



SECURITY AUDIT

curl

Vulnerabilities identified in curl, libcurl and wcurl
September 2025 – September 2026

Written and compiled by

Joshua Rogers, AISLE Vulnerability Research Team

Contributions by

Elnaz Babayeva, Jiri Balhar, Michal Bida, Michal Buran, Vaclav Cadek, Luigino Camastra, Tomas Dulka, Stanislav Fort, Evelina Gabasova, Michal Janocko, Tomas Klohna, Pavel Kohout, Ondrej Kokes, Adam Krivka, Jakub Kubik, Patrik Mada, Igor Morgenstern, Marek Pavelka, Joshua Rogers, Petr Simecek, Petr Stastny, Jan Tattermusch, Dmitrijs Trizna, Martin Votruba, Guido Vranken, Jakub Zikl

Date

October 2026

Scope

curl, libcurl and wcurl

Findings

35 CVEs · 4 Medium · 31 Low

Document Information

Title	curl Security Audit
Subject	Vulnerabilities identified in curl, libcurl and wcurl
Period covered	September 2025 – September 2026
Written and compiled by	Joshua Rogers, AISLE Vulnerability Research Team
Organisation	AISLE
Date	October 2026

Contributors

AISLE team members who contributed to the research behind this report:

Elnaz Babayeva	Michal Janocko	Joshua Rogers
Jiri Balhar	Tomas Klohna	Petr Simecek
Michal Bida	Pavel Kohout	Petr Stastny
Michal Buran	Ondrej Kokes	Jan Tattermusch
Vaclav Cadek	Adam Krivka	Dmitrijs Trizna
Luigino Camastra	Jakub Kubik	Martin Votruba
Tomas Dulka	Patrik Mada	Guido Vranken
Stanislav Fort	Igor Morgenstern	Jakub Zikl
Evelina Gabasova	Marek Pavelka	

About this report

This report documents the 35 vulnerabilities in curl, libcurl and wcurl that AISLE reported to the curl project between September 2025 and September 2026, and that the curl security team accepted and assigned CVE identifiers. For each finding it records what was wrong, which versions were affected, when it was reported and how it was fixed, with links to curl’s advisory and the fixing commit.

Contents

1	Executive Summary	5
2	Introduction	6
3	Scope and Methodology	6
3.1	Scope	6
3.2	Methodology	7
3.3	Severity ratings	7
3.4	Conventions	7
4	Summary of Findings	8
4.1	Reporting and disclosure over time	9
	Detailed Findings	10
5	Transport Security and Certificate Trust	10
03	MEDIUM Public-key pinning skipped for QUIC connections using GnuTLS	10
04	MEDIUM LDAPS TLS options applied process-wide across concurrent transfers	12
05	LOW Cached OpenSSL CA store ignores the partial-chain setting	13
13	LOW OpenSSL public-key pin not enforced when the server presents no certificate	14
16	LOW wolfSSL CA cache overrides the trust store chosen by the application	15
6	Connection Reuse and Session Isolation	17
08	LOW Connection reuse ignores client private-key settings	17
14	LOW Connection reuse ignores the native CA store setting	18
20	LOW Plain FTP transfer inherits another handle's client-certificate session	19
24	LOW Schannel reuses an auto-authenticated connection after a cross-origin redirect	20
7	Credentials and Authentication	22
07	LOW .netrc password sent for a different user on the same host	22
18	LOW Clearing the AWS SigV4 option does not disable request signing	23
22	LOW Redirected transfer reuses a connection authenticated with NTLM or Negotiate	24
23	LOW Malicious proxy can harvest origin credentials during CONNECT	25
34	LOW Fresh Kerberos ticket minted for a host chosen by a redirect	26
35	LOW .netrc password exposed in the effective URL after a relative redirect	27
8	SSH Host Verification	28
01	LOW wolfSSH backend never verifies the SFTP server's host key	28
10	LOW libssh host-key type mismatch reported as an unknown host	29
19	LOW curl tool skips SSH host verification after a redirect to SFTP or SCP	30
9	Cookie and HSTS State	32
15	LOW A tab before the Secure attribute silently drops the cookie's Secure flag	32
17	LOW Public-suffix host can set a cookie that is sent to all of its siblings	33
21	LOW Unloadable server cookie causes curl to erase its own cookie jar	34
25	LOW HSTS cache import silently drops a longer-lived subdomain policy	35
26	LOW Extreme HSTS max-age value wipes the persisted HSTS cache	36
28	LOW Oversized server cookie breaks later loads of the cookie jar	37
10	Memory Safety	39
06	MEDIUM GSASL context freed twice during SASL mechanism negotiation	39
09	LOW Use-after-free when a transfer is paused from the socket callback	40
11	LOW Use-after-free in the HTTP/2 stream-dependency tree after a handle reset	41
12	LOW Pooled OpenSSL connections outlive their provider library context	42

30	LOW	Failed FTP wildcard listing leaves an internal write callback installed	43
33	LOW	Unbounded recursion when freeing deeply nested MIME structures	44
11		Protocol Handling	46
27	LOW	IMAP APPEND with CRLF conversion lets message data escape as commands	46
29	LOW	Endless control responses defeat timeouts and stall the multi handle	47
31	LOW	TFTP server can turn a download into an upload of local data	48
12		Command-Line Tools	50
02	MEDIUM	wcurl can write outside the working directory via percent-encoded slashes	50
32	LOW	The -J no-clobber guarantee can be bypassed with 103 Early Hints	51
13		Conclusion	53
A		Finding Index	54

1 Executive Summary

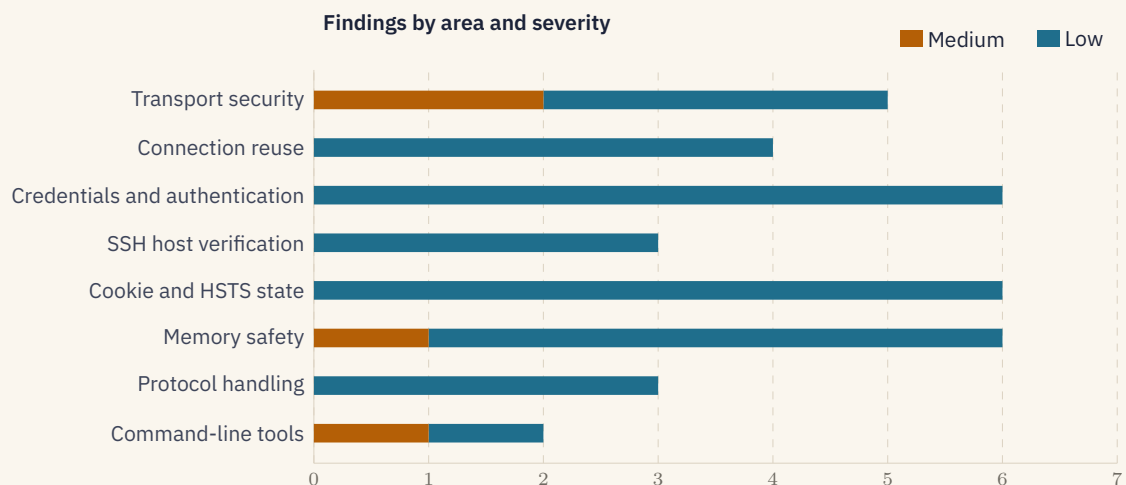
Between September 2025 and September 2026, AISLE's Vulnerability Research Team reported a series of security issues to the curl project. The curl security team accepted **35** of them as vulnerabilities and assigned each a CVE identifier: 34 in curl and libcurl, and one in wcurl, the download wrapper script that ships with curl. All 35 were discovered with AISLE's autonomous source code vulnerability detection engine and validated by AISLE researchers before being reported.



Every finding has been fixed and published in a curl security advisory, alongside curl releases 8.17.0 through 8.23.0. curl rated four of the findings **Medium** and the remaining 31 **Low**; none was rated High or Critical.

Key figures

- AISLE findings account for six of the 18 CVEs fixed in curl 8.21.0, six of the nine curl and libcurl CVEs in curl 8.22.0, and 18 of the CVEs published with curl 8.23.0.
- 14 of the 35 flaws had been in curl for at least five years when they were reported, and eight for more than a decade. The oldest dates back to curl 7.7, released in 2001.
- Seven were reported less than a year after the code that introduced them was committed.
- The findings span eight areas of the codebase, from TLS trust handling and connection reuse to cookie storage, memory safety and the command-line tools.



2 Introduction

“command line tool and library for transferring data with URLs”

curl.se

curl is among the most widely deployed pieces of software in the world. The `curl` command-line tool is present on practically every desktop and server operating system, and `libcurl` is embedded in operating systems, programming-language runtimes, vehicles, televisions, routers and countless other devices. It is also one of the most carefully maintained open source projects: it has been audited repeatedly, fuzzed continuously, and runs a public vulnerability disclosure programme with a demanding bar for what it accepts as a security issue.

This report collects every vulnerability that AISLE has reported to the curl project and that curl has accepted and assigned a CVE identifier. It is written as a single reference: it records what each flaw was, which versions it affected, when it was reported and how it was fixed, and it links each finding to the original report and to curl’s advisory.

The findings were discovered by AISLE’s autonomous source code vulnerability detection engine. The engine analysed curl’s source code and flagged candidate vulnerabilities. Researchers on AISLE’s Vulnerability Research Team then validated each candidate against current upstream builds, confirmed its security impact and reported it to the curl security team under coordinated disclosure. In many cases the report included a suggested patch, and several fixes were written by AISLE.

Of the 35 accepted CVEs, 31 were rated Low and four Medium on curl’s four-level scale. That spread is consistent with a mature codebase in which the remaining flaws tend to need specific build options, rarely used features or unusual application behaviour to be reached. It is also a reminder that such flaws persist: fourteen of the issues in this report had survived for at least five years of continuous review, testing and fuzzing.

3 Scope and Methodology

3.1 Scope

Targets	curl (command-line tool), libcurl, and wcurl
Source	https://github.com/curl/curl and https://github.com/curl/wcurl
Period	Reports submitted between 23 September 2025 and 18 September 2026
Versions examined	Development snapshots of curl from 8.17.0 through the 8.23.0 development cycle, and the matching wcurl releases
Configurations	Findings cover the default build and optional backends and features, including OpenSSL, GnuTLS, wolfSSL and Schannel for TLS, libssh and wolfSSH for SSH, and GNU SASL, libpsl and GSS-API
Included	Every report that curl accepted as a vulnerability and assigned a CVE
Excluded	Reports that curl classified as ordinary bugs or hardening, and reports still under assessment

3.2 Methodology

1. **Discovery.** AISLE's autonomous source code vulnerability detection engine analysed the curl and wcurl source trees and produced candidate findings.
2. **Validation.** An AISLE researcher reproduced each candidate against an upstream build, confirmed the root cause and the affected configurations, and assessed its practical impact.
3. **Reporting.** Validated findings were sent to the curl security team, in most cases through curl's HackerOne programme, usually with a suggested fix.
4. **Coordinated disclosure.** The curl security team triaged each report, assigned a severity and CVE, prepared the fix, and published an advisory together with the release containing it.

3.3 Severity ratings

Severities in this report are the ratings assigned by the curl security team. curl uses four levels, Low, Medium, High and Critical, and deliberately does not publish CVSS scores. When choosing a level, curl weighs how the flaw is reached (attack vector and complexity, required privileges, build options, protocols and platforms) against what a successful attack achieves.

LOW	Hard or unlikely to exploit, for example because of timing, platform requirements or rarely used options and protocols.
MEDIUM	Easier to exploit than Low, with fewer constraints or more common options, but usually needing something else to go wrong before it becomes serious.
HIGH	A serious problem with real-world impact that readily compromises confidentiality, integrity or availability.
CRITICAL	Easily exploited by a remote, unauthenticated attacker in a common configuration, leading to system compromise.

Third parties, including CISA, publish their own CVSS scores for curl CVEs. Those scores are often considerably higher than curl's ratings and are not used here.

3.4 Conventions

- Findings are numbered [AISLE-CURL-01](#) to [AISLE-CURL-35](#) in order of CVE identifier, and grouped by area.
- All dates are in UTC. *Reported* is the date the report reached the curl security team.
- *Affected* gives the first and last affected releases. For three findings (AISLE-CURL-18, -20 and -27), the fix shipped in curl 8.22.0 and the advisory followed with curl 8.23.0.
- *Disclosed* gives the curl release with which the advisory was published.
- Each CVE identifier links to curl's advisory, and each fix links to the commit that fixed it.

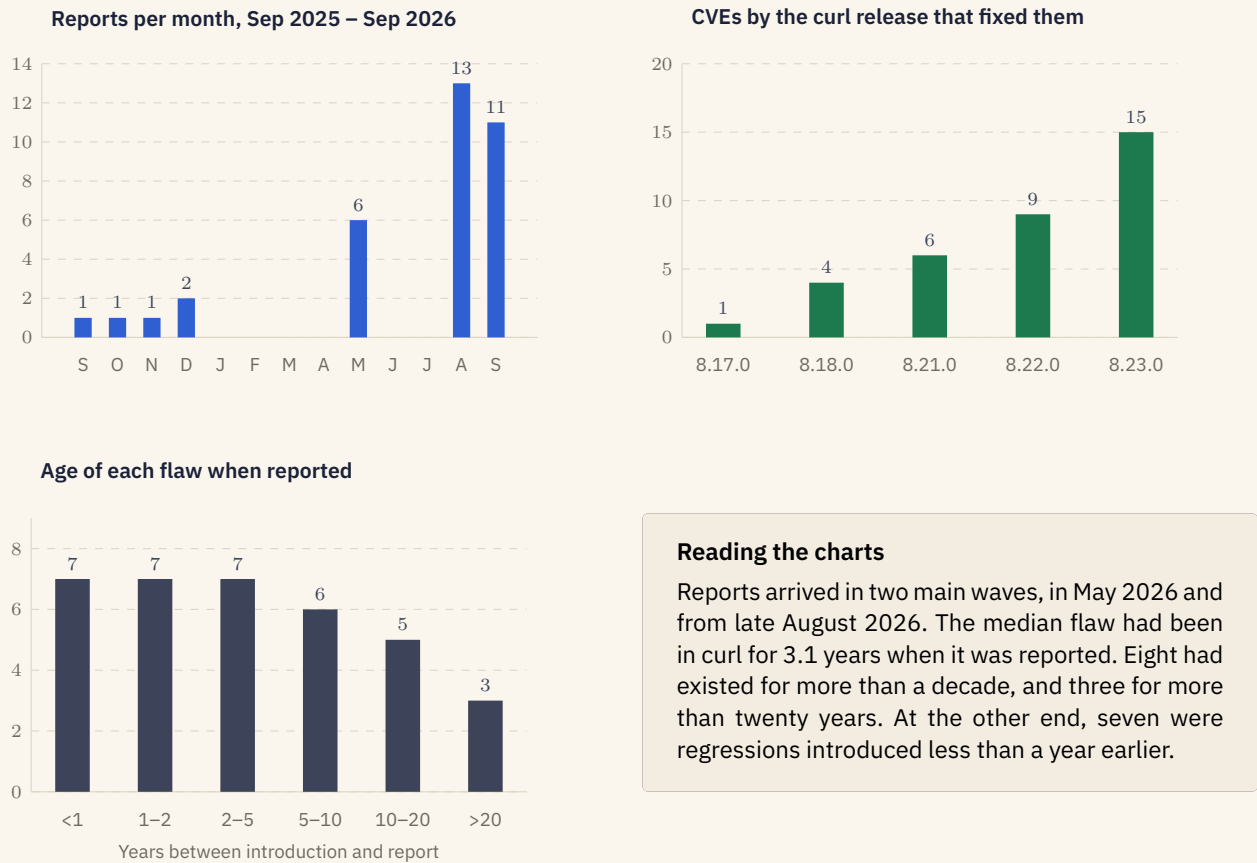
4 Summary of Findings

The table below lists all 35 findings grouped by area. Each identifier links to the detailed finding.

