4

CVE-2022-27782: TLS and SSH connection too eager reuse

Share:

TIMELINE

nyymi submitted a report to curl. May 1st (7 months ago)

Summary:

Curl fails to consider some security related options when reusing TLS connections. For example:

- CURLOPT_SSL_OPTIONS
- CURLOPT_PROXY_SSL_OPTIONS
- CURLOPT_CRLFILE
- CURLOPT_PROXY_CRLFILE

As a result for example TLS connection with lower security (`CURLSSLOPT_ALLOW_BEAST` , `CURLSSLOPT_NO_REVOKE`) connection reused when it should no longer be. Also connection that has been authenticated perviously with `CURLSSLOPT_AUTO_CLIENT_CERT` might be reused for connections that should not be.

Steps To Reproduce:

1. `(echo -ne "HTTP/1.1 200 OK\r\nContent-Length: 6\r\n\r\nHello\n"; sleep 5; echo -ne "HTTP/1.1 200 OK\r\nContent-Length: 6\r\n\r\nAgain\n") | openssl s_server -cert cert.pem -key privkey.pem -cert_chain chain.pem -accept 9443`
2. `curl -v --ssl-no-revoke --ssl-allow-beast https://targethost.tld:9443 -: https://targethost.tld:9443`

Connections are made using the same reused connection even though security settings change.

With curl built against openssl:

1. `curl http://cdp.geotrust.com/GeoTrustRSACA2018.crl | openssl crl -out testcrl.pem`
2. `curl -v https://curl.se -: --crlfile crlfile.pem https://curl.se`

With curl built against Schannel and revoked certificate:

1. `curl -v --ssl-no-revoke https://revoked.grc.com -: https://revoked.grc.com`

Second connection will reuse the existing connection even though revocation check is no longer requested.

Note:

There may be more options that might have the similar issues. These were the most obvious I could see (ones having obvious security impact).

Impact

Wrong identity (client certificate) or TLS security options being used for subsequent connections to the same hosts.

nyymi posted a comment. May 1st (7 months ago)

This is quite similar to 1555512 so it might make sense to combine these two into single CVE.

These options and features aren't as obscure as SRP though.

bagder curl staff posted a comment. May 1st (7 months ago)

Thank you for your report!

We will take some time and investigate your reports and get back to you with details and possible follow-up questions as soon as we can!

bagder curl staff posted a comment. May 1st (7 months ago)

> These options and features aren't as obscure as SRP though.

Right, not quite as obscure, but still very rarely used and especially with a different value for the same host name in subsequent transfers. This is indeed a flaw, but I really cannot see how any existing application anywhere is actually having any problems because of this.

nyymi posted a comment. Updated May 1st (7 months ago)

Well, there are more:

If the attacker knows the vulnerable application used ssh key authentication towards specific host with certain username and protocol they can then perform actions to the same host afterwards and abuse the connection reuse. PoC:

`curl --key sshkey --pass asdf1234 sftp://victim@targethost.tld/~/.profile -: sftp://victim@targethost.tld/~/.bash_history`

- I think the client certificate authentication (`CURLOPT_SSLKEY`, `CURLOPT_SSLKEY_BLOB`, `CURLOPT_KEYPASSWD`, `CURLOPT_SSLCERT`, `CURLOPT_SSLCERT_BLOB`) is similarly affected for TLS.

Granted both of these require a specific kind of application to be affected, but clearly such reuse is not desired.

nyymi posted a comment. May 1st (7 months ago)

Scenario 1:

- Application uses `CURLOPT_PROTOCOLS`, `CURLPROTO_HTTP | CURLPROTO_HTTPS | CURLPROTO_FTP | CURLPROTO_FTPS | CURLPROTO_SFTP` for easy handles.
- Application uses `sftp://` to fetch "work url list" from a remote server. The connection is protected by a SSH public key authentication.
- Application processes the "work url list" URLs with libcurl `CURLOPT_URL` and collects the responses.
- Application uses `sftp://` upload to push the "result file" to the remote server. Once uploaded successfully, it uses `CURLOPT_QUOTE` to `rm` the "work url list" and waits for another work list to appear.

If some of the URLs come from the attacker and the attacker can see the response to their URLs, *and* they know which user and hostname the `sftp://` connects with, they can now start exploiting the system by first using `sftp://user@targethost.tld/` to list the server root filesystem, and then examine any files that the sftp user has read access to.

Now, obviously the correct way would be to limit the easy handle to the specific protocols only: `CURLPROTO_SFTP` for the `sftp://` worklist fetch, push and delete, and `CURLPROTO_HTTP | CURLPROTO_HTTPS | CURLPROTO_FTP | CURLPROTO_FTPS` for the URL processing. If this is not done, such application would be exploitable as described.

Scenario 2:

is authenticated with a TLS client certificate.
- Application processes the "work url list" URLs with libcurl `CURLOPT_URL` and collects the responses.
- Application uses `https://` `PUT` to push the "result file" to the remote server. Once uploaded successfully, it uses `https://` `DELETE` to remove the "work url list" and waits for another work list to appear.

