§ ELEVEN IDENTIFIERS · TWELVE ROWS · EIGHT PIECES OF SOFTWARE

The vulnerabilities, and where they remain reachable

Supporting document 1 for *"TLS 1.3 never tells the server what a resumed connection must re-check"*, IAB Workshop on Accelerating the Deployment of Post-Quantum Authentication. León Acosta, QuantaKrypto. v1, 28 August 2026.
What the class is, the twelve instances and how they were counted, who benefits from each, the measurements showing the mechanism is live, and what is not established.

1. Summary

A resumed TLS session carries an authentication verdict forward and replays it without re-establishing the conditions that produced it. Eleven published identifiers span September 2014 to February 2026, eleven years and five months, across eight pieces of software. Only one is literally a revocation bypass, so the class is best described as a replayed verdict rather than a revocation failure.

The specification names the relevant quantity and recommends a bound. It does not require one, does not define one, and provides nothing on the wire. Implementations have consequently each built their own answer. Four parties do not resume under client authentication at all: Caddy and Cloudflare changed behaviour in response to disclosure; AWS Application Load Balancer is documented that way by AWS, which is why Hebrok et al. excluded it from measurement; and Google Cloud Load Balancer was measured that way by those authors, who "found that session tickets are disabled when enabling client authentication".

2. The class, the table, and the counting rules

Inclusion test, applied strictly: an implementation carried an authentication verdict forward on resumption and failed to re-establish a condition that produced it.

DATE	IDENTIFIER	SOFTWARE	CONDITION NOT RE-ESTABLISHED	WHO GAINS
Sep 2014	CVE-2014-3616	nginx	session resumed in a server{} block other than the one that produced the verdict	needs position
May 2015	CVE-2015-3644	stunnel	a failed client-certificate verdict, routed on reuse as if it had succeeded	outside party
Aug 2016	CVE-2016-5419	curl	client certificate identity on a reused connection	the holder itself
Apr 2017	CVE-2017-7468	curl	the same bug: the fix was lost in 7.52.0 and restored here	the holder itself
Jun 2020	CVE-2020-8172	Node.js	server certificate validation, cached before verifying	needs position
Mar 2021	CVE-2021-22890	curl	resumed the origin session with a ticket issued by the HTTPS proxy	needs position
Feb 2022	CVE-2022-21654	Envoy	client-certificate validation settings, omitted from the session id context	outside party
Jan 2024	CVE-2024-0853	curl	REVOCATION: cached although OCSP verification failed	needs position
Feb 2025	CVE-2025-23419	nginx	client certificate scope across virtual hosts	outside party

Feb 2025	CVE-2025-23419	Cloudflare	client certificate scope across zones (same identifier)	outside party
Jul 2025	CVE-2025-23048	Apache httpd	per-virtual-host trust store for client auth	outside party
Feb 2026	CVE-2025-68121	Go crypto/tls	ClientCAs mutated between handshakes while ticket keys are shared	outside party

2.1 Counts, all derived from the table above

- **11** distinct CVE identifiers. **All claims are stated over identifiers**, the smaller and therefore safe base.
- **12** table rows, because nginx and Cloudflare are separate deployments sharing CVE-2025-23419. A reader who counts rows finds more than is claimed, never fewer.
- **8** distinct pieces of software; curl appears four times
- **5 of 11** are the ones where a party outside the current trust scope gains standing access needing no network position, and in all five the replayed verdict is a client-certificate verdict. A wider "client-certificate or scoping" count was dropped: it conflated opposite threat models, and a headline number that needs a disclaimer to survive a recount does not belong in a document whose whole claim is that its numbers survive recounting
- **1 of 11** is literally a revocation bypass, CVE-2024-0853
- span **Sep 2014 to Feb 2026**, which is 137 months, eleven years and five months
- **2** further affected products carry no identifier: Caddy and Fastly. **DDoS-Guard** is a third, inferred rather than confirmed, from ten ASes operated by different companies that all used its service. LiteSpeed is excluded entirely, see 2.2.

2.2 Deliberately excluded, with reasons

Cases excluded because they do not meet the inclusion test. A headline number has to survive someone counting the table beneath it, and an inclusion rule has to survive someone testing it against near misses.