ID	Finding	CVE	Severity	Fixed in
TRANSPORT SECURITY AND CERTIFICATE TRUST				
03	QUIC public-key pinning skipped with GnuTLS	CVE-2025-13034	MEDIUM	curl 8.18.0
04	Threaded LDAPS TLS options applied globally	CVE-2025-14017	MEDIUM	curl 8.18.0
05	OpenSSL CA cache ignores partial-chain setting	CVE-2025-14819	LOW	curl 8.18.0
13	OpenSSL public-key pinning bypass	CVE-2026-80230	LOW	curl 8.22.0
16	wolfSSL CA cache overrides callback trust store	CVE-2026-82208	LOW	curl 8.22.0
CONNECTION REUSE AND SESSION ISOLATION				
08	Incomplete mTLS matching in connection reuse	CVE-2026-8932	LOW	curl 8.21.0
14	Native CA store ignored in connection reuse	CVE-2026-80231	LOW	curl 8.22.0
20	FTP reuse inherits client-certificate identity	CVE-2026-86411	LOW	curl 8.22.0
24	Automatic client certificate reused after redirect	CVE-2026-86415	LOW	curl 8.23.0
CREDENTIALS AND AUTHENTICATION				
07	.netrc password used for the wrong user	CVE-2026-8926	LOW	curl 8.21.0
18	Stale SigV4 signing after option reset	CVE-2026-86409	LOW	curl 8.22.0
22	NTLM/Negotiate connection reused after redirect	CVE-2026-86413	LOW	curl 8.23.0
23	Origin credentials leaked to an HTTP proxy	CVE-2026-86414	LOW	curl 8.23.0
34	Negotiate ticket sent to a foreign origin	CVE-2026-94265	LOW	curl 8.23.0
35	.netrc password in CURLINFO_EFFECTIVE_URL	CVE-2026-94266	LOW	curl 8.23.0
SSH HOST VERIFICATION				
01	No SFTP host verification with wolfSSH	CVE-2025-10966	LOW	curl 8.17.0
10	libssh key-type mismatch treated as new host	CVE-2026-9547	LOW	curl 8.21.0
19	Redirect to SFTP/SCP skips host verification	CVE-2026-86410	LOW	curl 8.23.0
COOKIE AND HSTS STATE				
15	Secure cookie flag dropped after a tab	CVE-2026-80255	LOW	curl 8.22.0
17	Public-suffix cookie not made host-only	CVE-2026-82209	LOW	curl 8.22.0
21	Server cookie erases the cookie jar	CVE-2026-86412	LOW	curl 8.23.0
25	HSTS import drops longer-lived child entry	CVE-2026-87862	LOW	curl 8.23.0
26	HSTS response erases persisted entries	CVE-2026-87863	LOW	curl 8.23.0
28	Server cookie aborts cookie jar load	CVE-2026-92386	LOW	curl 8.23.0
MEMORY SAFETY				
06	GSASL double free	CVE-2026-8925	MEDIUM	curl 8.21.0
09	Use-after-free on pause in socket callback	CVE-2026-9080	LOW	curl 8.21.0
11	HTTP/2 stream-dependency tree use-after-free	CVE-2026-10536	LOW	curl 8.21.0
12	OpenSSL provider use-after-free	CVE-2026-80229	LOW	curl 8.22.0

ID	Finding	CVE	Severity	Fixed in
30	FTP wildcard write-callback mix-up	CVE-2026-92388	LOW	curl 8.23.0
33	Stack overflow in curl_mime_free	CVE-2026-92391	LOW	curl 8.23.0
PROTOCOL HANDLING				
27	IMAP APPEND command injection with CRLF conversion	CVE-2026-87864	LOW	curl 8.22.0
29	Never-ending control-response loop	CVE-2026-92387	LOW	curl 8.23.0
31	TFTP transfer direction confusion	CVE-2026-92389	LOW	curl 8.23.0
COMMAND-LINE TOOLS				
02	wcurl path traversal via percent-encoding	CVE-2025-11563	MEDIUM	wcurl 2025.11.09
32	-J no-clobber bypass via 103 Early Hints	CVE-2026-92390	LOW	curl 8.23.0

4.1
 Reporting and disclosure over time



Detailed Findings

The following sections describe each finding. They are grouped by the area of curl they affect and, within each area, ordered by identifier. Every finding starts with a summary block giving its CVE, status, affected versions, fix and report link.

5 Transport Security and Certificate Trust

Findings that weaken certificate verification, public-key pinning or trust-store handling in one of curl's TLS backends.

AISLE-CURL-03**MEDIUM**

Public-key pinning skipped for QUIC connections using GnuTLS

CVE	CVE-2025-13034	Disclosed	curl 8.18.0, 7 Jan 2026
Component	HTTP/3 · ngtcp2 with GnuTLS	Scope	libcurl and curl tool
Affected	8.8.0 – 8.17.0	Fixed in	curl 8.18.0 (3d91ca8cdb)
Weakness	CWE-295 Improper Certificate Validation	Introduced	3210101088 , Mar 2024
Reported	9 Nov 2025	Report	Not filed on HackerOne

DESCRIPTION

CURLOPT_PINNEDPUBLICKEY (--pinnedpubkey in the curl tool) tells curl to accept a server only if its certificate carries a specific public key. The pin is meant to hold whatever the other verification settings are. For HTTP/3 connections made through ngtcp2 with GnuTLS, libcurl performs all of its peer checks in one call to `Curl_gtls_verifyserver()`: chain verification, hostname matching and the pin comparison. The QUIC glue code in `Curl_vquic_tls_verify_peer()` made that call only when hostname verification was enabled.

```
lib/vquic/vquic-tls.c · fix 3d91ca8cdb
```

```
#elif defined(USE_GNUTLS)
- if(conn_config->verifyhost) {
-     result = Curl_gtls_verifyserver(cf, data, ctx->gtls.session,
-                                     conn_config, &data->set.ssl, peer,
-                                     data->set.str[STRING_SSL_PINNEDPUBLICKEY]);
-     if(result)
-         return result;
- }
+ result = Curl_gtls_verifyserver(cf, data, ctx->gtls.session,
+                                 conn_config, &data->set.ssl, peer,
+                                 data->set.str[STRING_SSL_PINNEDPUBLICKEY]);
+ if(result)
+     return result;
```

With `CURLOPT_SSL_VERIFYHOST` switched off, the function was never reached and the pin was never compared, so the connection went ahead as though it had matched. GnuTLS over TCP and the other QUIC TLS backends were not gated this way. The flaw is the GnuTLS counterpart of CVE-2025-5025, an earlier pinning bypass in the wolfSSL QUIC path.

IMPACT

Only builds that use `ngtcp2` with GnuTLS for HTTP/3 are affected, and only transfers that request pinning while disabling standard certificate verification. That combination is less odd than it sounds. An application that pins a known key may see little point in a CA chain or hostname check and turn them off, leaving the pin as its only authentication of the server. With the pin ignored, an attacker able to intercept the QUIC connection could present any certificate and impersonate the server. HTTP/1 and HTTP/2 connections from the same build enforced the pin correctly. curl rated the issue Medium: the outcome is full server impersonation, but it is confined to one QUIC and TLS pairing and to a configuration the user must choose.

RECOMMENDATION

A check the caller explicitly requests, such as a key pin, should run on every connection and should never sit behind a condition that belongs to a different option. Each verification step should be governed by its own setting alone. Users who cannot upgrade can keep hostname verification enabled, build curl with a different TLS library for QUIC, or avoid HTTP/3 for transfers that depend on pinning.

RESOLUTION

curl removed the `verifyhost` condition, so `Curl_gtls_verifyserver()` is now called for every GnuTLS QUIC connection; the function already applies each of its checks according to the corresponding option. The fix shipped in curl 8.18.0. curl's advisory also lists the Rock-solid curl long-term support releases 8.16.1 and 8.14.2 as not affected.

AISLE-CURL-04

MEDIUM

LDAPS TLS options applied process-wide across concurrent transfers

CVE	CVE-2025-14017	Disclosed	curl 8.18.0, 7 Jan 2026
Component	LDAP · legacy LDAP backend	Scope	libcurl
Affected	7.17.0 – 8.17.0	Fixed in	curl 8.18.0 (39d1976b7f)
Weakness	CWE-567 Unsynchronized Access to Shared Data in a Multi-threaded Context	Introduced	ccba0d10b6 , Aug 2007
Reported	1 Dec 2025	Report	Not filed on HackerOne

DESCRIPTION

The flaw is in libcurl's legacy LDAP backend, `lib/ldap.c`, as used on platforms other than Windows. When setting up an LDAPS transfer, this code configured the TLS options (the CA certificate file and the certificate-verification mode) before it had created the LDAP session with `ldap_init()`. Having no session handle yet, it passed `NULL` to `ldap_set_option()`. In the LDAP C API a `NULL` handle does not mean “this connection”. It changes the library's global defaults, which every session created afterwards inherits.

lib/ldap.c · before the fix

```
rc = ldap_set_option(NULL, LDAP_OPT_X_TLS_CACERTFILE, ldap_ca);
...
rc = ldap_set_option(NULL, LDAP_OPT_X_TLS_REQUIRE_CERT, &ldap_option);
...
server = ldap_init(host, ipquad.remote_port);
```

Each LDAPS transfer therefore wrote its TLS settings into state shared by the whole process, then created a session that picked them up. If two threads set up LDAPS transfers at the same moment, one could create its session just after the other had overwritten the defaults, and connect with the other transfer's CA file and verification mode. The protocol version and network timeout were set the same way.

IMPACT

Only multi-threaded applications that run LDAPS transfers concurrently through the legacy backend are exposed. Builds using curl's OpenLDAP backend are not affected, nor is WinLDAP, whose API lacks these options, nor the curl tool. The dangerous case is a thread that disables certificate verification for one server: another thread's LDAPS connection set up at the same time could inherit that setting and accept a forged certificate, giving an on-path attacker access to the directory traffic and any bind credentials. curl rated the issue Medium. The consequence is a loss of certificate verification, but it needs concurrent transfers with differing TLS settings, and the shared defaults are read only while a session is being created, so the timing window is narrow.

RECOMMENDATION

Options meant for a single connection should be set on that connection's own handle, which means creating the handle before configuring it. Per-transfer code in a thread-safe library should never write process-wide defaults. Users who cannot upgrade can build libcurl with the OpenLDAP backend, give all concurrent LDAPS transfers the same TLS settings, or avoid LDAP altogether.

RESOLUTION

curl moved `ldap_init()`, and its Windows counterpart `ldap_sslinit()`, ahead of all option handling, so every `ldap_set_option()` call now targets the new session handle. The same change removed a block of long-disabled STARTTLS code. The fix shipped in curl 8.18.0, and curl's advisory also lists the Rock-solid curl long-term support releases 8.16.1 and 8.14.2 as not affected.

AISLE-CURL-05

LOW

Cached OpenSSL CA store ignores the partial-chain setting

CVE	CVE-2025-14819	Disclosed	curl 8.18.0, 7 Jan 2026
Component	TLS · OpenSSL CA cache	Scope	libcurl
Affected	7.87.0 – 8.17.0	Fixed in	curl 8.18.0 (cd046f6c93)
Weakness	CWE-295 Improper Certificate Validation	Introduced	3c16697ebd , Sep 2022
Reported	16 Dec 2025	Report	Not filed on HackerOne

DESCRIPTION

libcurl's OpenSSL backend keeps the parsed CA store (an `X509_STORE`) cached on the multi handle, so that transfers using the same CA bundle do not load and parse it again. A cached store lives for up to 24 hours by default, as set by `CURLOPT_CA_CACHE_TIMEOUT`. The store holds more than certificates. Unless the application sets `CURLSSLOPT_NO_PARTIALCHAIN` in `CURLOPT_SSL_OPTIONS`, libcurl sets OpenSSL's `X509_V_FLAG_PARTIAL_CHAIN` on the store, which lets any certificate in the CA bundle, intermediates included, serve as a trust anchor.

Before reusing a cached store, libcurl checked only that the CA file path matched. It did not compare the partial-chain policy under which the store had been built. A transfer that changed `CURLSSLOPT_NO_PARTIALCHAIN` on a reused easy or multi handle could therefore receive a store carrying the opposite policy. The fix records the setting with the cached store and treats a mismatch as a different store:

lib/vtls/openssl.c · fix [cd046f6c93](#)

```
struct ssl_primary_config *conn_config = Curl_ssl_cf_get_primary_config(cf);
+ struct ssl_config_data *ssl_config =
+   Curl_ssl_cf_get_config(cf, CURL_UNCONST(data));
+ if(mb->no_partialchain != ssl_config->no_partialchain)
+   return TRUE;
if(!mb->CAfile || !conn_config->CAfile)
  return mb->CAfile != conn_config->CAfile;
```

IMPACT

Applications using libcurl with OpenSSL, or one of its forks, are affected if they change `CURLSSLOPT_NO_PARTIALCHAIN` between transfers on a reused handle and leave CA caching enabled, as it is by default. The case that matters is a transfer that asked for complete chains but was given a store that allows partial ones: it could accept a server whose chain ends at an intermediate certificate in the CA bundle rather than at a root. The reverse mismatch only causes spurious failures. curl rated this Low. Certificate verification still ran against the application's own CA bundle, only the choice of trust anchors within it was wrong, and applications rarely toggle this option between transfers. The curl tool is not affected.

RECOMMENDATION

A cache lookup for security-relevant state should compare every setting that influenced how the cached object was built, not only the input it was built from. Users who cannot upgrade can disable CA caching by setting `CURLOPT_CA_CACHE_TIMEOUT` to zero, or apply the same `CURLSSLOPT_NO_PARTIALCHAIN` setting to every transfer that shares a handle.

RESOLUTION

curl now stores the partial-chain setting alongside each cached CA store and rebuilds the store when a transfer asks for a different policy. The fix shipped in curl 8.18.0, and curl's advisory also lists the Rock-solid curl long-term support releases 8.16.1 and 8.14.2 as not affected.

AISLE-CURL-13

LOW

OpenSSL public-key pin not enforced when the server presents no certificate

CVE	CVE-2026-80230	Disclosed	curl 8.22.0, 2 Sep 2026
Component	TLS · OpenSSL and forks	Scope	libcurl and curl tool
Affected	7.45.0 – 8.21.0	Fixed in	curl 8.22.0 (5267ed859d)
Weakness	CWE-295 Improper Certificate Validation	Introduced	8363656cb4 , Aug 2015
Reported	24 Aug 2026	Report	HackerOne #3969300

DESCRIPTION

When `CURLOPT_PINNEDPUBLICKEY` is set, libcurl's OpenSSL backend compares the pin with the public key in the server's certificate. This happens after the handshake, in `Curl_ossll_check_peer_cert()`, and needs a certificate to work with. Some TLS configurations let a handshake complete without any server certificate, for example anonymous cipher suites, which OpenSSL does not enable by default but an application can allow through its cipher list.

A 2015 change made this function tolerate a missing certificate when both `CURLOPT_SSL_VERIFYPEER` and `CURLOPT_SSL_VERIFYHOST` are disabled, since there is then nothing to verify. Its early exit jumps to the end of the function, past the pin check further down. A pinned transfer with standard verification off therefore accepted a server that presented no certificate, when it should have failed. Proxy pinning through `CURLOPT_PROXY_PINNEDPUBLICKEY` was bypassed the same way. The fix adds the missing condition:

lib/vtls/openssl.c · fix 5267ed859d

```
server_cert = SSL_get1_peer_certificate(octx->ssl);
if(!server_cert) {
    /* no verification at all, this maybe acceptable */
-   if(!(conn_config->verifypeer || conn_config->verifyhost))
+   if(!(conn_config->verifypeer || conn_config->verifyhost) &&
+       !pinned(cf, data))
    goto out;
```

IMPACT

libcurl and the curl tool are affected when built with OpenSSL or a fork such as BoringSSL, AWS-LC, LibreSSL or QuicTLS. The exposed users are those who rely on a pin as their only authentication of the server, with peer and hostname verification both disabled. An attacker who can intercept the connection and complete a handshake without a certificate has the connection accepted as though the pin had matched. curl rated this Low because every precondition is a deliberate weakening: verification must be off, and the client's TLS configuration most likely has to permit certificate-less handshakes, which OpenSSL does not allow by default.

RECOMMENDATION

A missing server certificate should be acceptable only when nothing the caller asked for depends on one. A configured pin depends on a certificate by definition, so its presence should turn a missing certificate into a hard failure. Users who cannot upgrade should keep standard verification enabled alongside pinning, and should not enable anonymous cipher suites.

