-to-end verified and secure I *think* we can work with that expectation, can't we? Then we should probably consider this a security issue.

mingtao posted a comment. Mar 18th (2 ye

Apologies for the delayed response!

> @mingtao did you by any chance look into how a patch for this problem could look like?

> This thing with the ticket coming after the handshake requires something proper thinking of how to deal with it correctly.

I'd like to think of the ticket callback interface the same as any other observer interface, where a caller subscribes to a (potentially) unbounded asynchronous strea events.

A close analogy I could think of would be `CURLOPT_WRITEFUNCTION`, which is also an asynchronous stream. If I wanted to share the same callback to use for multiple connections, I would need to allocate independent connection specific state, and use `CURLOPT_WRITEDATA` to associate my connection-specific state with the inte curl handle.

The session ticket interface is very similar. We have a single implementation `ossl_new_session_cb` (analogy: a single `CURLOPT_WRITEFUNCTION`), being shared betwe multiple SSL sessions (analogy: multiple curl handles).

OpenSSL invokes the callback, passing the `SSL*` pointer corresponding to the specific SSL session that issued the ticket. This serves as our analogous `userdata` since:

- You can get the corresponding SSL context that it was allocated from, using something like `SSL_get_SSL_CTX`. You could compare the resulting ctx pointer dire
- Most OpenSSL objects allow you to store arbitrary data along with the OpenSSL object (`SSL_set_ex_data`). Another possible implementation would be to hav store information on the context when it allocates the `SSL` object (is this for the proxy? is this for the destination).

I think the issue here is that `Curl_ssl_addsessionid` and `Curl_ssl_getsessionid` tries to infer connection-specific information *implicitly*, by checking the connect state flags. If `Curl_ssl_addsessionid` "lifts" the responsibility of determining which context to associate the session to the caller (e.g. take in a separate `isProxy` parameter), then the implementation can look something like:

Code 383 Bytes Wrap lines Copy Dow

```
static int ossl_new_session_cb(SSL* ssl, SSL_SESSION* ssl_sessionid) {
    // ....
    SSL_CTX* ctx = SSL_get_SSL_CTX(ssl);
    bool isProxy;
    if (ctx == connection_specific_data->proxy_ssl_context) {
       isProxy = true;
    } else if (ctx == connection_specific_data->context) {
       isProxy = false
    } else {
       assert(false);
    }

    Curl_ssl_addsessionid(...., isProxy);
}
```

Obviously, this might be easier said than done, but the idea should work.

> I'm very impressed by the report, the level of detail and the source code provided to reproduce this. Thanks @mingtao for submitting a first-class issue.

Thanks for your kind words @badger. :)

> Can there be a legitimate expectation of security when doing https over a malicious (https) proxy?

> If there can, and since TLS is end-to-end verified and secure I think we can work with that expectation, can't we? Then we should probably consider this a securit issue.

I agree with your reasoning.

bagder curl staff posted a comment. Mar 19th (2 ye

Thanks.

So this is how I'd like to play this, and please @mingtao let me know if you have any issues with this plan.

0. We agree that this is a security problem. I think this is already done more or less. If any team member reading this disagrees, please speak up sooner rather tha later.
1. We work out a patch for this problem, and I attach my first attempt here (see link below) - I didn't actually test this against the exploit code myself yet but I belie has the right spirit. @mingtao if you have an easy way to rerun the test code with the patch applied I'll be happy, otherwise I'll do it myself within a day or two. (patch includes fixes for other TLS backends as well, and I will at least make sure they build and run error-free with those before it ships.)
2. I want us to ship an advisory and fix in associated with the pending 7.76.0 release that is due to ship on March 31st (12 daus away), so there's not a lot of time.
3. I will write up a first take on security advisory for this problem and submit here. Probably within a few hours.
4. I want to pre-alert the 'distros' mailing list a few days/a week in advance of our planned release date. And then we need both a patch and an advisory - they dor have to be the final versions but should at least work and be accurate.
5. I know you said you are not "looking for a bounty" for this, but I am going to insist that you let us reward you for your hard work and excellent reporting. We have taken the discussion yet, but I would think we're looking at upwards a thoursand USD so nothing to sneeze at if you ask me.

1 attachment:
F1235718: 0001-vtls-add-an-isproxy-argument-to-Curl_ssl_getsessioni.patch

bagder curl staff changed the status to **Triaged**. Mar 19th (2 ye

bagder curl staff posted a comment. Mar 19th (2 ye

Here's my first take at a security advisory. Please read the description and see if it makes sense to you and that it actually matches the problem identified. The link the patch doesn't yet exist since we haven't yet agreed on one.