- **Triple handshake, 3SHAKE, and CVE-2014-1295 (2014).** A protocol-level flaw in the binding between a session and a certificate, fixed by extended master secret, RFC 7627. Not an implementation forgetting a re-check. The table opens in September 2014 on nginx CVE-2014-3616, a different 2014 finding by the same researcher which does meet the test; the triple handshake is excluded on its own merits, not to move the start date.
- **GnuTLS CVE-2020-13777 (2020).** An all-zero session ticket encryption key, which is a key generation defect. Not a skipped re-check.
- **(Open)LiteSpeed (2025).** Named as affected by Hebrok et al., but the paper reports the bypass "is not session resumption specific": changing the Host header on a normal connection was sufficient. LiteSpeed did ship a fix and no identifier was assigned, but what it fixed is that Host-header flaw rather than a resumption failure, so it is neither a table row nor an uncounted instance of this class. OpenLiteSpeed does not support client authentication at all.
- **Envoy CVE-2020-8664 (2020).** The static half of an SDS combined validation context was silently dropped. Real, and in the same subsystem as the entry now in the table, but the check never ran with the right settings in the first place: no verdict is carried across a resumption. Trust scoping, not this class.
- **Envoy CVE-2022-21656 and CVE-2022-21657 (2022).** Siblings of the included entry, shipped in the same February 2022 release: a subjectAltName type confusion and a missing extendedKeyUsage restriction. Both are handshake-time validation-logic bypasses, with no cached verdict and no resumption.
- **Istio CVE-2020-11767.** An authentication verdict carried forward across HTTP/2 connection *coalescing* rather than session resumption. A genuine sibling class, and arguably deserving its own table, but not this one.

Why CVE-2022-21654 is in and those are out. The test asks whether a verdict was carried forward and a producing condition left un-re-established. In a resumption design there are only two ways to be correct: re-run the check, or prove its preconditions still hold. The session id context is the mechanism TLS provides for that proof, and Envoy's was incomplete for six validation settings. Envoy's own pre-fix comment states the invariant it then failed to meet: "Hash all the settings that affect whether the server will allow/accept the client connection. This ensures that the client is always validated against the correct settings, even if session resumption across different listeners is enabled." The fix's added contract is blunter still, requiring that the hash cover all validation configuration "so that if this configuration is changed, sessions cannot be re-

used and must be re-negotiated and re-validated using the new settings". That is this paper's ask, written by an implementer in 2022. No formulation of the inclusion test admits nginx's cross-virtual-host carryover and rejects this.

Also excluded and recorded so nobody re-adds them: Tomcat CVE-2026-24734 (OCSP response validation), Traefik CVE-2026-32305 (SNI sniffing), curl CVE-2026-4873 (cleartext connection pool reuse, same family but not a certificate check), curl CVE-2024-2466 and CVE-2024-2379 (certificate check bypasses unrelated to resumption), Go CVE-2026-42505 (Encrypted Client Hello privacy leak), Erlang/OTP CVE-2026-42791 and Botan (OCSP responder validity).

2.3 Provenance for CVE-2014-3616, CVE-2015-3644 and CVE-2021-22890

CVE-2014-3616, nginx, session reuse across `server{}` blocks. Advisory, Maxim Dounin, 16 September 2014, verbatim: "A problem with SSL session cache in nginx was identified by Antoine Delignat-Lavaud. It was possible to reuse cached SSL sessions in unrelated contexts, allowing virtual host confusion attacks in some configurations by an attacker in a privileged network position (CVE-2014-3616). The problem affects nginx 0.5.6 - 1.7.4 if the same shared ssl_session_cache and/or ssl_session_ticket_key are used for multiple server{} blocks." Fixed in 1.7.5 and 1.6.2; NVD published 8 December 2014, so the row's date is the advisory. Fix commit `6ec0f4d8`, whose message is quoted in 2.4 below. **Who gains:** a party holding a position on the path, per the advisory's own "privileged network position". **Why it passes:** the resumed session applied a server-authentication verdict earned in one server context to another without re-establishing the certificate binding. The strongest objection is that "virtual host confusion" sounds like caching hygiene; the answer is that the carried object is the accepted-identity verdict and the cache is the vehicle.

CVE-2015-3644, stunnel, redirect bypass on reused sessions. Advisory verbatim: "In the affected versions, only the initial connection is redirected to the host(s) specified by "redirect". The subsequent connections established using reused SSL/TLS sessions are always forwarded to the host(s) specified by "connect" as if they were successfully authenticated." Affects 5.00 to 5.13, fixed in 5.14 (25 March 2015); reported by Johan Olofsson, advisory by Michal Trojnara; NVD published 14 May 2015. Triggers only in server mode with `verify = 2` or higher and `redirect` set. The fix stores the redirect decision as ex_data on the `SSL_SESSION` and reads it back on reuse. **Who gains:** an outside party with no network position at all, namely the rejected client itself. **Why it passes, and the one nuance:** what resumption carries is a routing decision *derived from* the verification verdict rather than the verdict itself. That does not weaken it, because the route is a strict function of the verdict, fail to decoy and pass to backend, so carrying the route forward is operationally identical.

