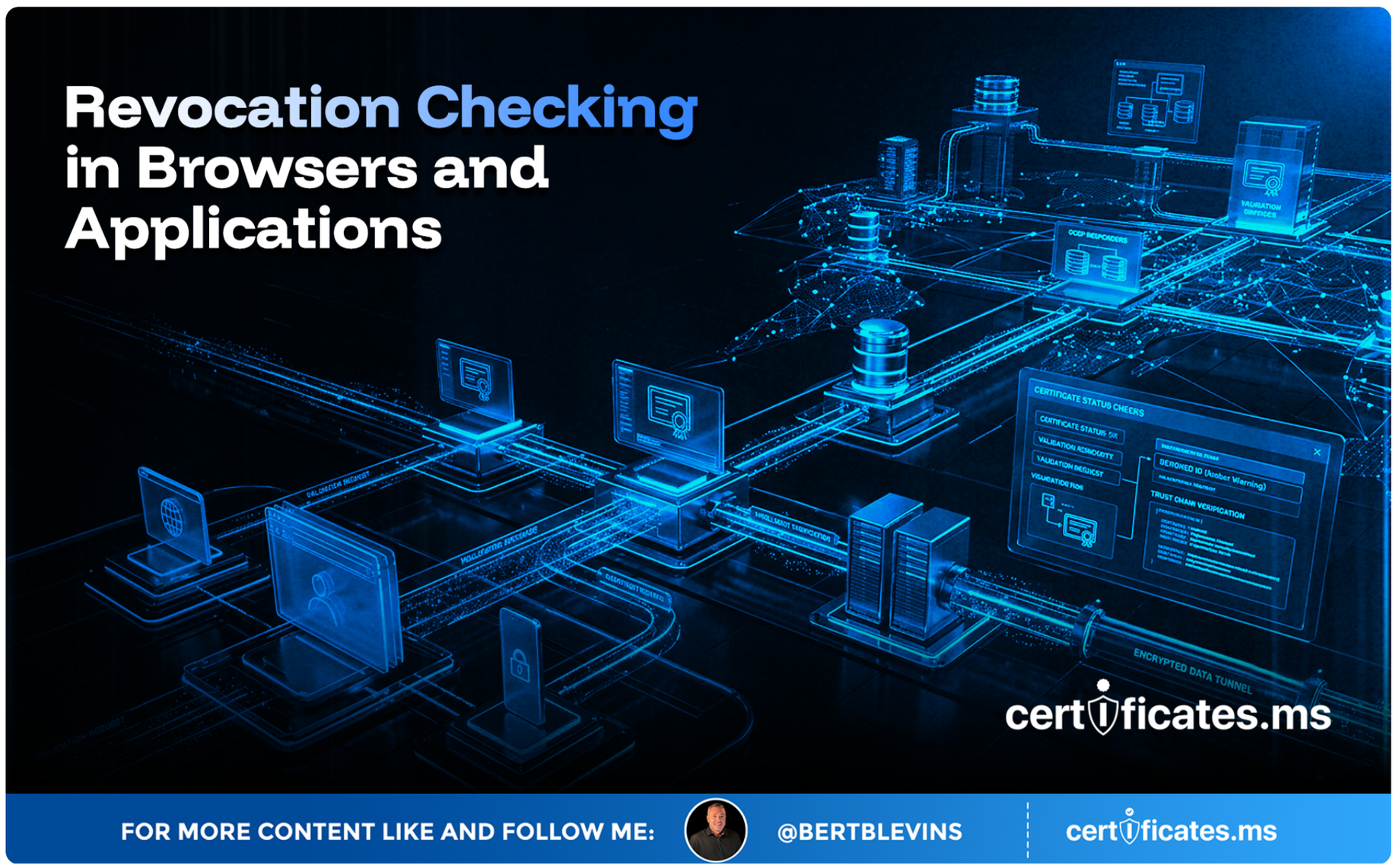
Revocation Checking in Browsers and Applications



Revocation exists to invalidate a certificate before its natural expiration, but a revocation mechanism only matters if clients actually check it. This article looks specifically at how browsers and applications handle revocation checking in practice, where the theory of CRLs and OCSP, covered in depth elsewhere in this series, meets the messier reality of what actually happens during a real connection.

The Uncomfortable Truth About Browser Revocation Checking

For years, major browsers have taken a considerably more relaxed approach to real-time revocation checking than most people assume. Fully checking OCSP or CRL status for every single connection introduces latency and a dependency on external infrastructure being available at the exact moment of connection, and browser vendors have generally concluded that the performance and reliability cost of strict, blocking revocation checks outweighs the security benefit for the average connection, particularly given how often OCSP responders have proven unreliable or slow in practice.