RESOLUTION

curl factored the pin lookup into a small helper, `pinned()`, and consults it on the no-certificate path, so a pinned connection without a server certificate now fails verification. The fix shipped in curl 8.22.0 and is also included in the Rock-solid curl long-term support releases 8.20.1, 8.16.1 and 8.14.2.

AISLE-CURL-16

LOW

wolfSSL CA cache overrides the trust store chosen by the application

CVE	CVE-2026-82208	Disclosed	curl 8.22.0, 2 Sep 2026
Component	TLS · wolfSSL CA cache	Scope	libcurl
Affected	8.9.1 – 8.21.0	Fixed in	curl 8.22.0 (ed0338befd)
Weakness	CWE-295 Improper Certificate Validation	Introduced	0f2876b2c3 , Jul 2024
Reported	27 Aug 2026	Report	HackerOne #3973090

DESCRIPTION

libcurl's wolfSSL backend can share one X.509 trust store between connections through a CA cache held on the multi handle. The cache applies only when peer verification is on, caching has not been disabled with `CURLOPT_CA_CACHE_TIMEOUT`, and trust comes solely from a CA file or wolfSSL's defaults. `Curl_wssl_setup_x509_store()` installs either the cached store or a newly populated one in the wolfSSL context. A flag, `x509_store_setup`, is meant to stop it running twice for a connection.

If the application has set `CURLOPT_SSL_CTX_FUNCTION`, libcurl sets up the store first and then runs the callback, which may replace the store with one of its own. The handshake code calls the setup function again unless the flag says it has already run.

lib/vtls/wolfssl.c · before the fix

```
if(data->set.ssl.fsslctx) {
    if(!wctx->x509_store_setup) {
        result = Curl_wssl_setup_x509_store(cf, data, wctx);
        ...
    }
    result = (*data->set.ssl.fsslctx)(data, wctx->ssl_ctx,
                                    data->set.ssl.fsslctxp);
    ...
}
if(!wssl->x509_store_setup) {
    result = Curl_wssl_setup_x509_store(cf, data, wssl);
}
```

The flag was set only in `wssl_populate_x509_store()`, which loads certificates into a fresh store and is skipped on a cache hit. After a hit the flag stayed false, the second call found the cached store, and it installed that store over the callback's. The server's certificate was then checked against the cached trust anchors instead of those the application had chosen.

IMPACT

Only libcurl built with wolfSSL is affected, and only applications whose `CURLOPT_SSL_CTX_FUNCTION` callback replaces the trust store in a configuration that is otherwise cacheable. The curl tool is not affected. The first connection populates the cache and behaves correctly; later connections on the same multi handle revert to the cached store. An attacker holding a certificate that chains to a CA in the cached store, but not in the store the callback installs, could then impersonate the server. curl rated this Low: it needs one TLS backend, a trust-replacing callback and a suitable certificate, and verification still runs against a genuine trust store, just not the intended one.

RECOMMENDATION

A run-once guard should be set on entry to the routine it protects, not in one of its branches, so that every path through the routine counts as having run. Users who cannot upgrade can set `CURLOPT_CA_CACHE_TIMEOUT` to zero so the cache is never used, or avoid `CURLOPT_SSL_CTX_FUNCTION` with the wolfSSL backend.

RESOLUTION

curl moved the assignment of `x509_store_setup` from `wssl_populate_x509_store()` to the start of `Curl_wssl_setup_x509_store()`, so the flag is set on the cache-hit path as well and a callback's trust store is no longer overwritten. The fix shipped in curl 8.22.0 and is also included in the Rock-solid curl long-term support releases 8.20.1, 8.16.1 and 8.14.2.

6 Connection Reuse and Session Isolation

Findings in which curl reused a pooled connection, or the identity attached to one, for a transfer whose configuration should have required a new connection.

AISLE-CURL-08

LOW

Connection reuse ignores client private-key settings

CVE	CVE-2026-8932	Disclosed	curl 8.21.0, 24 Jun 2026
Component	TLS · connection pool and session cache	Scope	libcurl
Affected	7.7 – 8.20.0	Fixed in	curl 8.21.0 (7541ae569d)
Weakness	CWE-305 Authentication Bypass by Primary Weakness	Introduced	a1d6ad2610 , Feb 2001
Reported	13 May 2026	Report	HackerOne #3733910

DESCRIPTION

Before libcurl reuses a pooled connection or resumes a cached TLS session, it checks that the new transfer asks for the same TLS setup as the one that created it. Both checks read the `ssl_primary_config` structure. `match_ssl_primary_config()` compares its fields one by one, and the session cache derives its peer key from it. The client certificate path was part of that structure, but the settings describing the private key were not. `CURLOPT_SSLKEY`, `CURLOPT_SSLKEYTYPE`, `CURLOPT_KEYPASSWD`, `CURLOPT_SSLKEYBLOB` and `CURLOPT_SSLCERTTYPE` were stored in the wider `ssl_config_data` structure, which neither check consults.

As a result, two easy handles that named the same certificate but a different key, key format or key password looked identical to libcurl. The second handle could take over the first one’s authenticated connection, or resume its TLS session, without ever loading a working key of its own. The flaw is similar to CVE-2022-27782, where a different set of TLS and SSH options was missing from the same matching logic.

IMPACT

Only applications that share a connection pool or TLS session cache between easy handles carrying different client credentials for the same server are affected. That pattern appears in multi-tenant services that make mutually authenticated requests on behalf of several users; the curl tool does not use it. There, a handle with the wrong key or no key password could be authenticated to the server as another handle’s identity. curl rated the issue Low because it needs this specific application design, and the handles involved are all configured by the same program.

RECOMMENDATION

Every option that shapes the identity a connection presents should take part in reuse matching and in the session cache key, wherever it happens to be stored. Applications that cannot upgrade should not share connections between handles with different client-certificate settings. Where that is unavoidable, disabling both connection reuse (`CURLOPT_FORBID_REUSE`) and session resumption (`CURLOPT_SSL_SESSIONID_CACHE`) on those handles closes the gap.

RESOLUTION

AISLE supplied the fix. It moves the five fields into `ssl_primary_config` and adds them to the reuse predicate, comparing the key password in constant time:

```
lib/vtls/vtls.c · fix 7541ae569d

    Curl_safecmp(c1->CRLfile, c2->CRLfile) &&
-   Curl_safecmp(c1->pinned_key, c2->pinned_key))
+   Curl_safecmp(c1->pinned_key, c2->pinned_key) &&
+   curl_strequal(c1->cert_type, c2->cert_type) &&
+   Curl_safecmp(c1->key, c2->key) &&
+   curl_strequal(c1->key_type, c2->key_type) &&
+   !Curl_timestrcmp(c1->key_passwd, c2->key_passwd))
    return TRUE;
```

The session cache key now records the actual certificate and key paths in place of a fixed `:CCERT` marker, and the password is kept out of the key. Two new tests cover the change. The fix shipped in curl 8.21.0 and is included in the Rock-solid curl long-term support releases 8.20.1, 8.16.1 and 8.14.2.

AISLE-CURL-14

LOW

Connection reuse ignores the native CA store setting

CVE	CVE-2026-80231	Disclosed	curl 8.22.0, 2 Sep 2026
Component	TLS · connection pool (Windows, macOS)	Scope	libcurl and curl tool
Affected	7.71.0 – 8.21.0	Fixed in	curl 8.22.0 (7be1e70cb6)
Weakness	CWE-488 Exposure of Data Element to Wrong Session	Introduced	148534db57 , Sep 2019
Reported	24 Aug 2026	Report	HackerOne #3969368

DESCRIPTION

`CURLSSLOPT_NATIVE_CA` (`--ca-native` in the curl tool) tells libcurl to trust the operating system's certificate store when verifying a server, in place of or alongside a CA bundle. Builds that use Apple's SecTrust, or that are configured for a native CA store, also switch it on by default when no custom CA file, path or blob is set.

libcurl kept the resulting `native_ca_store` flag in `ssl_config_data`, the per-transfer structure, and not in `ssl_primary_config`, which is copied onto each connection and compared by `match_ssl_primary_config()` when a pooled connection is considered for reuse. Two transfers to the same host with otherwise identical TLS settings could therefore share a connection even though one wanted the native store and the other did not. The connection had been verified under whichever setting created it, and the second transfer inherited that decision. It is the same class of gap as AISLE-CURL-08, applied to server verification.

IMPACT

libcurl and the curl tool on Windows and macOS are affected when transfers to one host differ in their native-store setting and share a connection pool. The operating system's store usually trusts far more roots than a curated bundle, including locally installed enterprise or interception CAs. A transfer that deliberately left the native store off could therefore run over a connection whose certificate would have failed its own verification, with no error reported. Both transfers still verify the server; they only differ in which roots they accept. That, together with the platform restriction and the need to mix settings for one host, explains the Low rating.

RECOMMENDATION

Every setting that decides which certificates are trusted should be part of the connection-reuse key, so that a connection is only shared between transfers with the same trust configuration. Users who cannot upgrade can set `CURLOPT_FORBID_REUSE` on transfers that use the native CA store, as curl's advisory suggests, or use one native-store setting consistently for any given host.

RESOLUTION

curl moved `native_ca_store` into `ssl_primary_config`, cloned it onto each connection and added it to `match_ssl_primary_config()`. The GnuTLS, OpenSSL, wolfSSL and rustls backends now read it from the connection's configuration. The fix shipped in curl 8.22.0 and is included in the Rock-solid curl long-term support releases 8.20.1, 8.16.1 and 8.14.2.

AISLE-CURL-20

LOW

Plain FTP transfer inherits another handle's client-certificate session

CVE	CVE-2026-86411	Disclosed	curl 8.23.0, 14 Oct 2026
Component	FTP · connection pool	Scope	libcurl and curl tool
Affected	8.21.0	Fixed in	curl 8.22.0 (72352daba0)
Weakness	CWE-488 Exposure of Data Element to Wrong Session	Introduced	4ff212f8ed , May 2026
Reported	27 Aug 2026	Report	HackerOne #3973190

DESCRIPTION

Some protocols, FTP among them, carry the `PROTOPT_SSL_REUSE` flag. It lets libcurl serve a plain `ftp://` request over a pooled connection to the same server that is already protected by TLS, because an encrypted connection is at least as good as a clear-text one. `url_match_ssl_use()` allowed this upgrade after checking only that the new request's scheme had the flag and that the pooled connection belonged to the same protocol family.

The TLS configurations were never compared. `url_match_ssl_config()` checks them only when the new request itself asks for TLS, either through a TLS scheme or by requesting `STARTTLS`, and a plain FTP request does neither. A transfer with no TLS settings at all could therefore be given a connection that another easy handle had set up with its own client certificate, and the server would treat it as coming from that certificate's owner. The fix adds the missing check:

```
lib/url.c · fix 72352daba0

    if(!(m->needle->scheme->flags & PROTOPT_SSL_REUSE) ||
        (get_protocol_family(conn->scheme) != m->needle->scheme->protocol))
        return FALSE;
+   /* We may reuse this as an auto-TLS upgrade, but only if the SSL
+    * config parameters match. */
+   if(!Curl_ssl_conn_config_match(m->data, conn, FALSE))
+       return FALSE;
```

IMPACT

The flaw affects libcurl and the curl tool when one connection pool serves both FTPS transfers made with a client certificate and plain FTP transfers to the same server. Requests the application meant to send without a certificate are instead attributed to the certificate holder, with that identity's access and session state on the server. Any other TLS setting that differed between the handles was skipped in the same way. curl rated the issue Low because it needs this mix of authenticated and unauthenticated FTP use in one pool, all under the application's control, and because the code was only present in curl 8.21.0.

RECOMMENDATION

Reuse across the TLS boundary should apply the same TLS configuration check as same-scheme TLS reuse, since the transfer ends up running over TLS either way. Users of curl 8.21.0 who cannot upgrade should keep FTP transfers with and without client certificates in separate connection pools, or set `CURLOPT_FORBID_REUSE` on the plain FTP handles.

RESOLUTION

curl now calls `Curl_ssl_conn_config_match()` before it lets a plain request reuse a TLS connection, as shown above, so the upgrade path requires matching TLS configurations. The four-line fix shipped in curl 8.22.0. The advisory was held back and published with curl 8.23.0.

AISLE-CURL-24

LOW

Schannel reuses an auto-authenticated connection after a cross-origin redirect

CVE	CVE-2026-86415	Disclosed	curl 8.23.0, 14 Oct 2026
Component	TLS · Schannel (Windows)	Scope	libcurl and curl tool
Affected	8.21.0 – 8.22.0	Fixed in	curl 8.23.0 (00271cfcb9)
Weakness	CWE-201 Information Exposure Through Sent Data	Introduced	d69bfad3fa , Jun 2026
Reported	3 Sep 2026	Report	HackerOne #3993870

DESCRIPTION

With the Schannel backend on Windows, `CURLSSLOPT_AUTO_CLIENT_CERT` lets the operating system choose a client certificate from the user's store and present it during the handshake. Since curl 8.21.0, libcurl computes the effective setting for each transfer and enables it only while the transfer is talking to its initial origin. After a redirect to a different origin, automatic certificate selection is switched off so that the user's certificate is not offered to a host the application did not choose.

The effective flag was written to `ssl_config_data`, however, and not to `ssl_primary_config`, which is what connection matching compares. The `CURLOPT_SSL_OPTIONS` bitmask is compared, but it is identical before and after the redirect. So when a redirect led to a host for which the pool already held a connection with a certificate presented on it, for example one opened by an earlier transfer that started at that host, libcurl reused that connection. The redirected request ran over a TLS session authenticated with the user's certificate, the outcome the origin check was meant to prevent.

lib/vtls/vtls_config.c · fix 00271cfcb9

```
(c1->verifystatus == c2->verifystatus) &&
+ (c1->auto_client_cert == c2->auto_client_cert) &&
...
+ sslc->primary.auto_client_cert =
+   Curl_peer_equal(origin, initial_origin) &&
+   !(options & CURLSSLOPT_AUTO_CLIENT_CERT);
...
- sslc->auto_client_cert = Curl_peer_equal(origin, initial_origin) &&
-   !(options & CURLSSLOPT_AUTO_CLIENT_CERT);
```

IMPACT

Affected users run libcurl or the curl tool on Windows with Schannel, enable automatic client certificates and follow redirects. A server that redirects such a client to another site can cause a request of its choosing to reach that site on a connection authenticated as the user, provided an authenticated connection to the target is already in the pool. The target is a host the user already trusts with the certificate, so nothing leaks to the redirecting server. Automatic certificate selection is off by default, and the timing condition narrows the flaw further, which is consistent with curl's Low rating.

RECOMMENDATION

A setting that libcurl derives per transfer, as opposed to one copied from an option, must be computed into the structure that connection matching compares. Otherwise the pool keys on the option and not on its effect. Users who cannot upgrade should avoid combining `CURLSSLOPT_AUTO_CLIENT_CERT` with `redirect following`, or name the certificate explicitly with `CURLOPT_SSLCERT`.

RESOLUTION

curl moved `auto_client_cert` into `ssl_primary_config`, copying and comparing it with the other connection settings, and changed Schannel to read it from the connection's configuration. The existing origin check was unchanged. A connection opened with automatic certificate selection is now reused only by transfers for which it is also enabled. The fix shipped in curl 8.23.0.

7 Credentials and Authentication

Findings in which credentials or authentication tokens could reach a party or a place they were not meant for.

AISLE-CURL-07

LOW

.netrc password sent for a different user on the same host

CVE	CVE-2026-8926	Disclosed	curl 8.21.0, 24 Jun 2026
Component	Authentication · .netrc	Scope	libcurl and curl tool
Affected	8.11.1 – 8.20.0	Fixed in	curl 8.21.0 (4ae1d7cc26)
Weakness	CWE-522 Insufficiently Protected Credentials	Introduced	e9b9bbac22 , Nov 2024
Reported	14 May 2026	Report	HackerOne #3735184

DESCRIPTION

When curl reads credentials from a .netrc file for a URL that names a user but gives no password, such as `https://user@example.com/`, libcurl looks for a machine entry that matches both the host and that login. The scanner counted a host match as success. If the file held an entry for the host under a different login, and none for the requested user, the scan ended with that entry’s password still in hand and returned it. libcurl then paired it with the username from the URL and used the pair to authenticate.