CVE-2021-22890, curl, TLS 1.3 session ticket proxy host mix-up. Advisory verbatim: "When using an HTTPS proxy and TLS 1.3, libcurl can confuse session tickets arriving from the HTTPS proxy but work as if they arrived from the remote server and then wrongly "short-cut" the host handshake." Cause, in the vendor's words: in TLS 1.3 tickets arrive post-handshake "and the code was not updated to take that changed behavior into account." Affects 7.63.0 to 7.75.0 on OpenSSL and forks, fixed in 7.76.0; reported by Mingtao Yang, patched by Daniel Stenberg; advisory 31 March 2021, NVD 1 April 2021. Fix commit `b09c8ee1`, "vtls: add 'isproxy' argument to Curl_ssl_get/addsessionid()", threading an `isproxy` flag through every backend so the two layers key separately. **Who gains:** a beneficiary holding a position on the path, since a malicious proxy is by definition already in the middle and must also present a certificate curl accepts. **Why it passes:** a validated-peer verdict from the proxy layer short-cut the origin handshake, leaving the origin certificate check un-re-established.

2.4 nginx twice, eleven years apart, and the honest qualifiers

This is the paper's strongest exhibit, and it is a documented three-commit chain rather than an inferred one. One caution about provenance: the two commit messages quoted below have no archived copy in the evidence set, and hg.nginx.org has since been decommissioned, so verify the exact wording against the current nginx repository before relying on it. Four other quotations are in the same position and are named here rather than left to be discovered: the curl commit messages in section 2.5, the AuthKEM draft quotation in supporting document 3, the Go 1.21 release-note wording, and the Cloudflare blog sentences. Every other quotation in these documents has a copy in the evidence set. The 2014 fix, commit `6ec0f4d8`, verbatim: "SSL: session id context now includes certificate hash. This prevents inappropriate session reuse in unrelated server{} blocks, **while preserving ability to restore sessions on other servers when using TLS Session Tickets.**" Same-certificate resumption was left permitted deliberately. CVE-2025-23419 lives in exactly that reserved space: its precondition is server blocks sharing one certificate, and therefore one session id context, but differing in `ssl_verify_client`. The 2025 fix, commit `46b9f5d3`, records why the earlier guards were not enough: "In OpenSSL, session resumption always happens in the default SSL context, prior to invoking the SNI callback", and under TLS 1.3 "SSL_get_servername() returns values received in the resumption handshake, which may be different from the value in the initial handshake." That commit cites the 2013 SNI restriction by name as insufficient, so the lineage runs 2013, 2014, 2025 in nginx's own commit messages.

Two qualifiers, because the exhibit is easy to overstate. First, the 2014 author closed the server-authentication direction and could not have closed the client-authentication direction with the same mechanism, because same-certificate resumption is a supported feature rather than an oversight. Second, the 2025 exposure also depended on a TLS 1.3 OpenSSL behaviour, resumption before the SNI callback, which did not exist in the TLS 1.2 world of 2014. The gap was left in 2014; what walked through it arrived with TLS 1.3. The claim this dossier makes is therefore narrow: the 2025 bug's precondition is precisely the same-certificate resumption the 2014 fix consciously preserved, and no text anywhere told anyone that the authentication policy also needed binding.

2.5 The curl timeline, since the paper leans on it

The paper's deployment argument cites curl disabling resumption under client certificates for fourteen months. Traced: the disable arrived in **7.50.1, August 2016**, commit `247d890da8`, "TLS: switch off SSL session id when client cert is used", setting `dest->sessionid = FALSE` in `Curl_clone_ssl_config` whenever a client certificate is configured. It regressed in 7.52.0 (CVE-2017-7468) and was restored in 7.54.0. It was **removed in 7.56.0, October 2017**, commit `9d3dde37a8`, whose message states that "reusing connections that are secured with a client certificate is now possible, and the statement 'TLS session resumption is disabled when a client certificate is used' in the old advisory is obsolete". August 2016 to October 2017 is the fourteen months the policy spanned; the 7.52.0 regression interrupted it for about four of them, so it was in force for roughly ten.

What replaced it, for completeness, because the trajectory is the better exhibit: matching on the full TLS configuration from 7.56.0, a credential-gated cache from **8.12.0, February 2025** (commit `fa0ccd9f1f`, which moved the key construction into `lib/vtls/vtls_scache.c`), and the client certificate path and key material hashed into the cache key itself in **8.21.0, June 2026** (commit `7541ae569d`, reported by Joshua Rogers). Ten years of unaided iteration to arrive at scope binding.