If there are no objections, I will request CVE for this this problem later today.

TLS 1.3 session ticket proxy host mixup

Project curl Security Advisory, March 31st 2021 -
Permalink

VULNERABILITY

Enabled by default, libcurl supports the use of TLS 1.3 session tickets to resume previous TLS sessions to speed up subsequent TLS handshakes.

When using a HTTPS proxy and TLS 1.3, libcurl can confuse session tickets arriving from the HTTPS proxy but work as if they arrived from the remote server and then wrongly "short-cut" the host handshake. The reason for this confusion is the modified sequence from TLS 1.2 when the session ids would provided only during the TLS handshake, while in TLS 1.3 it happens post hand-shake and the code was not updated to take that changed behavior into account.

When confusing the tickets, a HTTPS proxy can trick libcurl to use the wrong session ticket resume for the host and thereby circumvent the server TLS certificate check and make a MITM attack to be possible to perform unnoticed.

This flaw can allow a malicious HTTPS proxy to MITM the traffic.

We are not aware of any exploit of this flaw.

INFO

This flaw has existed in libcurl since commit
549310e907e in libcurl 7.63.0,
released on December 12, 2018.

It only exists when libcurl is built to use OpenSSL or one of its many forks. It can only trigger when TLS 1.3 is used with the HTTPS proxy and not with earlier TLS versions.

It might be worth highlighting that an HTTPS proxy is a proxy which libcurl communicates with over TLS specifically, and then speaks HTTPS through, making it two layers of TLS. It is different than the more common HTTP proxy setup, where libcurl just does normal TCP with the proxy.

The Common Vulnerabilities and Exposures (CVE) project has assigned the name CVE-2021-XXXX to this issue.

CWE-290: Authentication Bypass by Spoofing

Severity: Low

AFFECTED VERSIONS

- Affected versions: curl 7.63.0 to and including 7.75.0

advertised as such.

THE SOLUTION

Make sure the proxy/host distinction is done correctly.

A fix for CVE-2021-XXXX

(The patch URL will change in the final published version of this advisory)

RECOMMENDATIONS

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

A - Upgrade libcurl to version 7.76.0

B - Apply the patch to your local version

C - Avoid TLS 1.3 with HTTPS proxies

TIMELINE

This issue was reported to the curl project on March 17, 2021.

This advisory was posted on March 31st 2021.

CREDITS

This issue was reported by Mingtao Yang. Patched by Daniel Stenberg.

Thanks a lot!

1 attachment:
F1235777: CVE-2021-XXXX.md

bagder curl staff posted a comment. Mar 19th (2 ye

@mingtao: also, please let me know if you want to be credited like this or any other way or with your associated organization or what not

mingtao posted a comment. Updated Mar 19th (2 ye

Hi,

I've tried the patch you attached and it doesn't completely fix the issue. The patch adds an `isProxy` parameter to `Curl_ssl_getsessionid`, but it doesn't fix `Curl_ssl_addsessionid`. Control flow reaches here:

Code 534 Bytes Wrap lines Copy Dow

```
1   isproxy = SSL_get_ex_data(ssl, proxy_idx) ? TRUE : FALSE;
2
3   if(SSL_SET_OPTION(primary.sessionid)) {
4     bool incache;
5      // ...
6     if(isproxy)
7       incache = FALSE;
8     else
9      //....
10
11    if(incache) {
12       // ...
13    }
14
15    if(!incache) {
16      if(!Curl_ssl_addsessionid(data, conn, ssl_sessionid,
17                                0 /* unknown size */, sockindex)) {
18        /* the session has been put into the session cache */
19        res = 1;
20      }
21      else
22        failf(data, "failed to store ssl session");
23    }
```

which still causes the session to be stored in the wrong ssl session context.

I can, however, confirm that the isproxy variable stored on the SSL EX_DATA is correct though. I've tweaked your patch a bit to include the isProxy parameter to `Curl_ssl_addsessionid` and it now works and the mitm fails with a certificate error, as expected (since the proxy is no longer able to terminate the connection).