The pre-fix code in `netrc_finalize()` describes this outcome in its own comment:

```
lib/netrc.c · before the fix

if(!retcode) {
    /* success
     netrc_finalize() can return a password even when specific_login is set
     but our_login is false (e.g., host matched but the requested login
     never matched). See test 685. */
    const char *login = Curl_creds_has_user(ns->existing) ?
        ns->existing->user : ns->login;
    /* success without a password, set a blank one */
    const char *passwd = ns->password ? ns->password : "";
```

Only the last entry for the host mattered. Earlier non-matching entries were discarded when the next machine keyword was read.

IMPACT

libcurl and the curl tool are affected when .netrc lookup is enabled (`CURLOPT_NETRC`, `--netrc`, `--netrc-optional` or `--netrc-file`), the URL carries a username without a password, and the file has an entry for that host under some other login but none for the one requested. curl then sends another account’s password to the server, in a login attempt for the wrong user. The password goes to the host it was stored for, not to a third party, but it is exposed to anything that can see that exchange. The trigger is a local file layout that no remote party controls, which is why curl rated the issue Low.

RECOMMENDATION

A lookup for a specific login should succeed only when an entry with that login is found. A host match on its own must not supply a password for a different user. Users who cannot upgrade should give every login they request its own entry, supply the password through the URL or `CURLOPT_USERPWD` instead, or stop using .netrc for hosts that have more than one account.

RESOLUTION

curl rewrote the `.netrc` scanner so that a request for a named user fails unless that user's entry is found. Unit test 1304 gained a case confirming that a host entry with a non-matching login no longer yields its password. The fix shipped in curl 8.21.0, and the advisory also lists the 8.14.2, 8.16.1 and 8.20.1 maintenance releases as fixed.

AISLE-CURL-18

LOW

Clearing the AWS SigV4 option does not disable request signing

CVE	CVE-2026-86409	Disclosed	curl 8.23.0, 14 Oct 2026
Component	HTTP · AWS SigV4 authentication	Scope	libcurl
Affected	7.75.0 – 8.21.0	Fixed in	curl 8.22.0 (c3c70440e8)
Weakness	CWE-201 Information Exposure Through Sent Data	Introduced	08e8455ddd , Jul 2020
Reported	27 Aug 2026	Report	HackerOne #3973135

DESCRIPTION

Setting `CURLOPT_AWS_SIGV4` to a provider string makes libcurl sign HTTP requests with AWS Signature Version 4. The option handler does this by replacing the handle's authentication mask with `CURLAUTH_AWS_SIGV4`, since Basic would otherwise stay selected by default. To stop signing on a reused easy handle, an application sets the option back to `NULL`. That freed the stored string but left the mask alone. The fix adds the missing branch:

```
lib/setopt.c · fix c3c70440e8
```

```
if(CURL_EASY_STR(data, STRING_AWS_SIGV4))
    s->httpauth = CURLAUTH_AWS_SIGV4;
+ else
+     s->httpauth &= ~(uint32_t)CURLAUTH_AWS_SIGV4;
break;
```

With the bit still set and no provider string, libcurl fell back to its default provider, `aws:amz`, and kept signing every later request on the handle with whatever credentials the handle held. Only an explicit change to `CURLOPT_HTTPAUTH` would stop it, and an application that had just cleared the SigV4 option had no reason to make one.

IMPACT

The flaw affects libcurl applications that reuse one easy handle for several destinations, enabling SigV4 for one request and clearing it before the next. The later request still carries an AWS SigV4 Authorization header. That header contains the access key ID in clear, the credential scope, and a signature computed with the secret key, and it goes to a server that was never meant to receive any of them. The secret key itself is not sent, and the signature covers only that one request. Together with the need for a specific sequence of option changes on a reused handle, this is why curl rated the issue Low.

RECOMMENDATION

Clearing an option that switched on an authentication method should switch the method off again, returning the handle to the state it would be in had the option never been set. Applications that cannot upgrade should use a fresh easy handle (or `curl_easy_reset()`) for each destination, or set `CURLOPT_HTTPAUTH` explicitly after clearing `CURLOPT_AWS_SIGV4`, and clear `CURLOPT_USERPWD` once the AWS credentials are no longer needed.

RESOLUTION

curl now removes the CURLAUTH_AWS_SIGV4 bit whenever CURLOPT_AWS_SIGV4 is set to NULL, matching how CURLAUTH_HTTPSIG was already handled. Test 1978 gained a second transfer, made after the option is cleared, which must go out unsigned. The fix shipped in curl 8.22.0. The advisory was held back and published with curl 8.23.0.

AISLE-CURL-22

LOW

Redirected transfer reuses a connection authenticated with NTLM or Negotiate

CVE	CVE-2026-86413	Disclosed	curl 8.23.0, 14 Oct 2026
Component	HTTP · Negotiate (SPNEGO)	Scope	libcurl and curl tool
Affected	8.21.0 – 8.22.0	Fixed in	curl 8.23.0 (43029e2ec3)
Weakness	CWE-201 Information Exposure Through Sent Data	Introduced	8f71d0fde5 , May 2026
Reported	3 Sep 2026	Report	HackerOne #3993549

DESCRIPTION

NTLM and Negotiate authenticate a connection, not a request. Once the handshake is complete, the server treats every later request on that connection as coming from the authenticated user. libcurl's connection matcher accounts for this: a transfer that wants NTLM or Negotiate prefers a pooled connection to the same host on which the handshake has already completed, and is forced onto it when one exists. `url_attach_existing()` decided whether a transfer wanted these methods from the requested authentication mask and the protocol alone.

That left out redirects. Unless `CURLOPT_UNRESTRICTED_AUTH` or `--location-trusted` is set, libcurl authenticates only to the origin a transfer started at, and after a cross-origin redirect it stops sending `Authorization` headers. Connection matching did not apply the same rule. A redirected transfer still counted as an NTLM or Negotiate transfer, so it could be placed on a pooled connection already authenticated to the redirect target. The request then ran under the user's identity even though curl had sent no credentials for it.

IMPACT

Applications and curl tool invocations that use NTLM or Negotiate and follow redirects are affected. A server that can redirect the transfer could send it to a host where the user already has an authenticated connection in the pool, and have its request carried out there as that user, at an origin the transfer was not permitted to authenticate to. The attacker borrows an authenticated connection without learning any credential. It also needs such a connection to exist in the same pool at the right moment, which limits how easily the flaw can be used and is consistent with curl's Low rating.

RECOMMENDATION

Every decision that depends on authentication state should apply the same origin rule, and connection reuse is one of them. State that authenticates a connection should carry over to a transfer only when the transfer may authenticate to that origin. Until they can upgrade, users should avoid combining NTLM or Negotiate with redirect-following when the redirecting server is not trusted.

RESOLUTION

curl added a `Curl_auth_allowed_to_host()` check to the conditions that mark a transfer as wanting NTLM or Negotiate. A transfer that has left its initial origin is therefore no longer matched with, or forced onto, a connection authenticated with either method. The fix shipped in curl 8.23.0. It did not cover the case where curl opens a new connection to the redirect target and starts a handshake there, which is AISLE-CURL-34.

AISLE-CURL-23

LOW

Malicious proxy can harvest origin credentials during CONNECT

CVE	CVE-2026-86414	Disclosed	curl 8.23.0, 14 Oct 2026
Component	HTTP proxy · CONNECT tunnel	Scope	libcurl and curl tool
Affected	7.10.8 – 8.22.0	Fixed in	curl 8.23.0 (2956af6c74)
Weakness	CWE-201 Information Exposure Through Sent Data	Introduced	52ceab5e41 , Sep 2003
Reported	3 Sep 2026	Report	HackerOne #3993566

DESCRIPTION

To reach an origin through an HTTP proxy tunnel, libcurl first sends the proxy a CONNECT request. Until the proxy answers with a 2xx status, nothing has been relayed to the origin, so every response at that stage comes from the proxy. The only challenge a proxy can legitimately send there is a 407 with Proxy-Authenticate. The tunnel code also accepted a 401 with WWW-Authenticate and passed that challenge to `Curl_http_input_auth()` as if the origin had sent it:

```
lib/cf-h1-proxy.c · fix 2956af6c74
```

```
- if((checkprefix("WWW-Authenticate:", header) && (401 == k->httpcode)) ||  
-   (checkprefix("Proxy-authenticate:", header) && (407 == k->httpcode))) {  
+ if(checkprefix("Proxy-authenticate:", header) && (407 == k->httpcode)) {  
  
    bool proxy = (k->httpcode == 407);
```

The same condition appeared in the CONNECT-UDP header handler and in the generic tunnel-response code in `lib/http_proxy.c`. The forged challenge picked the origin authentication method, and libcurl then retried the CONNECT. That retry is built by `Curl_http_output_auth()`, which emits the origin's Authorization header alongside any proxy credentials. The origin credentials, computed for a challenge the proxy chose, were therefore sent to the proxy itself, in a request that never reaches the origin.

IMPACT

libcurl and the curl tool are affected when they tunnel through an HTTP proxy (HTTPS through a proxy, or `CURLOPT_HTTPPROXYTUNNEL` and `--proxytunnel`) with origin credentials configured. A malicious or compromised proxy can answer any CONNECT with a forged 401 and receive the origin credentials in the next request, before any tunnel exists. When the application allows several methods, as with `CURLAUTH_ANY` or `--anyauth`, the proxy can ask for Basic and collect the user name and password in cleartext. This works even when the origin is HTTPS, because the credentials never enter the TLS session. The attack needs a hostile proxy in the path and origin credentials configured for the transfer, which is consistent with curl's Low rating.

RECOMMENDATION

During CONNECT, a client should act only on proxy challenges and treat a 401 as a failed tunnel, because an origin challenge cannot appear before the tunnel exists. Users who cannot upgrade should use only trusted proxies when origin credentials are configured. They should also set `CURLOPT_HTTPAUTH` to the single method the origin needs rather than `CURLAUTH_ANY`, so that a forged challenge cannot select a weaker one.

RESOLUTION

curl removed the 401 and WWW-Authenticate handling from all three CONNECT response paths, in `lib/cf-h1-proxy.c` and `lib/http_proxy.c`. A proxy that answers CONNECT with a 401 now fails the connection attempt. The new test 2173 verifies this. The fix shipped in curl 8.23.0. The behaviour dated from 2003, when the CONNECT loop was first reworked to handle NTLM and Digest negotiation.

AISLE-CURL-34

LOW

Fresh Kerberos ticket minted for a host chosen by a redirect

CVE	CVE-2026-94265	Disclosed	curl 8.23.0, 14 Oct 2026
Component	HTTP · Negotiate (SPNEGO)	Scope	libcurl and curl tool
Affected	8.21.0 – 8.22.0	Fixed in	curl 8.23.0 (557a52f21e)
Weakness	CWE-200 Exposure of Sensitive Information to an Unauthorized Actor	Introduced	8f71d0fde5 , May 2026
Reported	18 Sep 2026	Report	HackerOne #4040551

DESCRIPTION

The fix for AISLE-CURL-22 kept redirected transfers off connections that were already NTLM- or Negotiate-authenticated. This finding was reproduced on a snapshot that already contained that fix, and it covers the other path: curl opens a new connection to the redirect target and runs the handshake itself. `Curl_http_output_auth()` allowed origin authentication when the transfer held credentials or when the host was one the transfer may authenticate to. For most schemes the first condition is safe, because the credentials held are the ones sent. Negotiate is different. With GSS-API it draws on the user's Kerberos credential cache, whatever credentials the transfer holds. If the redirected request carried any credentials at all, wherever they came from, `output_auth_headers()` requested a service ticket for the redirect target and sent it:

```
lib/http.c · fix 557a52f21e
```

```
- (!proxy && !Curl_checkheaders(data, STRCONST("Authorization")))) {  
+ (!proxy && !Curl_checkheaders(data, STRCONST("Authorization")) &&  
+  Curl_auth_allowed_to_host(data)) {  
    auth = "Negotiate";  
    result = Curl_output_negotiate(data, conn, proxy);
```

IMPACT

libcurl and the curl tool are affected when Negotiate is enabled (`CURLAUTH_NEGOTIATE` or `--negotiate`) and redirects are followed (`CURLOPT_FOLLOWLOCATION` or `-L`). A server that can redirect the transfer, and arrange for credentials to be present for the new target, can make curl obtain a fresh Kerberos ticket and send it, identifying the user, to a host of its choosing. The ticket is issued for that host's own service principal and cannot be replayed against other services, unless the application has enabled delegation with `CURLOPT_GSSAPI_DELEGATION`. Negotiate must be explicitly enabled, which is consistent with curl's Low rating.

RECOMMENDATION

A rule that limits authentication to permitted origins should depend on the destination, not on whether credentials happen to be present. It should also be enforced where the header is produced, so that every path, whether it reuses a connection or opens a new one, passes through the same check. Until they can upgrade, users should avoid combining Negotiate with automatic redirect-following towards servers they do not trust, and should leave GSS-API delegation disabled.

RESOLUTION

curl now sends an origin Negotiate token, or performs origin NTLM, only when the current host passes the `Curl_auth_allowed_to_host()` check. When the check fails, NTLM is marked done without sending anything. Proxy authentication is unchanged. The fix shipped in curl 8.23.0 alongside the fix for AISLE-CURL-22, so both the reused-connection and new-connection paths are closed in that release.

AISLE-CURL-35

LOW

.netrc password exposed in the effective URL after a relative redirect

CVE	CVE-2026-94266	Disclosed	curl 8.23.0, 14 Oct 2026
Component	Redirects · transfer information	Scope	libcurl and curl tool
Affected	7.62.0 – 8.22.0	Fixed in	curl 8.23.0 (7a6bd027d0)
Weakness	CWE-522 Insufficiently Protected Credentials	Introduced	46e164069d , Sep 2018
Reported	18 Sep 2026	Report	HackerOne #4041053

DESCRIPTION

CURLINFO_EFFECTIVE_URL, and `%{url_effective}` in the curl tool, report the URL a transfer ended at. libcurl answered with its internal working URL, which is not always the string the application supplied. When credentials came from `.netrc`, libcurl wrote the username and password back into its parsed URL handle:

lib/url.c · before the fix

```
result = Curl_creds_merge(ncreds_out->user, ncreds_out->passwd,
                          *pcreds, CREDNS_NETRC, pcreds);
...
/* for updated strings, we update them in the URL */
uc = curl_url_set(data->state.uh, CURLUPART_USER,
                  Curl_creds_user(*pcreds), CURLU_URLENCODE);
if(!uc)
    uc = curl_url_set(data->state.uh, CURLUPART_PASSWORD,
                     Curl_creds_passwd(*pcreds),
                     CURLU_URLENCODE);
```

A relative redirect is resolved against that handle, so the new URL inherited the userinfo. The effective URL reported after the redirect therefore contained the `.netrc` password in plain text, although it never appeared in any URL the user or application gave.

IMPACT

libcurl and the curl tool are affected when `.netrc` supplies the credentials (`CURLOPT_NETRC`, `--netrc`, `--netrc-optional` or `--netrc-file`) and a relative redirect is followed. Anything that records the effective URL, such as request logs, telemetry, progress output or `--write-out` templates, ends up holding the password. The redirecting server decides whether this happens, but it learns nothing new, since it has already received the same credentials in the `Authorization` header. The exposure is to local consumers of the URL, which is consistent with curl's Low rating.

RECOMMENDATION

Interfaces that report a URL to the application should build that string without userinfo, rather than return an internal representation that may have picked up credentials from elsewhere. Users who cannot upgrade should not log `CURLINFO_EFFECTIVE_URL`, `CURLINFO_REDIRECT_URL`, `%{url_effective}` or `%{redirect_url}` when using `.netrc`, or should strip the userinfo before storing or displaying them.

RESOLUTION

`CURLINFO_EFFECTIVE_URL` now returns a copy parsed with libcurl's URL API, with user and password removed and the result cached with the transfer information. This applies to any credentials in the URL, not only those from `.netrc`. Test 1876 covers it. The fix shipped in curl 8.23.0. A follow-up by Filippo Tedeschi, [ce35977303](#), applied the same stripping to `CURLINFO_REDIRECT_URL` and `%{redirect_url}`, which had the same flaw for relative targets that were not followed. Test 1877 covers that case.