3. The 2025 cluster: one reporter, not one cause

`CONFIRMED FROM THE PAPER PDF` S. Hebrok, T. L. Storm, F. M. Cramer, M. Radoy and J. Somorovsky, all Paderborn University, "STEK Sharing is Not Caring: Bypassing TLS Authentication in Web Servers using Session Tickets", 34th USENIX Security Symposium, 13 to 15 August 2025, Seattle.

> *"Our findings reveal that all four implementations – Apache, nginx, (Open)LiteSpeed, and Caddy – were vulnerable to client authentication bypasses. In our large-scale scans, we identified six clusters of vulnerable providers, including Fastly, which were susceptible to server authentication bypasses."*

> *"CVEs were assigned for the findings in Apache (CVE-2025-23048) and nginx (CVE-2025-23419). LiteSpeed has fixed the issue, and Caddy has disabled session tickets when using client authentication. Fastly binds the tickets to the certificate, and Cloudflare, as a preliminary fix, disabled session tickets when using client authentication."*

One source, not one cause. Two identifiers came from this work, CVE-2025-23048 and CVE-2025-23419, and the paper is explicit that the servers failed differently: Apache on an omitted SNI with a default host, Caddy on SNI matching the restricted host, nginx ignoring SNI entirely. Same class and same reporter, not a shared root cause. **LiteSpeed is excluded from section 2 entirely**, because the paper reports its bypass worked "without using tickets": changing the Host header on a normal connection was enough, which makes it not a resumption bug. OpenLiteSpeed does not support client authentication at all. DDoS-Guard is an inference, from ten ASes operated by different companies that all used its service.

On Cloudflare specifically, the paper describes setting up mutual TLS on two distinct domains under two accounts, resuming a ticket issued for the first against the second, and being served the protected content. That was disclosed through Cloudflare's bug bounty programme, and matches Cloudflare's own advisory of 7 February 2025.

One distinction the paper insists on and this dossier preserves. The same paper reports a separate Cloudflare SaaS finding which it states was "unrelated to session resumption and had a broader authentication impact". That finding is not part of this argument and must not be conflated with the mTLS resumption issue.

Fastly was reported earlier and separately: "we reported this to Fastly directly in our pre-scan phase in September 2023".

4. Threat model, and why the exposure is mutual TLS

What a bound does not protect against. If a server's private key is stolen after your first handshake, a chain bound does not help you, and it is worth being precise about why. It is not that your resumed connections still reach the legitimate server: a thief with the key and network position simply declines the PSK and forces a full handshake with the stolen key, and you reach them. The reason is narrower. A resumption bound governs the resumption path, and an attacker holding a stolen key has no need to take that path at all.

What it does protect against is the case where the party you resumed with is the party whose authority is later withdrawn: mis-issuance, compromise already present at first contact, a hosting or CDN relationship that ends, a domain that changes hands.

4.1 The asymmetry that decides the exposure

For a **server** certificate the attacker needs continued network position. When a domain changes hands or a CDN relationship ends, the client resolves DNS and connects to the new operator, who cannot decrypt the old ticket. The former operator benefits only while it still receives the traffic.

For a **client** certificate in mutual TLS the revoked party simply keeps connecting. It holds the resumption secret already and needs no network position whatever.

That asymmetry, and not any argument about browser behaviour, is recorded in the last column of the section 2 table: every identifier where an outside party gains standing access without network position is a client-certificate verdict, five of five.

4.2 Mesh exposure, shown rather than asserted

The mesh population is not OpenSSL. Envoy, the data plane for Istio and Consul, defaults to **BoringSSL**; Linkerd uses **rustls**; Consul's own control plane is **Go**. The measurement in section 5 covers the stack beneath nginx, HAProxy, Apache, curl and Python.

Read this section with supporting document 2, section 2.3, which narrows it: Envoy binds scope and CRL contents into the session id context, so the 2025 scoping class does not reach it, and the residual is temporal.

The mesh case can be sourced rather than asserted. Envoy's `DownstreamTlsContext.disable_stateless_session_resumption` is a plain boolean with no default of true in the proto, so it defaults to false: **stateless resumption is enabled by default on mutual-TLS listeners**. (`disable_stateful_session_resumption` is not evidence here: the proto scopes it to TLS 1.2 and earlier.) BoringSSL stores the client certificate in the session, and per Cloudflare's advisory "upon resuming that session, the client certificate is not revalidated against the full chain of trust, and the original handshake's verification status is respected". BoringSSL's own header says `SSL_CTX_set_reverify_on_resume` is "only respected on clients".

