Related:
Bug Bounty
Changelog
Donate
FAQ
Security Problems
Security Process
Vulnerabilities Table

TLS session caching disaster

Project curl Security Advisory, May 26th 2021 - Permalink

VULNERABILITY

libcurl can be tricked into using already freed memory when a new TLS session is negotiated or a client certificate is requested on an existing connection. For example, this can happen when a TLS server requests a client certificate on a connection that was established without one. A malicious server can use this in rare unfortunate circumstances to potentially reach remote code execution in the client.

OpenSSL can declare a "new session" for different reasons, including the initial TLS handshake completion, TLS 1.2 (or earlier) renegotiation, or TLS 1.3 client certificate requests. When libcurl at run-time sets up support for session ID caching on a connection using OpenSSL, it stores pointers to the transfer in-memory object for later retrieval when OpenSSL considers a new session to be established.

However, if the connection is used by multiple transfers (like with a reused HTTP/1.1 connection or multiplexed HTTP/2 connection) that first transfer object might be freed before the new session is established on that connection and then the function will access a memory buffer that might be freed. When using that memory, libcurl might even call a function pointer in the object, making it possible for a remote code execution if the server could somehow manage to get crafted memory content into the correct place in memory.

We are not aware of any exploit of this flaw.

INFO

The flaw can only happen in libcurl built to use OpenSSL (or one of its forks).

This flaw has existed in curl since commit a304051620b92 in libcurl 7.75.0, released on February 3, 2021.

The Common Vulnerabilities and Exposures (CVE) project has assigned the name CVE-2021-22901 to this issue.

CWE-416: Use After Free

Severity: High

Steps to remote code execution

1. libcurl built to use OpenSSL (BoringSSL and libressl work the same)
2. A multi interface using application
3. One of the following:

- create and use a first easy handle to do HTTP/1.1 over TLS to a malicious server
- free that easy handle with `curl_easy_cleanup()`
- create and use a second easy handle to do HTTP/1.1 over TLS with to the same server such that the TLS connection is reused

  or

- more than one concurrent easy handle created that do HTTP/2 over a TLS connection to a malicious server,
- the *first* easy handle to use the connection must be freed with `curl_easy_cleanup()`
- at least one easy handle remaining in use of the same connection

4. The attacking server needs to figure out heap address details in order to know what payload contents to provide
5. The necessary exact memory address in the heap gets populated by memory contents controlled by the server
6. The attacker starts a new handshake (on TLS 1.2 or earlier), or sends a TLS 1.3 client certificate request, or otherwise triggers OpenSSL to consider a new session to be established

For a remote code execution, the client needs to perform (potentially many) more transfers (and thus have more easy handles) to allow the server to place crafted contents into heap memory. Instead of remote code execution, the client could instead crash or otherwise experience undefined behaviour.

AFFECTED VERSIONS

- Affected versions: curl 7.75.0 to and including 7.76.1
- Not affected versions: curl < 7.75.0 and curl >= 7.77.0

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

THE SOLUTION

When the transfer is detached from the connection, it clears the association to it from the session ID cache logic.

A fix for CVE-2021-22901

RECOMMENDATIONS

A - Upgrade curl to version 7.77.0

B - Apply the patch to your local version

C - Build libcurl to use another TLS backend

TIMELINE

This issue was reported to the curl project on April 29, 2021.

This advisory was posted on May 26, 2021.

CREDITS

- Reported-by: Harry Sintonen
- Patched-by: Harry Sintonen, Daniel Stenberg
- Help-by: Brad Spencer

Thanks a lot!