Code 1.84 KiB Wrap lines Copy Dow

```
1  diff --git a/lib/vtls/openssl.c b/lib/vtls/openssl.c
2  index 3b778fd4f..a48ec474a 100644
3  --- a/lib/vtls/openssl.c
4  +++ b/lib/vtls/openssl.c
5  @@ -2480,7 +2480,7 @@ static int ossl_new_session_cb(SSL *ssl, SSL_SESSION *ssl_sessionid)
6       }
7
8       if(!incache) {
9  -      if(!Curl_ssl_addsessionid(data, conn, ssl_sessionid,
```

```
13          res = 1;
14  diff --git a/lib/vtls/vtls.c b/lib/vtls/vtls.c
15  index b3f48a34f..c3ef6ce4c 100644
16  --- a/lib/vtls/vtls.c
17  +++ b/lib/vtls/vtls.c
18  @@ -485,6 +485,7 @@ void Curl_ssl_delsessionid(struct Curl_easy *data, void *ssl_sessionid)
19    */
20   CURLcode Curl_ssl_addsessionid(struct Curl_easy *data,
21                                  struct connectdata *conn,
22  +                               bool isProxy,
23                                  void *ssl_sessionid,
24                                  size_t idsize,
25                                  int sockindex)
26  @@ -497,7 +498,6 @@ CURLcode Curl_ssl_addsessionid(struct Curl_easy *data,
27     int conn_to_port;
28     long *general_age;
29   #ifndef CURL_DISABLE_PROXY
30  -  const bool isProxy = CONNECT_PROXY_SSL();
31     struct ssl_primary_config * const ssl_config = isProxy ?
32       &conn->proxy_ssl_config :
33       &conn->ssl_config;
34  diff --git a/lib/vtls/vtls.h b/lib/vtls/vtls.h
35  index e3d14d6da..2b43e7744 100644
36  --- a/lib/vtls/vtls.h
37  +++ b/lib/vtls/vtls.h
38  @@ -246,6 +246,7 @@ bool Curl_ssl_getsessionid(struct Curl_easy *data,
39    */
40   CURLcode Curl_ssl_addsessionid(struct Curl_easy *data,
41                                  struct connectdata *conn,
42  +                               const bool isProxy,
43                                  void *ssl_sessionid,
44                                  size_t idsize,
45                                  int sockindex);
```

(This diff does not contain the changes for the other backends. It additionally introduces a warning since `sockindex` is no longer used)

bagder curl staff posted a comment. Mar 19th (2 ye

> This diff does not contain the changes for the other backends. It additionally introduces a warning since sockindex is no longer used

Thanks! I'll merge your patch, fix the warnings and submit a new full patch in a bit.

mingtao posted a comment. Mar 19th (2 ye

> So this is how I'd like to play this, and please @mingtao let me know if you have any issues with this plan.

> I know you said you are not "looking for a bounty" for this, but I am going to insist that you let us reward you for your hard work and excellent reporting. We haver taken the discussion yet, but I would think we're looking at upwards a thoursand USD so nothing to sneeze at if you ask me.

Should you choose to do this, you will likely see a donation to the Open Source Collective 501c6 for an amount of `$BOUNTY - $US_GOVERNMENT_TAXES`. From all of the bureaucracy that you've encountered in your US visa application, I'd like to imagine that you wouldn't want to indirectly sponsor those clowns. ;)

In all seriousness though, as an avid curl user (both at $DAYJOB and my own personal life), I have no problems with not taking a bounty. I haven't had the time to contribute to curl meaningfully (in code), but I can certainly help report bugs when I see them.

> Here's my first take at a security advisory. Please read the description and see if it makes sense to you and that it actually matches the problem identified. The li the patch doesn't yet exist since we haven't yet agreed on one.

My only suggestion is to somehow highlight the fact that this is unlikely to be exploited, since curl performs its own certificate validation even on resumed connections. Otherwise, LGTM

bagder curl staff posted a comment. Mar 19th (2 ye

> My only suggestion is to somehow highlight the fact that this is unlikely to be exploited, since curl performs its own certificate validation even on resumed connections

Uhm. Then I'm not sure I follow. If curl's certificate validation is still done, how exactly can you MITM the connection then? I thought the proxy would trick curl to resume the session and bypass the validation.

mingtao posted a comment. Updated Mar 19th (2 ye

> how exactly can you MITM the connection then

In the PoC I gave, the proxy certificate contained the domain of the mitm target.

When curl later performs its certificate checks against the target, it uses the session stored in the proxy SSL_SESSION. Curl doesn't perform a full chain verificati it checks that the target domain is contained within the cert. I've specifically set it up in the PoC such that it would pass the validation.