So the one library with a re-verification control has it on the client side, while in mutual TLS the party that must re-check is the server examining a client certificate.

4.3 How much of the internet this touches, measured where it can be

The paper leads on three numbers. All three are other people's measurements, cited rather than reproduced here, and each is bounded differently.

QUANTITY	VALUE	POPULATION AND DATE	SOURCE
Reachable TLS hosts that issue a session ticket	**94.8%** 1,108,322 of 1,169,527	Tranco top 1M expanded to 2,063,760 proposed domains; 1,463,917 open on :443; scanned Dec 2024 to Jan 2025. Ticket support was probed *by sending an HTTP request*, Hebrok et al., section 5.1.1, which is the condition our own corroboration was checked against	Hebrok et al., USENIX Security 2025, Table 3
Connections completed by resumption	**10 to 94%** by vantage; see section 6	Cloudflare's post-quantum experiment traffic, not a host census. Resumption was worth 30 to 50% off handshake time	Cloudflare, *The TLS Post-Quantum Experiment*
All websites terminating TLS at Cloudflare	**24.9%** 84.5% of sites whose reverse proxy is known	All websites, August 2026	W3Techs

What the first two together establish. Resumption is available on almost every host that could offer it, and at Cloudflare's vantage it carries roughly half of real connections. So the per-connection saving the paper computes is not a corner case, though section 6 records how far resumption rates vary by vantage and why they must not be averaged. That is the argument for the mechanism mattering. It is not, by itself, an argument that the mechanism is broken.

What the third establishes, and what it does not. It does not say that a quarter of the internet uses mutual TLS. It says that when Cloudflare disabled resumption under mutual TLS it did so network-wide, at a provider that terminates TLS for a quarter of all websites, so the retreat was executed at provider granularity rather than per customer. The size of the mutual-TLS population itself has never been measured globally, and this paper does not estimate it.

The counterweight, stated in full because it cuts against the argument. The same internet-scale scan sampled 59,752,483 pairs, escalated 7,218,628 for similarity analysis, manually reviewed 4,370 resumptions, and classified **176 as vulnerable**, mostly clustered behind two DDoS-mitigation providers, DDoS-Guard and Variti, with isolated cases in three unrelated ASes. Most remaining hits were a Cloudflare SaaS routing issue the authors themselves classify as unrelated to session resumption. So the *server*-authentication half of this class is rare in the deployed web. The *client*-authentication half, which is where every 2025 identifier and the whole of the exposure argument sits, was never scanned at that scale: it needs credentials at both endpoints, and the authors evaluated three CDN services by hand instead. Its prevalence is therefore unknown, not low, and nothing here should be read as claiming otherwise. Separately, not one of the eleven identifiers appears in CISA's KEV catalogue. The cost being paid today is the retreat, not exploitation.

5. Measurements

5.1 The server side: a resumed session carrying an expired CLIENT certificate

This is the measurement behind the paper's staleness case, and it is the one that matters, because section 4 argues the exposure lives in mutual TLS where the replayed verdict is a client certificate.

Harness: `mtls/gen.sh` and `mtls/run.sh`, OpenSSL 3.6.1 (27 January 2026), macOS arm64, run 26 August 2026, about ninety seconds end to end. Server: `openssl s_server -tls1_3 -Verify 1 -verify_return_error -CAfile ca.pem`. One variable: a client certificate issued valid for 45 seconds, with `keyUsage=digitalSignature` and `extendedKeyUsage=clientAuth`. The server certificate and trust store are constant across all three connections.

#	CLIENT CERTIFICATE	HANDSHAKE	SERVER
1	valid	full (new)	accepted
2	**EXPIRED**	**Reused**	**ACCEPTED**
3	EXPIRED	full (new), control	rejected

Connection 3, verbatim from the server: `depth=0 CN=test-client / verify error:num=10:certificate has expired / notAfter=Aug 26 06:16:52 2026 GMT / error:0A000086:SSL routines:tls_process_client_certificate:certificate verify failed:ssl/statem/statem_srvr.c:3914`.

The strongest evidence is an absence. The server log contains exactly **two** verification sequences for **three** connections. Connection 2 produced no verify callback output at all, and on that connection the client reports "No client certificate CA names sent". The server did not evaluate the expired certificate and decide it was still acceptable. It never asked for one; the identity came out of the session.

A defect in the first run, recorded because it would have invalidated the result. The harness originally used `-Verify 1` alone, without `-verify_return_error`. Under that configuration the CONTROL PASSED: OpenSSL's server callback prints `verify error:num=10:certificate has expired` and then returns 1, so the server accepted the expired certificate on a full handshake too, and the experiment demonstrated nothing. The control row exists precisely to catch that, and did. Anyone reproducing this must use both flags.