8 SSH Host Verification

Findings that weaken verification of an SSH server’s identity for SCP and SFTP transfers.

AISLE-CURL-01

LOW

wolfSSH backend never verifies the SFTP server’s host key

CVE	CVE-2025-10966	Disclosed	curl 8.17.0, 5 Nov 2025
Component	SFTP · wolfSSH backend	Scope	libcurl and curl tool
Affected	7.69.0 – 8.16.0	Fixed in	curl 8.17.0 (b011e3fcfb)
Weakness	CWE-322 Key Exchange without Entity Authentication	Introduced	6773c7ca65 , Jan 2020
Reported	23 Sep 2025	Report	HackerOne #3355218

DESCRIPTION

libcurl can use one of several SSH libraries for SFTP, selected when curl is built. The libssh and libssh2 backends check the server’s host key before authenticating, against a known-hosts file (CURLLOPT_SSH_KNOWNHOSTS), a pinned key fingerprint or an application callback. The wolfSSH backend, added in 7.69.0, did none of this. Its connect code created the session, registered a user-authentication callback and handed control to wolfSSH’s SFTP connect routine, which runs the key exchange and authentication in one go. curl never installed wolfSSH’s public-key check callback and read none of the host-key options, so any key the server offered was accepted. The backend even listed an SSH_HOSTKEY state, shared with the other backends, that its state machine never entered.

The omission dates from the day the backend was added. It was documented as incomplete, its tests were known to fail, and no user ever reported the gap.

IMPACT

Only builds that selected wolfSSH as the SSH backend are affected, and only for SFTP, since the backend never supported SCP. Against such a build, anyone who can intercept the connection to an SFTP server, for example through DNS spoofing or a hostile network, can present their own host key and pose as the server. They can read and alter the transferred files and capture the user’s password, which was the only working authentication method because public-key authentication was never implemented in this backend. curl rates this Low because wolfSSH was rarely chosen, documented as incomplete and not known to be used in production.

RECOMMENDATION

Every SSH backend should verify the host key before authenticating, and a backend that cannot do so should refuse to connect rather than carry on. Users of affected builds should rebuild curl with libssh or libssh2, which verify host keys, or avoid `sftp://` transfers with those builds.

RESOLUTION

curl chose removal over repair. curl 8.17.0 drops the wolfSSH backend entirely, deleting `lib/vssh/wolfssh.c` along with its build, CI and documentation support, on the grounds that nobody had ever reported a bug in it or asked for improvements. The advisory also lists the long-term support releases 8.16.1 and 8.14.2 as not affected.

AISLE-CURL-10

LOW

libssh host-key type mismatch reported as an unknown host

CVE	CVE-2026-9547	Disclosed	curl 8.21.0, 24 Jun 2026
Component	SCP/SFTP · libssh backend	Scope	libcurl
Affected	7.69.0 – 8.20.0	Fixed in	curl 8.21.0 (0b8dbbc63c)
Weakness	CWE-297 Improper Validation of Certificate with Host Mismatch	Introduced	507cf6a13d , Feb 2020
Reported	20 May 2026	Report	HackerOne #3751712

DESCRIPTION

An application that sets `CURLOPT_SSH_KNOWNHOSTS` and installs a `CURLOPT_SSH_KEYFUNCTION` callback leaves the final decision on an SSH server's host key to its own code. libcurl looks the host up in the `known_hosts` file, sums up the outcome as one of the `CURLKHMATCH_*` values and passes that verdict to the callback. The libssh backend derives the verdict from the state returned by `ssh_session_get_known_hosts_entry()`. One of those states, `SSH_KNOWN_HOSTS_OTHER`, means that the file holds a key for this host, but of a different type from the one the server has just offered. libcurl grouped it with the "not found" states.

```
lib/vssh/libssh.c · fix 0b8dbbc63c
```

```
case SSH_KNOWN_HOSTS_OK:
    keymatch = CURLKHMATCH_OK;
    break;
case SSH_KNOWN_HOSTS_OTHER:
+   keymatch = CURLKHMATCH_MISMATCH;
+   break;
case SSH_KNOWN_HOSTS_NOT_FOUND:
case SSH_KNOWN_HOSTS_UNKNOWN:
case SSH_KNOWN_HOSTS_ERROR:
    keymatch = CURLKHMATCH_MISSING;
```

A server that had switched from, say, an Ed25519 key to an RSA key therefore looked to the callback like a host it had never seen, rather than one whose key had changed.

IMPACT

Only libcurl built with the libssh backend is affected; the libssh2 backend classifies the case correctly. The application must also combine a `known_hosts` file with its own key callback. Without a callback, libcurl rejects anything short of an exact match, and the curl tool does not install one. The risk lies in callbacks that treat an unknown host more leniently than a changed key. A trust-on-first-use callback typically answers `CURLKHMATCH_MISSING` with `CURLKHSTAT_FINE_ADD_TO_FILE`. Faced with an attacker who intercepts the connection and offers a key of another type, such a callback would accept the impostor and have libcurl record its key in the `known_hosts` file. The narrow set of affected configurations explains the Low rating.

RECOMMENDATION

Code that translates a backend's known-hosts status into libcurl's `CURLKHMATCH_*` values should keep "no record for this host" and "a different key is recorded" apart, because callbacks base their policy on that difference. Any status it cannot classify should be reported as a mismatch. Applications that cannot upgrade should treat `CURLKHMATCH_MISSING` as grounds for rejection rather than automatic acceptance, or build libcurl with libssh2, as curl's advisory suggests.

RESOLUTION

The fix, submitted by AISLE, gives `SSH_KNOWN_HOSTS_OTHER` its own case returning `CURLKHMATCH_MISMATCH`, so the callback sees a changed host key for what it is. It shipped in curl 8.21.0. The advisory also lists the long-term support releases 8.20.1, 8.16.1 and 8.14.2 as not affected.

AISLE-CURL-19

LOW

curl tool skips SSH host verification after a redirect to SFTP or SCP

CVE	CVE-2026-86410	Disclosed	curl 8.23.0, 14 Oct 2026
Component	curl tool · SCP/SFTP options	Scope	curl tool
Affected	7.81.0 – 8.22.0	Fixed in	curl 8.23.0 (6ae9befb2f)
Weakness	CWE-297 Improper Validation of Certificate with Host Mismatch	Introduced	18270893ab , Nov 2021
Reported	27 Aug 2026	Report	HackerOne #3973176

DESCRIPTION

The curl tool turns its command-line options into libcurl settings once per URL, before the transfer starts. `ssh_setopts()` in `src/config2setopts.c` applies the SSH settings: the private and public key files, the `--hostpubsha256` and `--hostpubmd5` fingerprints, and the `known_hosts` file the tool locates on the user's behalf. The function returned early unless the URL on the command line was itself an `scp://` or `sftp://` URL.

src/config2setopts.c · fix 6ae9befb2f

```
-static CURLcode ssh_setopts(struct OperationConfig *config, CURL *curl,
-                             const char *use_proto)
+static CURLcode ssh_setopts(struct OperationConfig *config, CURL *curl)
{
    CURLcode result;

-    if(use_proto != proto_scp && use_proto != proto_sftp)
+    if(!proto_scp && !proto_sftp)
        return CURLE_OK;
```

A transfer can change scheme, though, when libcurl follows a redirect. One that began as HTTP and was redirected to SFTP or SCP reached the SSH server with none of these options set. libcurl checks an SSH host key only when told how, so it accepted whatever key the server presented. CVE-2026-12064 left the same options unset by a different route: the tool guessed the wrong scheme for a schemeless URL used with `--proto-default`. Here the scheme is read correctly, but from the wrong URL.

IMPACT

Only the curl tool is affected, and only when it follows redirects with `--location` and the user has also widened `--proto-redir` to include SCP or SFTP. By default, libcurl follows redirects only to HTTP, HTTPS, FTP and FTPS. With both options in place, the server the user contacts, or anyone able to tamper with an unprotected first request, can send curl on to an SSH server of the attacker's choosing. curl connects without checking that server's host key and disregards any fingerprint the user supplied, so the attacker can impersonate the SSH server for the rest of the transfer. The explicit opt-in to SSH redirects keeps the rating at Low.

RECOMMENDATION

A client should apply security options for every protocol a transfer may end up using, not only the one it starts with, so that a redirect can never land on a protocol whose protections were skipped. Until they upgrade, users should keep SCP and SFTP out of `--proto-redir`, as the default does, and give curl `sftp://` and `scp://` URLs directly rather than reach them through a redirect.

RESOLUTION

In curl 8.23.0, `ssh_setopt()` no longer looks at the starting URL: it applies the SSH options whenever the tool is built with SCP or SFTP support. Because it now runs for HTTP transfers too, the tool no longer refuses to start when it cannot find a `known_hosts` file, and instead points libcurl at the default location in the user's home directory. Test 1679 was adjusted to match.

9 Cookie and HSTS State

Findings in the parsing, scoping and persistence of cookies and HSTS policies.

AISLE-CURL-15

LOW

A tab before the Secure attribute silently drops the cookie's Secure flag

CVE	CVE-2026-80255	Disclosed	curl 8.22.0, 2 Sep 2026
Component	HTTP cookies · Set-Cookie parser	Scope	libcurl and curl tool
Affected	8.13.0 – 8.21.0	Fixed in	curl 8.22.0 (4f6aa41a01)
Weakness	CWE-201 Information Exposure Through Sent Data	Introduced	1aea05a6c2 , Feb 2025
Reported	26 Aug 2026	Report	HackerOne #3972395

DESCRIPTION

curl parses a Set-Cookie: header as a series of semicolon-separated items: the cookie's name and value first, then attributes such as Path, Domain or the bare keyword `Secure`. The parser reads each item's name with `curlx_str_cspn()`, which returns the run of characters before the first delimiter, and moves on to the next item only if it then finds a semicolon. When the parser was rewritten on top of curl's string-parsing helpers in curl 8.13.0, the tab character ended up in that delimiter set.

```
lib/cookie.c · fix 4f6aa41a01

/* we have a <name>=<value> pair or a stand-alone word here */
- if(!curlx_str_cspn(&ptr, &name, ";\t\r\n=")) {
+ if(!curlx_str_cspn(&ptr, &name, ";\r\n=")) {
    struct Curl_str val;
    bool sep = FALSE;
    curlx_str_trimblanks(&name);
```

If a server put a tab after the semicolon, before `Secure`, the run came back empty and nothing was read. The loop then found a tab where it expected a semicolon, and stopped. curl stored the cookie with the attributes parsed up to that point and without its `Secure` flag; any attributes later in the header were lost as well. A space in the same position was harmless, because it falls inside the run and is trimmed off the name.

IMPACT

Any application that accepts cookies is affected, the curl tool included, whenever a server separates `Secure` from the preceding item with a tab. A cookie the server meant to confine to HTTPS is stored without that restriction. curl will then include it in plain HTTP requests to the same host, where a network attacker can read it. An attacker cannot bring this about alone. It depends on the server's unusual formatting and on the client later making, or being led into, an unencrypted request to that host, which keeps the rating at Low.

RECOMMENDATION

A tokeniser should treat each whitespace character consistently: a character it trims as a blank should not also end a token. Attributes that carry security meaning, such as `Secure`, deserve particular care, since losing one fails open. Users who cannot upgrade should avoid plain HTTP to hosts whose cookies matter, as curl's advisory recommends, and server operators can separate attributes with spaces.

RESOLUTION

curl removed the tab from the delimiter set, so a leading tab now falls inside the item and is trimmed like any other blank. The fix shipped in curl 8.22.0 with test 2885, which sets one cookie with a tab before `Secure` and one with a space. The advisory lists the long-term support releases 8.20.1, 8.16.1 and 8.14.2 as not affected.

AISLE-CURL-17

LOW

Public-suffix host can set a cookie that is sent to all of its siblings

CVE	CVE-2026-82209	Disclosed	curl 8.22.0, 2 Sep 2026
Component	HTTP cookies · Public Suffix List	Scope	libcurl and curl tool
Affected	7.46.0 – 8.21.0	Fixed in	curl 8.22.0 (95c1e8915d)
Weakness	CWE-201 Information Exposure Through Sent Data	Introduced	e77b5b7453 , Sep 2015
Reported	26 Aug 2026	Report	HackerOne #3972385

DESCRIPTION

In builds with libpsl, libcurl consults the Public Suffix List before it accepts a cookie's `Domain` attribute. The check stops a site from setting a cookie for a shared suffix such as `co.uk` or `github.io`, which would then be sent to every site beneath it. libpsl makes one exception: a host may name its own domain, even when that domain is itself a listed suffix. curl accepted such a cookie but stored it like any other cookie with an explicit `Domain`, with the `tailmatch` flag set so that it matches the domain and all of its subdomains. A cookie that `github.io` set with `Domain=github.io` was therefore sent to `attacker.github.io` and to every other site under that suffix. RFC 6265 requires the opposite: a cookie whose domain is a public suffix and equals the request host is to be kept host-only.

IMPACT

The flaw affects libcurl and the curl tool in builds with PSL support, and needs two things to line up. A host that is itself a public suffix must set a cookie naming its own domain, and curl must later send a request to a sibling host under that suffix. On a shared hosting platform whose apex serves content, those siblings belong to other tenants, and any of them would receive the apex's cookie, possibly a session token. As curl's advisory notes, an attacker cannot plant such a cookie, only receive one, which is why the rating is Low.

RECOMMENDATION

When a cookie's domain is a public suffix and equals the host that set it, the cookie should be accepted only as host-only, as RFC 6265 prescribes, never scoped to the whole suffix. Until they upgrade, users who visit both a public-suffix apex and sites beneath it should not share a cookie jar between those transfers. Operators of such an apex can avoid sending a `Domain` attribute that names their own host.

RESOLUTION

curl now clears the `tailmatch` flag whenever it accepts a cookie whose domain is a public suffix and equals the host that set it.

lib/cookie.c · fix 95c1e8915d

```
    acceptable = psl_is_cookie_domain_acceptable(psl, lcase, lcookie);
+
+    /* if the cookie is acceptable, but is set for a PSL domain, then it
+       cannot be tailmatching */
+    if(acceptable && co->tailmatch &&
+       curl_strequal(co->domain, domain) &&
+       psl_is_public_suffix(psl, lcookie))
+       co->tailmatch = FALSE;
```

The same change moved the PSL check earlier in `Curl_cookie_add()`, ahead of the `__Host-` prefix check that relies on the flag. The fix shipped in curl 8.22.0 with a new test, 2318, and updated expectations in two existing ones. The advisory lists the long-term support releases 8.20.1, 8.16.1 and 8.14.2 as not affected.

AISLE-CURL-21

LOW

Unloadable server cookie causes curl to erase its own cookie jar

CVE	CVE-2026-86412	Disclosed	curl 8.23.0, 14 Oct 2026
Component	HTTP cookies · cookie jar	Scope	libcurl and curl tool
Affected	8.18.0 – 8.22.0	Fixed in	curl 8.23.0 (6ded3e5944)
Weakness	CWE-754 Improper Check for Unusual or Exceptional Conditions	Introduced	3f0629ca44 , Nov 2025
Reported	27 Aug 2026	Report	HackerOne #3973201

DESCRIPTION

`cookie_load()` reads a cookie file one line at a time and hands each line to `Curl_cookie_add()`. Unusable lines are meant to be skipped, and a comment in the loop said that failures were not passed back to the caller. The loop condition said otherwise: any error, from reading a line or from adding its cookie, ended the load at that line.

lib/cookie.c · fix 6ded3e5944

```
    result = Curl_cookie_add(data, ci, lineptr, NULL, NULL,
                             (headerline ? COOKIE_HTTPHEADER : 0) |
                             COOKIE_NOEXPIRE | COOKIE_SECURE |
                             (flags & COOKIE_NOPSL));
-    /* File reading cookie failures are not propagated back to the
-       caller because there is no way to do that */
+    /* Ignore individual cookie problems unless we ran out of memory */
+    if(result != CURLE_OUT_OF_MEMORY)
+        result = CURLE_OK;
+    }
+    } while(!result && !eof);
```

Since the cookie code's error handling was reworked in curl 8.18.0, that error also reaches `Curl_pretransfer()` and the transfer fails. The jar is still written back. `cookie_load()` marks the cookie store as running even when it returns an error, and `Curl_flush_cookies()` takes that flag to mean loading completed. When the handle is closed, the partial store replaces the file named by `CURLOPT_COOKIEJAR`, and every cookie after the failing line is gone from disk. AISLE-CURL-28 is a separate flaw on the reading side, where a line too long for the loader's buffer stops the read.