For a non-malicious proxy that only contains its own hostname, it will likely fail with a certificate error.

bagder curl staff posted a comment. Mar 19th (2 ye

easier to make such proxies and then trick curl users into actually using such those malicious things.

bagder curl staff posted a comment. Mar 19th (2 ye

Updated patch. I also test-built it with mbedTLS, bearssl, GnuTLS and wolfSSL.

1 attachment:
F1236017: v2-0001-vtls-add-isproxy-argument-to-Curl_ssl_get-addsess.patch

bagder curl staff posted a comment. Mar 19th (2 ye

Advisory slightly updated.

TLS 1.3 session ticket proxy host mixup

Project curl Security Advisory, March 31st 2021 -
Permalink

VULNERABILITY

Enabled by default, libcurl supports the use of TLS 1.3 session tickets to resume previous TLS sessions to speed up subsequent TLS handshakes.

When using a HTTPS proxy and TLS 1.3, libcurl can confuse session tickets arriving from the HTTPS proxy but work as if they arrived from the remote server and then wrongly "short-cut" the host handshake. The reason for this confusion is the modified sequence from TLS 1.2 when the session ids would provided only during the TLS handshake, while in TLS 1.3 it happens post hand-shake and the code was not updated to take that changed behavior into account.

When confusing the tickets, a HTTPS proxy can trick libcurl to use the wrong session ticket resume for the host and thereby circumvent the server TLS certificate check and make a MITM attack to be possible to perform unnoticed.

This flaw can allow a malicious HTTPS proxy to MITM the traffic. Such a malicious HTTPS proxy needs to provide a certificate that curl will accept for the MITMed server for an attack to work - unless curl has been told to ignore the server certificate check.

We are not aware of any exploit of this flaw.

INFO

This flaw has existed in libcurl since commit
549310e907e in libcurl 7.63.0,
released on December 12, 2018.

It can only trigger when TLS 1.3 is used with the HTTPS proxy and not with earlier TLS versions. It *cannot* trigger with TLS 1.2 or earlier versions.

It might be worth highlighting that an HTTPS proxy is a proxy which libcurl communicates with over TLS specifically, and then speaks HTTPS through, making it two layers of TLS. It is different than the more common HTTP proxy setup, where libcurl just does normal TCP with the proxy.

The Common Vulnerabilities and Exposures (CVE) project has assigned the name CVE-2021-XXXX to this issue.

CWE-290: Authentication Bypass by Spoofing

Severity: Low

AFFECTED VERSIONS

This issue only exists when libcurl is built to use OpenSSL or one of its forks.

- Affected versions: curl 7.63.0 to and including 7.75.0
- Not affected versions: curl < 7.63.0 and curl >= 7.76.0

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

THE SOLUTION

Make sure the proxy/host distinction is done correctly.

A fix for CVE-2021-XXXX

(The patch URL will change in the final published version of this advisory)

RECOMMENDATIONS

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

A - Upgrade libcurl to version 7.76.0

B - Apply the patch to your local version

TIMELINE

This issue was reported to the curl project on March 17, 2021.

This advisory was posted on March 31st 2021.

CREDITS

This issue was reported by Mingtao Yang. Patched by Daniel Stenberg.

Thanks a lot!

1 attachment:
F1236029: CVE-2021-XXXX.md

mingtao posted a comment. Mar 19th (2 ye

I can confirm that the updated patch works and addresses the issue.

> @mingtao: also, please let me know if you want to be credited like this or any other way or with your associated organization or what not

Could you credit as "Mingtao Yang, Facebook"?

> Advisory slightly updated.

Thanks!

bagder curl staff posted a comment. Mar 19th (2 ye

> I can confirm that the updated patch works and addresses the issue.

Thanks!

> Could you credit as "Mingtao Yang, Facebook"?

Will do. My local version now has that amended.

bagder curl staff updated CVE reference to CVE-2021-22890. Mar 19th (2 ye

Mar 19th (2 years ago)
bagder curl staff
changed the report title from **TLS 1.3 Session Tickets issued by HTTPS proxies are stored under the non-proxy SSL Context, allowing for a MiTM under a very specific scenario** to **CVE-2021-TLS 1.3 session ticket proxy host mixup**.

url has decided that this report is not eligible for a bounty. Mar 19th (2 ye

The reporter has declined monetary reward. We treat that as a donation to the project and we humbly accept it and will use that money to pay future rewards in th bug-bounty program!

bagder curl staff closed the report and changed the status to **o Resolved**. Mar 31st (2 ye

bagder curl staff requested to disclose this report. Mar 31st (2 ye

The release and the advisory are published