What this establishes and what it does not. Establishes: on OpenSSL 3.6.1, a server configured to require and enforce client certificates accepts a resumed session whose client certificate has expired, and rejects the identical certificate on a fresh connection to the same server with the same trust store. Does not establish: anything about revocation as distinct from expiry, about other stacks, or about frequency in the field. One stack, one version, one platform, one run, offered as an existence proof of the mechanism on the server side.

5.2 The client side: OpenSSL, with a control

A local TLS 1.3 server with a certificate valid for 40 seconds, issued by a local CA. An OpenSSL client saves a session with `-sess_out` while the certificate is valid, holding the connection open long enough to receive the TLS 1.3 NewSessionTicket, since the ticket arrives after the handshake and a client that exits immediately saves nothing. After expiry it reconnects twice, once with `-sess_in` and once without.

#	CERTIFICATE	HANDSHAKE	OPENSSL CLIENT VERDICT
1	valid	full (new)	Verify return code: 0 (ok)
2	EXPIRED	**Reused**	**Verify return code: 0 (ok)**
3	EXPIRED	full (new), control	Verify return code: 10 (certificate has expired)

Row 3 is what makes this conclusive. The same client, with the same trust store, presented with the same expired certificate, rejects it on a fresh connection and accepts it on a resumed one. The difference is not the certificate, the trust store, or the policy. It is only which code path the connection took. OpenSSL is the stack beneath nginx, HAProxy, Apache, curl and Python, which is most of section 2's table. For the mesh population specifically, see 4.2: that is BoringSSL and rustls, not OpenSSL.

5.3 Firefox: a second stack, not a claim about browser exposure

Certificate valid 45 seconds, page reloading every 15 seconds with `Connection: close` so each load is a new handshake from the same running process. Connections 1 to 4 occurred while valid; connections 5 to 9, from 59.8s to 120.0s, all resumed with the certificate expired, with no error and no fallback. This matches NSS source: on client-side stateless resumption the cached peer certificate is copied forward with `CERT_DupCertificate` and the `authCertificate` callback is not invoked.

What this result is for. It confirms the code path exists in a second, independent stack. It is deliberately not offered as evidence of browser exposure, which this dossier and the paper both concede is small. Using Firefox as evidence on one page and discounting browsers on another would be inconsistent, so the claim is limited here explicitly.

5.4 Chrome, from source rather than measured

Chromium calls `SSL_CTX_set_reverify_on_resume(ssl_ctx_.get(), 1)`, documented in BoringSSL as reverifying stored certificates when resuming, so an expired certificate fails. The staleness window is `CachingCertVerifier`, clamped to 30 minutes. This catches expiry but not revocation, since Chrome performs no default online leaf revocation check.

Chrome was not measured because doing so would have required adding a CA to the macOS system trust store. The asymmetry in evidence quality is stated rather than hidden.

5.5 Why expiry is the right thing to measure

The paper's concern is revocation and the measurements test expiry. The connection is that short-lived certificates are the industry's chosen replacement for revocation. As lifetimes fall toward 47 days and below, an expiry check on resumption is the revocation check that most of the ecosystem will actually receive.

6. Open items and known weaknesses

- Chrome's behaviour is source-established, not measured, for the reason in 5.4.
- Safari is untested and its behaviour unknown.
- Each measurement is a single run on a single version and platform.
- The Apache "no chain bound" finding is by inspection of mod_ssl, not confirmed by a maintainer.
- **Current vendor behaviour is not re-verified.** The Fastly, Caddy, Cloudflare and Google Cloud Load Balancer entries in supporting document 2, section 2.4, describe what Hebrok et al. reported or measured in 2025. Cloudflare's is described by those authors as a preliminary fix. Before any of this is stated in the present tense, each should be re-checked against current vendor documentation or behaviour.
- **Resumption rates differ by vantage point and must not be averaged.** Holz et al. (arXiv:1907.12762, April 2019 data, ICSI Notary passive measurement) found 9.7% of TLS 1.3 connections carried the PSK extension, with servers accepting 92% of those. Cloudflare reported roughly 40% of HTTPS connections as resumptions in 2017, and in 2025 reported 27% of QUIC connections with at least one HTTP request as resumptions, "avoiding the need to transmit certificates". Chou and Cao (arXiv:2604.24869) measure 94.16% for CDN-terminated TLS 1.3 against 46.09% for non-CDN. The spread is vantage and seven years of drift, not disagreement; the honest statement is that the population is large and its exact size is not established here.
- **Mutual TLS deployment share is not measurable and none is claimed.** The best available figure is volume at one vantage: Dong et al., "Mutual TLS in Practice" (IMC 2024), observed over 1.2 billion mutual TLS connections on one campus network across 23 months, with 2.2M unique server and 3.4M unique client certificates.
- **Stateless versus stateful ticket share cannot be measured from outside**, because RFC 8446 section 4.6.1 makes the ticket "an opaque label" that "MAY be either a database lookup key or a self-encrypted and self-authenticated value". Only defaults are verifiable: OpenSSL documents that "by default OpenSSL will use stateless tickets", nginx enables tickets by default, and BoringSSL and Envoy offer no stateful mode at TLS 1.3.
- **No exploitation is known.** None of the eleven identifiers appears in CISA's Known Exploited Vulnerabilities catalogue as of 26 August 2026, and no incident report or weaponised proof of concept was found. Stated once, without special pleading: a resumed session that should have been refused leaves no distinguishing artifact, so absence of evidence is weaker here than for classes that crash things.