IMPACT

Users are affected when they load and save the same file: the curl tool with `-b` and `-c` pointing at one path, or a libcurl application that sets `CURLOPT_COOKIEFILE` and `CURLOPT_COOKIEJAR` alike. If a cookie a server has set is saved in a form the loader then rejects, the next run fails and rewrites the jar without every cookie that followed it, including those for unrelated sites. The result is lost sessions and preferences,

not disclosure. It requires this persistence pattern, which is why curl rated it Low.

RECOMMENDATION

A loader for a persistent store should skip entries it cannot use and give up only on conditions, such as running out of memory, that make the whole result untrustworthy. When loading does fail, the store should not be written back, since saving an incomplete copy turns a read error into data loss. Users who cannot upgrade can keep the input and output cookie files separate, or back up a shared jar before each run.

RESOLUTION

curl 8.23.0 changed `cookie_load()` to discard any error from adding an individual cookie other than `CURLE_OUT_OF_MEMORY`, as the excerpt shows, so one unusable entry no longer ends the load. In the affected releases, however, `Curl_cookie_add()` could fail only when memory ran out. The error a server could actually provoke came from reading an over-long line, and that path was closed by the fix for AISLE-CURL-28, also in curl 8.23.0. Both fixes are needed to close this finding.

AISLE-CURL-25

LOW

HSTS cache import silently drops a longer-lived subdomain policy

CVE	CVE-2026-87862	Disclosed	curl 8.23.0, 14 Oct 2026
Component	HSTS · cache import	Scope	libcurl and curl tool
Affected	8.11.0 – 8.22.0	Fixed in	curl 8.23.0 (a1614c3bcf)
Weakness	CWE-201 Information Exposure Through Sent Data	Introduced	a94973805d , Oct 2024
Reported	26 Aug 2026	Report	HackerOne #3972357

DESCRIPTION

When libcurl loads a persisted HSTS cache, from the file named by `CURLOPT_HSTS` (`--hsts` in the curl tool) or through `CURLOPT_HSTSREADFUNCTION`, each entry passes through `hsts_add_host_expire()`. That function first looks for an existing record of the host. It creates a new entry if the lookup finds nothing, and updates the existing one only if the names match exactly.

The lookup was the problem. Whenever the entry being loaded carried `includeSubDomains`, it asked `hsts_check()` for a subdomain-aware match. If the cache already held a parent such as `.example.test`, a later line for `.child.example.test` matched that parent. The result was neither empty nor an exact match, so neither branch ran and the child entry was dropped, even when its expiry lay well beyond the parent's.

```
lib/hsts.c · fix a1614c3bcf

/* only add it if not already present */
- e = hsts_check(h, host, hostlen, subdomain);
+ e = hsts_check(h, host, hostlen, FALSE);
if(!e)
    result = hsts_create(h, host, hostlen, subdomain, expires);
/* 'host' is not necessarily null-terminated */
else if((hostlen == strlen(e->host) &&
        curl_strnequal(host, e->host, hostlen))) {
```

The exact-name guard on the update branch came from the fix for CVE-2024-9681, which stopped a child's expiry from overwriting its parent's. That fix left the lookup unchanged, and the earlier overwrite became an omission.

IMPACT

libcurl applications and the curl tool are affected when they persist HSTS state and the cache holds a parent domain and a subdomain that both carry `includeSubDomains`, with the parent listed first. At first nothing changes, because the parent's policy still covers the child. Once the parent expires, however, the child and its own subdomains lose HSTS protection long before their policy intended, and plain `http://` URLs for them are no longer upgraded to HTTPS. The dropped entry is also absent from the next save, so the loss is permanent. An attacker cannot cause the condition and can only exploit it from a network position during that window, which is consistent with the Low rating.

RECOMMENDATION

The duplicate check during import should compare host names exactly, since its only job is to find an earlier record of the same host. Subdomain matching belongs in the connection-time lookup that decides whether a URL must be upgraded. Applications that cannot upgrade and load entries through `CURLOPT_HSTSREADFUNCTION` can avoid the problem by supplying more specific hosts before their parent domains, because the lookup only compares against entries already loaded.

RESOLUTION

curl changed the import lookup to disable subdomain matching, as the excerpt shows. A loaded entry is now merged only with a record for exactly the same host and is otherwise created in its own right. Test 3109 covers a parent that expires before its child. The fix shipped in curl 8.23.0.

AISLE-CURL-26

LOW

Extreme HSTS max-age value wipes the persisted HSTS cache

CVE	CVE-2026-87863	Disclosed	curl 8.23.0, 14 Oct 2026
Component	HSTS · cache persistence	Scope	libcurl and curl tool
Affected	8.2.0 – 8.22.0	Fixed in	curl 8.23.0 (08d50a791b)
Weakness	CWE-201 Information Exposure Through Sent Data	Introduced	0c667188e0 , Jul 2023
Reported	26 Aug 2026	Report	HackerOne #3972373

DESCRIPTION

A Strict-Transport-Security response header tells libcurl for how many seconds a host must be treated as HTTPS-only. `Curl_hsts_parse()` accepted any `max-age` up to the largest `time_t` and added it to the current time, so a server could create an entry whose expiry lay far beyond any year a calendar conversion can represent.

The damage happened when the cache was saved. `Curl_hsts_save()` writes each entry through `hsts_out()`, which converts the expiry with `curlx_gmtime()`. For such an entry the conversion failed and the loop stopped. On the error path the temporary file was deleted and never renamed into place. That would have been harmless, except that `Curl_fopen()` had already opened the real cache file for writing, and so truncated it, before redirecting output to the temporary file. The persisted cache was left empty.

```
lib/hsts.c · before the fix

for(e = Curl_llist_head(&h->list); e; e = n) {
    struct stsentry *sts = Curl_node_elem(e);
    n = Curl_node_next(e);
    result = hsts_out(sts, out);
    if(result)
        break;
}
curlx_fclose(out);
if(!result && tempstore && curlx_rename(tempstore, file))
    result = CURLE_WRITE_ERROR;
if(result && tempstore)
    unlink(tempstore);
```

IMPACT

Any HTTPS server visited by a libcurl application or the curl tool with HSTS persistence enabled can send the header; it needs no other position. The next save, which happens when the easy handle is closed, then erases every stored entry, including those for unrelated hosts. The error is discarded at that point, so the application is never told. Later runs start with no HSTS state and will follow plain `http://` URLs to hosts that should have been upgraded. Taking advantage of that still requires a separate network attacker able to intercept those requests, which is why curl rates the issue Low.

RECOMMENDATION

Values supplied by a server should be bounded when they are parsed, well within the range the date code can handle. Separately, a problem with one entry should not abort the whole save, and a failed save should leave the previous file intact rather than a truncated one. Until they can upgrade, users who depend on a persisted HSTS cache can keep a backup of the file, or load it with `CURLHSTS_READONLYFILE` set in `CURLOPT_HSTS_CTRL` so that libcurl never rewrites it.

RESOLUTION

curl 8.23.0 caps `max-age` at two years during parsing, including values that overflow, so no entry can reach an expiry the date conversion rejects. `hsts_out()` now skips an entry it cannot format, and the save loop writes the remaining entries regardless. Test 1862 checks that an extreme `max-age` leaves existing entries intact. The fix did not change how `Curl_fopen()` opens the destination file.

AISLE-CURL-28

LOW

Oversized server cookie breaks later loads of the cookie jar

CVE	CVE-2026-92386	Disclosed	curl 8.23.0, 14 Oct 2026
Component	HTTP cookies · cookie jar	Scope	libcurl and curl tool
Affected	8.7.0 – 8.22.0	Fixed in	curl 8.23.0 (445aa87e13)
Weakness	CWE-754 Improper Check for Unusual or Exceptional Conditions	Introduced	142ac257b3, Feb 2024
Reported	11 Sep 2026	Report	HackerOne #4021979

DESCRIPTION

A cookie jar holds one cookie per line in the Netscape format: domain, subdomain flag, path, secure flag, expiry, name and value, separated by tabs, with an optional `#HttpOnly_` prefix. `cookie_load()` reads the file with `Curl_get_line()` into a buffer capped at `MAX_COOKIE_LINE`, 5,000 bytes. The same constant limits an incoming `Set-Cookie:` header, but a saved line is not the header. It carries the name, value, domain and path, and adds the flags, the expiry and the separators. A large cookie that curl accepted could therefore be saved as a line that curl would not read back.

```
lib/cookie.c · fix 445aa87e13
```

```
    struct dynbuf buf;
    bool eof = FALSE;
    curlx_dyn_init(&buf, MAX_COOKIE_LINE);
+   /* accept a little longer lines so that a full cookie fits and does not
+    abort the file read */
+   curlx_dyn_init(&buf, MAX_COOKIE_LINE + 48);
    do {
        result = Curl_get_line(&buf, fp, &eof);
```

Before curl 8.7.0, `Curl_get_line()` skipped an over-long line and carried on. Since its conversion to a dynamic buffer, such a line is an error that ends the read, and no cookie below it is loaded. The fix for AISLE-CURL-21 does not cover this path, because the error arises while the line is being read, before any cookie is parsed.

IMPACT

Any libcurl application or curl tool user that persists cookies is exposed once a server has set a large enough cookie and curl has saved it. On the next load, the cookies after that line are missing. From curl 8.7.0 to 8.17.0 the transfer carries on without them; from 8.18.0 the load error fails the transfer. Where the same file is also the output jar, the save at the end writes back only what was loaded, the loss described in AISLE-CURL-21. The consequence is lost state rather than exposed data, which is why curl rated it Low.

RECOMMENDATION

Anything a program writes to a persistent store it must be able to read back, so the reader's limits should follow from what the writer can produce rather than from a separate constant. A reader that meets an over-long line should skip that entry, as curl did before 8.7.0, rather than stop. Until they upgrade, users can keep separate input and output cookie files so that a truncated load cannot overwrite the original.

RESOLUTION

curl 8.23.0 raises the loader's limit by 48 bytes, as the excerpt shows, so that a full-size cookie saved by curl fits and no longer aborts the read. Only the reading side changed: the limits on incoming cookies and the format of saved lines are as before. A line longer than the new limit, such as one added by hand, still ends the read.

10 Memory Safety

Use-after-free, double-free and unbounded recursion flaws in libcurl.

AISLE-CURL-06**MEDIUM**

GSASL context freed twice during SASL mechanism negotiation

CVE	CVE-2026-8925	Disclosed	curl 8.21.0, 24 Jun 2026
Component	SASL · GNU SASL (IMAP, POP3, SMTP)	Scope	libcurl and curl tool
Affected	8.15.0 – 8.20.0	Fixed in	curl 8.21.0 (3da249e1f0)
Weakness	CWE-415 Double Free	Introduced	ab650379a8 , Jun 2025
Reported	14 May 2026	Report	HackerOne #3735193

DESCRIPTION

libcurl can delegate SCRAM-SHA-256 and SCRAM-SHA-1 authentication for IMAP, POP3 and SMTP to GNU SASL (`libsasl`). During mechanism selection, `sasl_choose_gsasl()` calls `Curl_auth_gsasl_is_supported()` for each SCRAM variant that both the server and the user allow. That function creates a GSASL context with `gsasl_init()` and then tries `gsasl_client_start()`. If the start failed, it released the context with `gsasl_done()` but left the pointer in the per-connection `gsasldata` structure.

Since the authentication state moved into connection metadata with an automatic destructor, that structure is cleaned up when the connection closes. The destructor calls `Curl_auth_gsasl_cleanup()`, which hands the stored pointer to `gsasl_done()` once more. Unless a later successful attempt had replaced it, the same context was freed a second time. The fix is a single line:

lib/vauth/gsaslc · fix 3da249e1f0

```
res = gsasl_client_start(gsasl->ctx, mech, &gsasl->client);
if(res != GSASL_OK) {
    gsasl_done(gsasl->ctx);
+   gsasl->ctx = NULL;
    return FALSE;
}
```

IMPACT

libcurl and the curl tool are affected when built with the GNU SASL backend and authenticating to an IMAP, POP3 or SMTP server that advertises a SCRAM mechanism which `gsasl_client_start()` then fails to start, for example because the installed `libsasl` does not implement it. The server chooses which mechanisms to offer; the local library decides whether starting one fails. Freeing the same heap block twice can corrupt allocator state and at least crash the process. curl considers it hard for an attacker to control which memory the second free affects, but cannot rule out abuse because the sequence and timing depend partly on server behaviour. It rates the issue Medium.

RECOMMENDATION

A pointer released outside its owner's destructor should be cleared immediately, so that the destructor finds an empty slot and does nothing. The other authentication objects moved into connection metadata by the same change deserve the same check. Users who cannot upgrade can restrict SASL to a mechanism that GNU SASL does not handle, using `CURLOPT_LOGIN_OPTIONS` (`--login-options` in the curl tool), or avoid IMAP, POP3 and SMTP with such builds.

RESOLUTION

curl now clears the context pointer as soon as the failed start has released it, as the excerpt shows. The cleanup routine can pass a NULL client handle to `gsasl_finish()`, so the build also requires `libgsasl 1.6.0` or later, whose `gsasl_finish()` accepts NULL. The fix shipped in curl 8.21.0, and the advisory lists the long-term support releases 8.16.1 and 8.20.1 as fixed.

AISLE-CURL-09

LOW

Use-after-free when a transfer is paused from the socket callback

CVE	CVE-2026-9080	Disclosed	curl 8.21.0, 24 Jun 2026
Component	Multi interface · event-based socket API	Scope	libcurl
Affected	8.13.0 – 8.20.0	Fixed in	curl 8.21.0 (5ab34cba42)
Weakness	CWE-416 Use After Free	Introduced	cfc657a48d , Feb 2025
Reported	19 May 2026	Report	HackerOne #3749204

DESCRIPTION

An application that drives libcurl with `curl_multi_socket_action()` learns which sockets to watch through its `CURLMOPT_SOCKETFUNCTION` callback. For each socket, libcurl keeps a `mev_sh_entry` in a hash table, counting the transfers that want to read or write. When that combined interest changes, `mev_sh_entry_update()` invokes the callback and, once it returns, records the new state in the entry.

`curl_easy_pause()` may be called from any libcurl callback, including this one. Pausing re-enters the event code, which reassesses what the transfer needs. If no transfer then needs the socket, libcurl forgets it and frees its entry. Control returns to `mev_sh_entry_update()`, which still holds the old pointer and writes two fields through it. The fix looks the entry up again instead:

```
lib/multi_ev.c · fix 5ab34cba42

rc = multi->socket_cb(data, s, comboaction, multi->socket_userp,
                      entry->user_data);
mev_in_callback(multi, FALSE);
- entry->announced = TRUE;
...
- entry->action = (unsigned int)comboaction;
+ entry = mev_sh_entry_get(&multi->ev.sh_entries, s);
+ if(entry) {
+   DEBUGASSERT(entry->magic == SH_ENTRY_MAGIC);
+   entry->announced = TRUE;
+   entry->action = (unsigned int)comboaction;
+ }
```

IMPACT

Only libcurl applications that use the event-based multi interface and call `curl_easy_pause()` from inside their socket callback are exposed. The curl tool does not use this API. Whether the write happens is decided by the application's own code rather than by a server, and what is written is an internal flag and an event mask, not attacker-supplied data. Depending on the allocator, the stray write may go unnoticed, corrupt a reused allocation or crash the process. The narrow usage pattern and the absence of remote control explain the Low rating.

RECOMMENDATION

Code that invokes an application callback should assume that any libcurl state reachable from that callback may have changed by the time it returns, and should look objects up again rather than reuse pointers taken before the call. Applications that cannot upgrade should not pause from within the socket callback. They can record the request instead and call `curl_easy_pause()` once control is back in their own event loop.