Chrome's Approach: CRLSets

Chrome does not perform live OCSP or CRL checks for the vast majority of certificates it encounters. Instead, it relies on CRLSets, a compact, centrally curated list of the most significant revoked certificates, pushed to Chrome installations through the browser's own update mechanism rather than checked live per connection. This approach trades comprehensive revocation coverage for speed and reliability, covering the highest-impact revocations, such as those from a compromised CA, while accepting that a smaller, individually revoked certificate might not be caught by this mechanism for every user immediately.

Firefox's Approach: CRLite and OneCRL

Firefox uses a related but distinct approach, combining OneCRL, a curated list similar in spirit to Chrome's CRLSets, with CRLite, a more comprehensive compressed data structure designed to represent revocation status for a much larger set of certificates efficiently, distributed to the browser without requiring a live query for every connection. This gives Firefox somewhat broader revocation coverage than the curated-list-only approach, while still avoiding the latency and reliability problems of live per-connection OCSP checks.

Why OCSP Stapling Remains the Best Real-Time Option

For situations where genuinely current, per-certificate revocation status matters, OCSP stapling, covered in more detail elsewhere in this series, remains the most practical mechanism, since it shifts the burden of checking OCSP status onto the server rather than the client, and delivers a signed, time-stamped response as part of the TLS handshake itself. Properly configured OCSP stapling gives clients meaningfully fresher revocation information than relying purely on a browser's periodically updated curated revocation list.

Revocation Checking Outside the Browser

Applications outside the browser context, including custom TLS clients, internal service-to-service communication, and mobile applications, often need to implement their own revocation checking explicitly, since many TLS libraries do not perform revocation checks by default the way browsers attempt to. Developers building anything security-sensitive that consumes TLS certificates directly should explicitly verify what revocation checking behavior their TLS library provides out of the box, rather than assuming equivalent protection to what a modern browser implements, since the defaults vary considerably across languages and libraries.

The Practical Implication of Shrinking Certificate Lifespans

One underappreciated effect of the industry-wide move toward much shorter certificate validity periods is a corresponding reduction in how much any single organization needs to depend on revocation checking working perfectly in the first place. A compromised certificate valid for only a matter of weeks represents a meaningfully smaller window of exposure than one that could otherwise remain valid for a year, somewhat reducing, though certainly not eliminating, the practical cost of imperfect revocation checking across the ecosystem.

Revocation Checking for AI Agents and Automated Clients

AI agents and automated service clients making high volumes of outbound TLS connections face the same revocation-checking tradeoffs as any other application, and given the sheer connection volume involved, the latency cost of strict, blocking revocation checks can be particularly noticeable. Organizations building AI-driven infrastructure should make a deliberate choice about revocation checking behavior for their internal TLS clients, rather than simply inheriting whatever a library's default happens to be, weighing the specific risk profile of the connections an agent is making against the performance cost of stricter checking.

The Countdown Is Already Running: 200 Days, 100 Days, 47 Days

Every certificate conversation in 2026 eventually arrives at the same clock, and it is worth closing on it here. The CA/Browser Forum's Ballot SC-081v3 is not a proposal under discussion; it is an approved, already-in-motion schedule. Maximum public TLS certificate lifetimes fall from 398 days to 200 days on March 15, 2026. They fall again to 100 days on March 15, 2027. By March 15, 2029, they drop to just 47 days, with domain validation itself needing to be re-proven roughly every 10 days.

01

Translate that into operational terms and the picture gets stark quickly. An organization currently renewing certificates a few times a year will be handling renewal events on the order of every couple of weeks by the end of this countdown, across every endpoint it operates. Manual tracking, calendar reminders, and a spreadsheet somebody checks once a month will not survive contact with that cadence. What has always been an occasional chore is becoming a continuous, automated operation, whether an organization plans for it or not.

02

As the schedule below continues shrinking maximum certificate lifetimes toward 47 days, the exposure window any imperfect revocation mechanism has to cover keeps getting smaller too, but that shrinking window is only a safety net for organizations that have already automated renewal; it is no substitute for actually checking revocation status where it genuinely matters.

03

The 200-day, 100-day, and 47-day milestones are not distant hypotheticals; the first has already arrived. Organizations that build the automation loop now, generating keys, vaulting them securely, brokering issuance across Certificate Authorities through APIs, and rebinding certificates to live endpoints without manual intervention, will meet each deadline without disruption. Organizations that wait will be rebuilding their certificate operations under deadline pressure, with far less room for error and far less time to get it right. The countdown is the call to action. The only real decision left is whether to automate on your own schedule, or on the CA/Browser Forum's.