- **Even the cheapest condition has drawn a compatibility complaint.** golang/go#77359 asks Go to stop checking notAfter on resumption when `InsecureSkipVerify` is set, because resumption is now stricter than the initial handshake. Nothing here is free.
- CVE-2025-68121's severity is disputed and **Go itself assigned none**: its CNA record carries no CVSS and GO-2026-4337 has no severity field. NVD rates it 10.0 critical, CISA-ADP 9.1, and GHSA-h355-32pf-p2xm 4.8 moderate. That last is an *unreviewed* GitHub stub with no package listed, so it is the weakest of the three and should not be used to argue the issue is minor. Cite the rater, never "critical" alone.
- HAProxy's series has now been backported to the 3.4, 3.3 and 3.2 branches, but not 3.0, and no stable release contains it: 3.4.3, 3.3.13, 3.2.22 and 3.0.26 all shipped 29 July 2026. The only tag containing it is the 3.5-dev5 snapshot of 21 August. Some of those backports landed within a day of this writing, so this item dates faster than the rest.
- AWS App Mesh is deliberately not cited: it reaches end of support on 30 September 2026.
- The mesh window is a ceiling, not a typical case, for the reason in supporting document 2, section 2.3, which is Istio's certificate rotation swapping the SDS context rather than any library timeout being reached: BoringSSL's self-generated ticket key rotates every two days and dies with the process, so restarts shorten it.
- Browser exposure is small: session caches are in memory only, resumption state is partitioned by site, and measured browser resumption lifetimes are frequently under an hour.
- RFC 9846 is Standards Track and published, July 2026; every quotation from it in this submission was checked against the RFC text rather than a draft.

7. Revision record

Every substantive correction made during review, kept here so the body of this dossier states only what is currently believed. Three corrections stay in the body instead of here, because a reader could otherwise repeat the error: the Merkle Tree Certificates quotation in supporting document 3, which belongs to the individual draft rather than the working-group draft; the excluded-CVE list in 2.2, which exists so nobody re-adds them; and the caution in supporting document 3 against deriving a constructed chain's parameters by subtraction from its own totals, which is how three readers independently reached the wrong answer.