RESOLUTION

curl now fetches the entry again by socket after the callback returns and updates it only if it still exists. Debug builds also gain a magic value in `mev_sh_entry`, checked on use and cleared on free, so stale pointers of this kind trip an assertion. AISLE co-wrote the patch. It shipped in curl 8.21.0, and the advisory lists the long-term support releases 8.14.2, 8.16.1 and 8.20.1 as fixed.

AISLE-CURL-11

LOW

Use-after-free in the HTTP/2 stream-dependency tree after a handle reset

CVE	CVE-2026-10536	Disclosed	curl 8.21.0, 24 Jun 2026
Component	HTTP/2 · stream dependencies	Scope	libcurl
Affected	7.88.0 – 8.20.0	Fixed in	curl 8.21.0 (bfbff7852f)
Weakness	CWE-416 Use After Free	Introduced	71b7e01610 , Dec 2022
Reported	20 May 2026	Report	HackerOne #3751697

DESCRIPTION

With `CURLOPT_STREAM_DEPENDS` or `CURLOPT_STREAM_DEPENDS_E`, an application could make one HTTP/2 transfer depend on another. libcurl kept these relationships as a tree spanning easy handles. Each handle's option block held a pointer to its parent handle and a linked list of nodes pointing to its children, and `Curl_close()` called `data_priority_cleanup()` to unlink a handle from both sides before freeing it.

`curl_easy_reset()` had no such step. It cleared the whole option block at once, erasing the handle's parent pointer and child list while the other handles in the tree kept their pointers to it.

lib/easy.c · `curl_easy_reset()` before the fix

```
/* zero out UserDefined data: */
Curl_freereset(data);
memset(&data->set, 0, sizeof(struct UserDefined));
Curl_init_userdefined(data);
```

The tree was now one-sided. When either handle was freed with `curl_easy_cleanup()`, nothing removed the links pointing at it, and the later cleanup of the other handle followed them into the freed `Curl_easy`, reading and writing its priority fields.

IMPACT

Only applications that use libcurl directly, build an HTTP/2 dependency tree with these rarely used options and then reset one of the handles involved are exposed. The curl tool offers no way to set stream dependencies. The sequence is entirely under the application's control, and a server cannot trigger it. The use-after-free touches a few pointer and flag fields in a freed handle and can corrupt the heap or crash the process. Given how seldom the options are used, and that HTTP/2 stream dependencies are deprecated, curl rates the issue Low.

RECOMMENDATION

Resetting a handle should release every relationship it has with other handles, exactly as closing it does, before its options are cleared. Where cross-handle state of this kind is hard to keep consistent and the feature has little use, removing it is a reasonable alternative. Applications that cannot upgrade should stop using `CURLOPT_STREAM_DEPENDS` and `CURLOPT_STREAM_DEPENDS_E`, or avoid calling `curl_easy_reset()` on any handle that is part of a dependency tree.

RESOLUTION

curl removed HTTP/2 stream-dependency support rather than repair the tree handling. From 8.21.0 both options are accepted for compatibility but do nothing and are documented as deprecated. The parent and child fields and the code that maintained them are gone, and libcurl no longer sends dependency information, although `CURLOPT_STREAM_WEIGHT` still works. RFC 9113 had already deprecated the priority scheme these options relied on. The advisory lists the long-term support releases 8.14.2, 8.16.1 and 8.20.1 as fixed.

AISLE-CURL-12

LOW

Pooled OpenSSL connections outlive their provider library context

CVE	CVE-2026-80229	Disclosed	curl 8.22.0, 2 Sep 2026
Component	TLS · OpenSSL 3 providers	Scope	libcurl and curl tool
Affected	8.14.0 – 8.21.0	Fixed in	curl 8.22.0 (7ea37abc6a)
Weakness	CWE-416 Use After Free	Introduced	f2ce6c46b9 , Apr 2025
Reported	24 Aug 2026	Report	HackerOne #3969255

DESCRIPTION

libcurl loads an OpenSSL 3 provider when the application names one with `CURLOPT_SSLENGINE`, or implicitly when a client key or certificate is given as a PKCS#11 URI. To do so it creates a private `OSSL_LIB_CTX` library context, stores it in the easy handle's state and builds the connection's `SSL_CTX` from it with `SSL_CTX_new_ex()`. OpenSSL does not take a reference on the context it is given, and libcurl kept none on the connection's behalf.

Connections do not belong to easy handles, though. Under the multi interface a finished transfer hands its connection back to a shared pool, where it stays open for another handle to pick up. When the handle that loaded the provider was cleaned up, libcurl freed the library context along with the rest of its provider state. The pooled connection kept its `SSL_CTX` and TLS session, both still pointing into that context, so the next I/O on the connection, or any post-handshake processing, reached into freed memory inside OpenSSL.

IMPACT

Providers exist only in builds against OpenSSL 3 or later; no OpenSSL fork offers them. Within those builds, the application must load a provider, typically to reach a hardware token or TPM, and must let a connection created under it outlive the handle that created it. That lifecycle is ordinary for multi-interface applications, and the curl tool is affected as well. No attacker action is involved. The result is a use-after-free in OpenSSL's provider code, most plausibly a crash of a long-running process. The narrow build and usage conditions, and a trigger that lies in the application's own handle lifecycle rather than in anything a remote party sends, explain curl's Low rating.

RECOMMENDATION

Anything a pooled connection depends on should live at least as long as the connection. Either the connection should hold its own reference to the library context, or connections built on a per-handle context should never enter the pool. Users who cannot upgrade can follow the advisory and set `CURLOPT_FORBID_REUSE` on transfers that use a provider, so that the connection is closed when the transfer ends instead of being kept for reuse.

RESOLUTION

curl took the second route. When `Curl_oss1_ctx_init()` builds an `SSL_CTX` from a per-handle library context, it now marks the connection for closure, so it is never offered for reuse:

lib/vtls/openssl.c · fix 7ea37abc6a

```
octx->ssl_ctx =
#ifdef OPENSSSL_HAS_PROVIDERS
    data->state.libctx ?
        SSL_CTX_new_ex(data->state.libctx, data->state.propq, req_method):
    #endif
        SSL_CTX_new(req_method);
...
+ #ifdef OPENSSSL_HAS_PROVIDERS
+   if(data->state.libctx)
+       /* forbid connection reuse with provider/engine use */
+       connclose(data->conn);
+ #endif
```

The change shipped in curl 8.22.0 and was backported to the Rock-solid curl long-term support releases 8.20.1, 8.16.1 and 8.14.2.

AISLE-CURL-30

LOW

Failed FTP wildcard listing leaves an internal write callback installed

CVE	CVE-2026-92388	Disclosed	curl 8.23.0, 14 Oct 2026
Component	FTP · wildcard matching	Scope	libcurl
Affected	7.21.0 – 8.22.0	Fixed in	curl 8.23.0 (ba8f8a7f27)
Weakness	CWE-416 Use After Free	Introduced	0825cd80a6 , May 2010
Reported	11 Sep 2026	Report	HackerOne #4022070

DESCRIPTION

With `CURLOPT_WILDCARDMATCH` enabled, libcurl downloads every file in an FTP directory whose name matches a pattern. It first retrieves a `LIST` of the directory and parses the listing itself, then fetches each matching file through the application's normal write callback. To route the listing into its parser, `init_wc_data()` in `lib/ftp.c` overwrote the handle's copies of `CURLOPT_WRITEFUNCTION` and `CURLOPT_WRITEDATA`, keeping the application's values in the per-transfer wildcard state:

lib/ftp.c · before the fix

```
/* backup old write_function */
ftpwc->backup.write_function = data->set.fwrite_func;
/* parsing write function */
data->set.fwrite_func = Curl_ftp_parselist;
/* backup old file descriptor */
ftpwc->backup.file_descriptor = data->set.out;
/* let the writefunc callback know the transfer */
data->set.out = data;
```

The originals were restored only on entering the `CURLWC_MATCHING` state, which follows a successfully received listing. If the `LIST` transfer failed, the wildcard state went to its error path and was freed, taking the saved values with it. The handle was left with `Curl_ftp_parselist()` as its write callback and a pointer to itself as the callback's user data.

IMPACT

Any failed listing leaves the handle in this state: an error reply, a dropped data connection or a timeout is enough. From then on the application's write callback is gone. Data from a later transfer on the same handle is handed to the list parser, which outside a wildcard listing has no state to work with, and the process can crash. A copy made with `curl_easy_duphandle()` inherits the stale user-data pointer, which still refers to the original handle. Once the original is cleaned up, a write through the copy uses a freed easy handle. Wildcard matching is rarely used and the dangerous step depends on how the application treats its handles after a failure, which is why curl rates this Low.

RECOMMENDATION

Fields that hold an application's settings should hold nothing else. An internal consumer of transfer data, such as a listing parser, belongs in the transfer's write path and should be selected only while the listing is in progress, so there is no restore step for an error path to skip. Until they can upgrade, applications using FTP wildcard matching should discard an easy handle after a failed wildcard transfer, or set `CURLOPT_WRITEFUNCTION` and `CURLOPT_WRITEDATA` again before reusing or duplicating it.

RESOLUTION

curl 8.23.0 no longer modifies the application's write-callback settings. The client writer in `lib/cw-out.c` asks a new helper, `Curl_ftp_listmode()`, whether the handle is still receiving a wildcard listing, and sends body data to `Curl_ftp_parselist()` only while it is. The backup fields and both halves of the save-and-restore logic were removed.

AISLE-CURL-33

LOW

Unbounded recursion when freeing deeply nested MIME structures

CVE	CVE-2026-92391	Disclosed	curl 8.23.0, 14 Oct 2026
Component	MIME API	Scope	libcurl
Affected	7.56.0 – 8.22.0	Fixed in	curl 8.23.0 (d7546809d5)
Weakness	CWE-674 Uncontrolled Recursion	Introduced	ce0881edee , Sep 2017
Reported	12 Sep 2026	Report	HackerOne #4024222

DESCRIPTION

The MIME API builds multipart bodies as trees. A part can carry a complete MIME structure of its own, attached with `curl_mime_subparts()`, and the parts of that structure can do the same, to any depth. `curl_mime_free()` released such a tree by recursion, though indirectly. For each part it called `Curl_mime_cleanpart()`, which ran the part's content destructor. For a part holding subparts, that destructor was `mime_subparts_free()`, and it called `curl_mime_free()` on the nested structure.

Every level of nesting therefore added several frames to the call stack, and nothing bounded how many levels there could be. A deep enough chain of nested parts exhausted the stack before the free could finish, and the process crashed.

IMPACT

Only applications that build MIME trees are affected, and only when the nesting becomes very deep; ordinary multipart bodies nest one or two levels. Reaching a dangerous depth takes an application that mirrors externally supplied structure into MIME parts without a limit, for instance when re-encoding nested messages it has received. The outcome is a stack overflow that ends the process, a denial of service. Because the depth is chosen by the application's own code and such designs are uncommon, curl rates the issue Low.

RECOMMENDATION

Tearing down a structure whose depth is set by the caller, or influenced by outside input, should not consume stack in proportion to that depth. An iterative walk with an explicit work list keeps stack use constant. Applications that build MIME trees from external input should cap the nesting depth themselves, which also protects them on older libcurl releases.

RESOLUTION

curl 8.23.0 rewrote `curl_mime_free()` as a single loop. When it meets a part that owns subparts, it detaches the nested structure and pushes it onto a stack of structures still to be freed, threading that

stack through their parent pointers. When the current structure has no parts left, it is freed and the loop moves to the next one on the stack:

```
lib/mime.c · fix d7546809d5

+     if(part->kind == MIMEKIND_MULTIPART && part->arg &&
+         part->freefunc == mime_subparts_free) {
+         curl_mime *subparts = (curl_mime *)part->arg;
+         part->freefunc = NULL;
+         cleanup_part_content(part);
+         subparts->parent = mime->parent;
+         mime->parent = (curl_mimepart *)subparts;
+     }
+     ...
+     curl_mime *parent = (curl_mime *)mime->parent;
+     curlx_free(mime);
+     mime = parent;
```

The fix adds test 3595, which builds and frees a 50-level nested structure.

11 Protocol Handling

Findings in curl's handling of the IMAP, FTP, POP3, SMTP and TFTP protocols.

AISLE-CURL-27

LOW

IMAP APPEND with CRLF conversion lets message data escape as commands

CVE	CVE-2026-87864	Disclosed	curl 8.23.0, 14 Oct 2026
Component	IMAP · APPEND	Scope	libcurl and curl tool
Affected	7.30.0 – 8.21.0	Fixed in	curl 8.22.0 (3b547de301)
Weakness	CWE-130 Improper Handling of Length Parameter Inconsistency	Introduced	f8c8c76d39 , Mar 2013
Reported	26 Aug 2026	Report	HackerOne #3972403

DESCRIPTION

IMAP's APPEND command uploads a message as a literal. The client sends the command with the message length in braces, and the server then reads exactly that many bytes as message data before it expects the next command. The announced length must therefore match the bytes that follow.

libcurl announced the upload size it knew before sending, taken from `CURLOPT_INFILESIZE` or from the size of a MIME body. It did not allow for CRLF conversion. With `CURLOPT_CRLF` set, or the curl tool's `--crlf` option, the upload passes through a reader that turns every bare line feed into a CR LF pair, so more bytes reach the wire than were announced. The server stops treating the upload as message data at the announced length and reads the remainder as IMAP commands. libcurl already refused an APPEND of unknown size, but that check looked only at the size before conversion.

IMPACT

Applications are affected when they append messages over IMAP with CRLF conversion enabled and the message content is not entirely their own, for example when storing mail or files that originate elsewhere. Someone who can shape that content can place IMAP commands beyond the end of the announced literal. The server executes them in the client's authenticated session, where they can delete, copy or flag messages and otherwise act on the user's mailboxes. Conversion is off by default and IMAP messages are normally in CRLF form already, so the combination is uncommon. Together with the need to influence the uploaded data, that explains curl's Low rating.

RECOMMENDATION

The length announced for a literal must be computed after every transformation applied to the upload. Where a transformation makes the final size impossible to know in advance, the command should be refused rather than sent with a guess. Users who cannot upgrade should not combine `CURLOPT_CRLF` or `--crlf` with IMAP APPEND, and should convert line endings themselves before handing the message to libcurl, so that the announced size is exact.

RESOLUTION

The fix shipped in curl 8.22.0, while the advisory followed with 8.23.0, as it did for AISLE-CURL-18 and AISLE-CURL-20. `imap_perform_append()` now takes the size from the whole reader chain through `Curl_creader_total_length()`. The CRLF converter reports its output length as unknown, so any APPEND with conversion enabled is refused with a dedicated error:

```

lib/imap.c · fix 3b547de301

- /* Check we know the size of the upload */
+ /* Check we know the size of the upload. This takes all readers
+  * into account. Especially crlf conversions which make the size
+  * unpredictable, e.g. -1. */
+ data->state.infilesize = Curl_creader_total_length(data);
+ if(data->state.infilesize < 0) {
-     failf(data, "Cannot APPEND with unknown input file size");
+     if(data->set.crlf)
+         failf(data, "Cannot APPEND with CRLF conversion making size unknown");
+     else
+         failf(data, "Cannot APPEND with unknown input file size");
+     return CURLE_UPLOAD_FAILED;

```

AISLE-CURL-29

LOW

Endless control responses defeat timeouts and stall the multi handle

CVE	CVE-2026-92387	Disclosed	curl 8.23.0, 14 Oct 2026
Component	FTP, IMAP, POP3, SMTP · response reader	Scope	libcurl
Affected	8.10.0 – 8.22.0	Fixed in	curl 8.23.0 (753735af87)
Weakness	CWE-835 Loop with Unreachable Exit Condition	Introduced	a5598b6fcb, Aug 2024
Reported	11 Sep 2026	Report	HackerOne #4022003

DESCRIPTION

libcurl’s FTP, IMAP, POP3 and SMTP code shares one reader for control-connection replies, Curl_pp_readresp() in lib/pingpong.c. Since curl 8.10.0 it drains its input in a loop instead of reading once per call, because data held in TLS buffers might otherwise never trigger another socket event. Each pass reads up to 900 bytes, feeds every complete line to the protocol’s end-of-reply test, and goes round again whenever the read filled the buffer. The loop ends when a final reply line arrives, when a read comes up short, or when no data is ready.