If some of the URLs come from the attacker and the attacker can see the response to their URLs, *and* they know which hostname the `https://` connects to they can now issue authenticated `HTTP` `GET` requests directly to the target host system. This could for example expose the full "work url list" and the "result file" (from previous processing cycle), assuming the attacker has knowledge of their paths. If the server has directory indexing enabled, it may expose more sensitive information, too. Any information available for the client certificate authenticated user could be exposed to the attacker.

NOTE: Scenario 1 is confirmed. I haven't yet confirmed scenario 2.

Of course these scenarios are purpose built to be exploitable. There are numerous way to build applications that would not be exploitable, even with these vulnerabilities.

bagder curl staff posted a comment. May 2nd (7 months ago)

> I think the client certificate authentication (CURLOPT_SSLKEY, CURLOPT_SSLKEY_BLOB, CURLOPT_KEYPASSWD, CURLOPT_SSLCERT, CURLOPT_SSLCERT_BLOB) is similarly affected for TLS.

The function `lib/vtls/vtls.c:Curl_ssl_config_matches()` verifies that CURLOPT_SSLCERT and CURLOPT_SSLCERT_BLOB match options. Is that check bad?

nyymi posted a comment. May 2nd (7 months ago)

> Is that check bad?

Nah it seems I just jumped the gun there. So strike Scenario 2.

nyymi posted a comment. May 2nd (7 months ago)

I guess SCP/SFTP support will require adding `Curl_ssh_config_matches()` (or whatever it might be called) to perform similar checks for the SSH options.

make sense? One issue here would of course be that it would likely lead to somewhat reduced performance when the blocking would be too trigger-happy.

bagder curl staff posted a comment. May 2nd (7 months ago)

> would it be safer to reverse the connection re-use decision logic?

It might be, yes. Quite a large refactor but certainly a subject to discuss outside of this particular issue.

bagder curl staff posted a comment. May 2nd (7 months ago)

Here's my list of options I will work on making sure are properly compared. I think it makes sense to treat this as a single issue rather than two simply because they overlap so much.

TLS

CURLOPT_SSL_OPTIONS
CURLOPT_PROXY_SSL_OPTIONS
CURLOPT_CRLFILE
CURLOPT_PROXY_CRLFILE
CURLOPT_TLSAUTH_TYPE
CURLOPT_TLSAUTH_USERNAME
CURLOPT_TLSAUTH_PASSWORD
CURLOPT_PROXY_TLSAUTH_TYPE
CURLOPT_PROXY_TLSAUTH_USERNAME
CURLOPT_PROXY_TLSAUTH_PASSWORD

SSH

CURLOPT_SSH_PUBLIC_KEYFILE
CURLOPT_SSH_PRIVATE_KEYFILE

nyymi posted a comment. May 2nd (7 months ago)

Looks good as far as I can tell.

One issue that libcurl currently has is that if any of the files themselves pointed by `CURLOPT_` change during lifetime of the application the results are not fully consistent or predictable. Attempt to guarantee any kind of consistency in such situations would get awfully complicated, of course.

might point to a different location depending on current working directory.

The use of "current directory" is problematic in general. It is not clear if this will be resolved at the time `setopt` is called or runtime later. If it is resolved at runtime later, it gets tricky to see if connection can be reused or not.

This comment in the relevant code is rather to the point I think:

Code 124 Bytes Wrap lines Copy Download

```
            /* To ponder about: should really the lib be messing about with the
               HOME environment variable etc? */
```

One way to handle this would be to resolve the HOME / currentdir situation at the time of `setopt`. But this wouldn't really resolve the "default" case...

bagder curl staff posted a comment. May 2nd (7 months ago)

> that if any of the files themselves change during lifetime

I think we should clearly document this restriction/functionality and live with it. The same goes for the $HOME treatment.

bagder curl staff posted a comment. May 2nd (7 months ago)

Here's my attempt at fixing the missing checks for all the above mentioned options.

2 attachments:
F1714165: 0002-url-check-SSH-config-match-on-connection-reuse.patch
F1714166: 0001-lib-check-more-TLS-details-for-connection-reuse.patch

bagder curl staff posted a comment. May 2nd (7 months ago)

Advisory draft.

1 attachment:
F1714202: CVE-2022-TLSSSHREUSE.md

bagder curl staff changed the status to ○ **Triaged**. May 2nd (7 months ago)

bagder curl staff updated CVE reference to **CVE-2022-27782**. May 2nd (7 months ago) changed the report title from **TLS connection reuse issues** to **CVE-2022-27782: TLS and SSH connection too eager reuse**.

nyymi posted a comment. May 2nd (7 months ago)

Patch and advisory look good to me.

bagder curl staff posted a comment. May 5th (7 months ago)

I have notified distros@openwall about this issue now. Set for announcement with the pending release on May 11.

bagder curl staff closed the report and changed the status to ○ **Resolved**. May 11th (7 months ago)

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

bagder curl staff requested to disclose this report. May 11th (7 months ago)

nyymi agreed to disclose this report. May 11th (7 months ago)

This report has been disclosed. May 11th (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.