1. **Mesh exposure.** An early draft argued from 4.2 that meshes are exposed as a class. Envoy binds scope and CRL contents into the session id context, so the 2025 scoping class does not reach it; the residual is temporal. Corrected in supporting document 2, section 2.3.
2. **The constructed chain's parameters.** Review argued the intermediate must be ML-DSA-44 by subtraction from the leaf. A revision accepted that and relabelled it. The construction script records ML-DSA-65, and there is no X.509 overhead in the construction to reason about: the 14 bytes that looked like an impossible residue are the length of `CN=example.com`. Corrected in supporting document 3, section 1.1.
3. **Byte accounting.** An early version of supporting document 3, section 1.1b, gave totals without derivations, used 568 bytes of encoding per certificate in one paragraph and 660/500 in the next, and applied a framing allowance to one row and not the others, in the direction that flattered its own congestion-window conclusion. Three figures are withdrawn: 17,171 as "raw material" (it includes 29 bytes of subject strings; raw is 17,142), 13,681 (not reproducible from any stated constant set), and 11,020 (an encoded figure plus a raw signature). The reconciliation is 1,287, not 1,311. The conclusion survived rebuilding and now carries a floor analysis.
4. **The post-quantum cost direction.** An early draft had the incentive argument backwards for mutual TLS. No stack puts the *server's* chain in a wire ticket; Go and BoringSSL serialise the *client's*. Corrected in supporting document 2, section 2.2.
5. **The semantics count.** Counting HAProxy's three library builds and Go's two GOOS branches as separate semantics gave nine. Both are transient. Six is the count of session bindings; the seventh responder, Apache, enforces elsewhere.
6. **wolfSSL.** A draft said HAProxy's commit left wolfSSL with no bound at all. The commit does not say that; it says the clamp is skipped. Corrected in supporting document 2, section 2.2.
7. **Envoy and CRLs.** A draft listed "no CRL" among the things resumption skips while the same paragraph said CRL contents are hashed, which was self-contradictory, and described `ssl_session_is_resumable()` as checking "nothing else" when it also checks the stored client-credential form and QUIC versus TCP. Both corrected in supporting document 2, section 2.3.
8. **The mesh cost model.** A revision attached Istio's 45-second `drain_duration` to CRL publication; that figure is the hot-restart drain and a CRL push drains nothing. It also gave "about 900 handshakes per second fleet-wide" alongside "under one per second per sidecar", which cannot both hold at forty sessions per sidecar, and "40 ms of CPU", which is about four times the measured figure. Corrected in supporting document 2, section 2.5.
9. **The novelty claim.** A revision of the paper claimed the identity-versus-policy point was not in the prior literature. It is: Hebrok et al. propose binding "the rules that the certificate passed initially" and explicitly reject SNI-only binding. Corrected in supporting document 2, section 2.6, which also records that the defective local text extraction searched first returns zero hits for "countermeasure" and would have appeared to confirm the false claim.
10. **The proposal.** A revision of this dossier still carried the previous four-item proposal (define, require with `notAfter` as floor, an age ceiling ranked third, optionally expose the age) after the paper had restructured around parity, and a cross-reference still pointed at it. Supporting document 2, section 4, now mirrors the paper.
11. **Items 3 and 4 contradicted each other.** A digest changes when a status snapshot is replaced, not when it merely expires unreplaced, so item 4 alone let a hard-fail endpoint resume past `nextUpdate`. Item 4 now requires pairing. Corrected in supporting document 2, section 4.
12. **Three table rows.** CVE-2014-3616, CVE-2015-3644 and CVE-2021-22890 were added to the table before their provenance was written; they are traced in 2.3. Apache httpd was placed with the unbounded implementations while the survey table credited it with a condition bound; it is now the seventh responder, enforcing at the HTTP layer.
13. **Quotation accuracy.** RFC 9846 writes "7 days", not "seven days", and RFC 8446 said clients MUST NOT *cache* tickets that long where 9846 says *use*. The claim that the RFC contains no normative text on certificate validity at resumption was softened: there is no required check, but RFC 9846 section 4.7.1 does RECOMMEND lifetime limits.

8. References

This is the **shared bibliography for the whole submission**, reproduced identically in each supporting document so that any one of them can be read on its own. Not every entry is cited in this particular document.

S. Hebrok, T. L. Storm, F. M. Cramer, M. Radoy, J. Somorovsky, "STEK Sharing is Not Caring: Bypassing TLS Authentication in Web Servers using Session Tickets", 34th USENIX Security Symposium, August 2025. https://www.usenix.org/conference/usenixsecurity25/presentation/hebrok

RFC 9846, TLS 1.3, section 4.7.1, July 2026. https://www.rfc-editor.org/rfc/rfc9846.txt

RFC 9190, EAP-TLS 1.3. https://datatracker.ietf.org/doc/rfc9190/

L. Chuat et al., "SoK: Delegation and Revocation, the Missing Links in the Web's Chain of Trust", IEEE EuroS&P 2020. https://arxiv.org/abs/1906.10775

R. Holz et al., "The Era of TLS 1.3", IMC 2019. https://arxiv.org/abs/1907.12762

Cloudflare, "Resolving a Mutual TLS session resumption vulnerability", 7 February 2025. https://blog.cloudflare.com/resolving-a-mutual-tls-session-resumption-vulnerability/

Cloudflare, "The TLS Post-Quantum Experiment". https://blog.cloudflare.com/the-tls-post-quantum-experiment/ (resumption share of connections, and its latency value).

W3Techs, "Usage Statistics and Market Share of Cloudflare", August 2026. https://w3techs.com/technologies/details/cn-cloudflare

BoringSSL include/openssl/ssl.h; Chromium net/socket/ssl_client_socket_impl.cc; NSS lib/ssl/tls13con.c. All retrieved 25 August 2026.

nginx changeset 8086:496241338da5, 12 October 2022, shipped in 1.23.2. HAProxy commit 9f1d590999c2, 18 August 2026.

`Scripts: pdf/rows.py (the table and its derived counts), fftest/server.py, fftest/nonbrowser.sh, fftest/RESULT.md.`