A server can make sure none of those happens. If it sends an unbroken stream of complete lines that never form the end of a reply, such as FTP continuation lines, and keeps every read full, the function does not return. libcurl checks timeouts and services other transfers only between such calls, so none of that takes place while the loop runs.

IMPACT

A libcurl application talking to a hostile server over any of these protocols, or over a plain-text control connection an attacker can inject into, can be held inside a single call for as long as the stream continues. Configured limits such as CURLOPT_TIMEOUT never fire. With the multi interface, every other transfer on the same multi handle stalls too, because they all run on the calling thread. The attacker must keep sending data to sustain the stall, and nothing is disclosed or corrupted. The effect is denial of service against the application, which curl rates Low.

RECOMMENDATION

A loop that drains buffered input should still bound the work it does per call and hand control back once the bound is reached, leaving the rest of the data for the next invocation. Until they can upgrade, applications that talk to untrusted FTP, IMAP, POP3 or SMTP servers should keep those transfers on their own thread or process, so a stalled control connection cannot hold up unrelated work.

RESOLUTION

curl 8.23.0 limits the loop to ten reads per call. When the limit is reached, `Curl_pp_readresp()` returns without error and leaves the reply pending, and the state machine resumes reading on a later pass, after timeouts and other transfers have had their turn:

```
lib/pingpong.c · fix 753735af87

do {
...
    if(!pp->overflow) {
+       if(!maxloops--) /* don't get stuck */
+       return CURLE_OK;
+
        result = pingpong_read(data, sockindex, buffer, sizeof(buffer),
                                &gotbytes);
...
    } while(gotbytes == sizeof(buffer));
```

AISLE-CURL-31

LOW

TFTP server can turn a download into an upload of local data

CVE	CVE-2026-92389	Disclosed	curl 8.23.0, 14 Oct 2026
Component	TFTP	Scope	libcurl
Affected	7.15.0 – 8.22.0	Fixed in	curl 8.23.0 (63dd390fab)
Weakness	CWE-200 Exposure of Sensitive Information to an Unauthorized Actor	Introduced	56d9624b56 , Sep 2005
Reported	11 Sep 2026	Report	HackerOne #4022568

DESCRIPTION

A TFTP transfer runs in one direction only. The client sends either a Write Request (WRQ), which the server acknowledges with an ACK before the client starts sending DATA blocks, or a Read Request (RRQ), which the server answers directly with DATA. libcurl's TFTP state machine let the server's first reply decide which of the two was happening. In `tftp_send_first()`, an ACK led into the transmit path and a DATA packet into the receive path, regardless of the request libcurl had actually sent, and nothing later in the transfer checked the direction again. The fix adds the missing checks:

```
lib/tftp.c · fix 63dd390fab

case TFTP_EVENT_ACK: /* Connected for transmit */
+   if(!data->state.upload)
+   return CURLE_TFTP_ILLEGAL;
    result = tftp_connect_for_tx(state, event);
    break;

case TFTP_EVENT_DATA: /* Connected for receive */
+   if(data->state.upload)
+   return CURLE_TFTP_ILLEGAL;
    result = tftp_connect_for_rx(state, event);
```

A server that answers a download request with an ACK turns the download into an upload. libcurl then fills each outgoing DATA block from the handle's read callback. On a handle never set up for uploading, that is either a callback left over from earlier use or libcurl's default, which reads from standard input. In the opposite case, a server that answers an upload with DATA has its blocks passed to the application's write callback as if they had been downloaded.

IMPACT

The disclosure affects applications that use libcurl for TFTP downloads. A malicious TFTP server, or anyone able to spoof one on the unauthenticated UDP path, can make libcurl send it whatever the read callback returns. Depending on how the application configured the handle, that may be data from an earlier upload on a reused handle or the process's standard input. The reverse case lets a server feed unexpected content to an upload's write callback but reveals nothing. curl rates this Low: TFTP is rarely used outside closed networks, the attacker must be or impersonate the server, and what leaks depends entirely on the state of the read callback.

RECOMMENDATION

A TFTP client should fix the direction of a transfer when it sends the RRQ or WRQ, and reject any packet that contradicts it, starting with the first reply. It should never infer its role from what the server sends. Applications that cannot upgrade can give download handles a read callback that returns `CURL_READFUNC_ABORT`, and should reset `CURLOPT_READFUNCTION` and `CURLOPT_READDATA` when reusing a handle after an upload. Because TFTP authenticates neither side, it should carry sensitive data only where the server is trusted.

RESOLUTION

curl 8.23.0 checks the requested direction at each of the three places where libcurl previously acted on the server's packet type: the first reply in `tftp_send_first()`, the transmit handler `tftp_tx()`, and DATA delivery in `tftp_receive_packet()`. A packet that contradicts the direction now ends the transfer with `CURLE_TFTP_ILLEGAL`. The change is confined to `lib/tftp.c`.

12 Command-Line Tools

Findings that affect only the `curl` command-line tool or the `wcurl` script, not `libcurl`.

AISLE-CURL-02

MEDIUM

wcurl can write outside the working directory via percent-encoded slashes

CVE	CVE-2025-11563	Disclosed	curl 8.18.0, 7 Jan 2026
Component	wcurl	Scope	wcurl
Affected	wcurl 2024.12.08 – 2025.11.04	Fixed in	wcurl 2025.11.09 (65546bae01)
Weakness	CWE-35 Path Traversal	Introduced	e01d578582 , Dec 2024
Reported	6 Oct 2025	Report	Not filed on HackerOne

DESCRIPTION

wcurl, the download wrapper script distributed with curl, names the local file after the last path segment of the URL. By default it also percent-decodes that segment, so that a URL ending in an encoded non-ASCII name produces a readable filename. Because wcurl cuts the segment at the last literal slash, the raw name cannot contain a directory separator. Decoding undid that guarantee. An encoded slash (%2F) or backslash (%5C) was decoded like any other character, and the result reached curl through `--output` as a real separator. The decoder already left encoded control characters alone, but nothing excluded the separators.

A final segment that looked like an ordinary filename in the URL could therefore decode into a relative path that climbs out of the current directory, or into an absolute path, and curl would write the download there.

IMPACT

Anyone who runs wcurl on a URL supplied by someone else can have the file written outside the directory they ran it in. Nothing on the command line hints at this, because wcurl chose the name. On curl 7.83.0 and later, wcurl passes `--no-clobber`, so existing files are never overwritten; the attacker can instead create a new file at a path of their choosing, within the user’s permissions, where another program may later read it. With older curl versions, existing files can be replaced as well. curl rates this Medium because no special configuration is needed, only an untrusted URL.

RECOMMENDATION

Decoding a filename must never add path structure to it. A tool that derives a local name from a URL should leave encoded separators undecoded, or check the decoded name and refuse it if it contains a separator. Users who cannot upgrade can pass `--no-decode-filename`, or name the output explicitly with `-o`, `-O` or `--output`.

RESOLUTION

The first fix, in wcurl 2025.11.04, listed the encodings that must stay encoded but left out each entry’s percent sign. `is_safe_percent_encode()` receives the whole three-character sequence, so the comparison below never matched:

```
wcurl - wcurl 2025.11.04, before the complete fix
```

```
readonly UNSAFE_PERCENT_ENCODE="2F 5C"
...
upper_str=$(printf "%s" "${1}" | tr "[:lower:]" "[:upper:]")
for unsafe in ${UNSAFE_PERCENT_ENCODE}; do
    if [ "${unsafe}" = "${upper_str}" ]; then
        return 1
    ...
        && is_safe_percent_encode "${decode_out}"; then
```

wcurl 2025.11.09 added the missing % and tightened a test that had passed despite the bug. curl 8.17.0 bundled the incomplete release; the wcurl in curl 8.18.0 has the complete fix.

AISLE-CURL-32

LOW

The -J no-clobber guarantee can be bypassed with 103 Early Hints

CVE	CVE-2026-92390	Disclosed	curl 8.23.0, 14 Oct 2026
Component	curl tool · --remote-header-name	Scope	curl tool
Affected	8.11.0 – 8.22.0	Fixed in	curl 8.23.0 (c24a69d8cd)
Weakness	CWE-73 External Control of File Name or Path	Introduced	b0c82239c2 , Sep 2024
Reported	12 Sep 2026	Report	HackerOne #4023192

DESCRIPTION

With `--remote-header-name (-J)`, the curl tool names the output file after the server's Content-Disposition header and, by default, refuses to overwrite an existing file of that name. This protection depends on the file not being opened before that header has been seen. CVE-2020-8177 showed what happens when `--show-headers (-i)` breaks that ordering. curl first rejected the combination outright, and in 8.11.0 began supporting it by holding header lines in memory until the filename was settled.

That buffering sat inside the Content-Disposition handling, and two kinds of header line never reached it. The header callback parses only 2xx and 3xx responses for Content-Disposition, so the lines of a 103 Early Hints interim response went straight to the `-i` output code. An ETag line consumed by `--etag-save` was diverted in the same way. Either line opened the output file on the spot, under the name derived from the URL and with the truncating semantics of plain `-O`. The fix inserts a check in front of that open:

```
src/tool_cb_hdr.c · fix c24a69d8cd
```

```
+ if(hdrcbdata->honor_cd_filename && !cd_checked &&
+   (scheme == proto_http || scheme == proto_https))
+   return buffer_header(hdrcbdata, str, cb);
+
+ if(!outs->stream && !tool_create_output_file(outs, per->config))
+   return CURL_WRITEFUNC_ERROR;
```

IMPACT

Only the curl tool is affected, and only when `-J` and `-i` are used together against an HTTP(S) server that sends a 103 response first, or when `--etag-save` is added and the response carries an ETag. A malicious server can then make curl truncate an existing file in the output directory whose name matches the URL-derived name, and write response headers into it, before the `-J` logic that would have refused to touch an existing file has run. The damage is limited to that one name in that one directory. curl rates this Low given the narrow option combination and the attacker's limited choice of target.

RECOMMENDATION

Every decision that depends on response headers should be complete before the output file is opened. Rather than buffering in selected code paths, the tool should hold back all header lines until the filename is final, including those of informational responses and headers consumed by other options. Until they upgrade, users should avoid combining `-J` with `-i` for untrusted servers, or name the output explicitly with `-o`.

RESOLUTION

curl 8.23.0 moves the buffering into a helper, `buffer_header()`, and calls it from the header output path of `tool_header_cb()` whenever `-J` is active over HTTP(S) and the Content-Disposition logic has not already handled the line. No header line can now open the file early. New tests 3037 and 3038 cover the 103 and `--etag-save` cases.

13 Conclusion

Over twelve months, AISLE reported 35 vulnerabilities that the curl project accepted and assigned CVEs. All 35 have been fixed and published in curl security advisories.

None of the findings was rated High or Critical, which reflects the care that goes into curl and the conservative way the project rates severity. At the same time, many of these flaws had been present for years, some for decades, in one of the most heavily reviewed, tested and fuzzed codebases in open source. That is a strong sign that autonomous source code analysis can find real, previously unknown vulnerabilities even in mature software, and that its findings are worth the effort of careful validation and coordinated disclosure.

AISLE thanks the curl security team for fast, thorough and good-humoured triage throughout. Daniel Stenberg, Stefan Eissing and Viktor Szakats wrote and reviewed the fixes, and James Fuller helped assess many of the reports. Thanks also to Filippo Tedeschi for a follow-up fix, and to wcurl's maintainers Samuel Henrique and Sergio Durigan Junior, together with Xi Ruoyao, for fixing the wcurl finding.

A Finding Index

Cross-reference of every finding with its CVE, HackerOne report, dates and fix commit.

ID	CVE	HackerOne	Reported	Disclosed	Fix commit
01	CVE-2025-10966	#3355218	23 Sep 2025	curl 8.17.0, 5 Nov 2025	b011e3fcfb
02	CVE-2025-11563	—	6 Oct 2025	curl 8.18.0, 7 Jan 2026	65546bae01
03	CVE-2025-13034	—	9 Nov 2025	curl 8.18.0, 7 Jan 2026	3d91ca8cdb
04	CVE-2025-14017	—	1 Dec 2025	curl 8.18.0, 7 Jan 2026	39d1976b7f
05	CVE-2025-14819	—	16 Dec 2025	curl 8.18.0, 7 Jan 2026	cd046f6c93
06	CVE-2026-8925	#3735193	14 May 2026	curl 8.21.0, 24 Jun 2026	3da249e1f0
07	CVE-2026-8926	#3735184	14 May 2026	curl 8.21.0, 24 Jun 2026	4ae1d7cc26
08	CVE-2026-8932	#3733910	13 May 2026	curl 8.21.0, 24 Jun 2026	7541ae569d
09	CVE-2026-9080	#3749204	19 May 2026	curl 8.21.0, 24 Jun 2026	5ab34cba42
10	CVE-2026-9547	#3751712	20 May 2026	curl 8.21.0, 24 Jun 2026	0b8dbbc63c
11	CVE-2026-10536	#3751697	20 May 2026	curl 8.21.0, 24 Jun 2026	bfbff7852f
12	CVE-2026-80229	#3969255	24 Aug 2026	curl 8.22.0, 2 Sep 2026	7ea37abc6a
13	CVE-2026-80230	#3969300	24 Aug 2026	curl 8.22.0, 2 Sep 2026	5267ed859d
14	CVE-2026-80231	#3969368	24 Aug 2026	curl 8.22.0, 2 Sep 2026	7be1e70cb6
15	CVE-2026-80255	#3972395	26 Aug 2026	curl 8.22.0, 2 Sep 2026	4f6aa41a01
16	CVE-2026-82208	#3973090	27 Aug 2026	curl 8.22.0, 2 Sep 2026	ed0338befd
17	CVE-2026-82209	#3972385	26 Aug 2026	curl 8.22.0, 2 Sep 2026	95c1e8915d
18	CVE-2026-86409	#3973135	27 Aug 2026	curl 8.23.0, 14 Oct 2026	c3c70440e8
19	CVE-2026-86410	#3973176	27 Aug 2026	curl 8.23.0, 14 Oct 2026	6ae9befb2f
20	CVE-2026-86411	#3973190	27 Aug 2026	curl 8.23.0, 14 Oct 2026	72352daba0
21	CVE-2026-86412	#3973201	27 Aug 2026	curl 8.23.0, 14 Oct 2026	6ded3e5944
22	CVE-2026-86413	#3993549	3 Sep 2026	curl 8.23.0, 14 Oct 2026	43029e2ec3
23	CVE-2026-86414	#3993566	3 Sep 2026	curl 8.23.0, 14 Oct 2026	2956af6c74
24	CVE-2026-86415	#3993870	3 Sep 2026	curl 8.23.0, 14 Oct 2026	00271cfcb9
25	CVE-2026-87862	#3972357	26 Aug 2026	curl 8.23.0, 14 Oct 2026	a1614c3bcf
26	CVE-2026-87863	#3972373	26 Aug 2026	curl 8.23.0, 14 Oct 2026	08d50a791b
27	CVE-2026-87864	#3972403	26 Aug 2026	curl 8.23.0, 14 Oct 2026	3b547de301
28	CVE-2026-92386	#4021979	11 Sep 2026	curl 8.23.0, 14 Oct 2026	445aa87e13
29	CVE-2026-92387	#4022003	11 Sep 2026	curl 8.23.0, 14 Oct 2026	753735af87
30	CVE-2026-92388	#4022070	11 Sep 2026	curl 8.23.0, 14 Oct 2026	ba8f8a7f27
31	CVE-2026-92389	#4022568	11 Sep 2026	curl 8.23.0, 14 Oct 2026	63dd390fab
32	CVE-2026-92390	#4023192	12 Sep 2026	curl 8.23.0, 14 Oct 2026	c24a69d8cd
33	CVE-2026-92391	#4024222	12 Sep 2026	curl 8.23.0, 14 Oct 2026	d7546809d5
34	CVE-2026-94265	#4040551	18 Sep 2026	curl 8.23.0, 14 Oct 2026	557a52f21e
35	CVE-2026-94266	#4041053	18 Sep 2026	curl 8.23.0, 14 Oct 2026	7a6bd027d0