GRIDINSOFT
HELP CENTER

OFFICIAL SUPPORT GUIDE

C

CDN SSL/TLS Security: Edge-to-Origin HTTPS Guide

A CDN usually terminates one TLS connection from the visitor and creates another to the origin. Learn how certificates, strict origin validation, HTTPS redirects, HSTS, origin lockdown, and mTLS fit together.

PUBLISHER
Gridinsoft Support

UPDATED
Oct 8, 2026

READ TIME
4 min

CONTENTS

In this guide

01 How TLS works through a CDN

02 Visitor, edge, and origin certificates

03 Encryption modes and the dangerous gap

04 Secure CDN TLS checklist

05 What TLS does and does not protect

06 Common failures

07 Frequently asked questions

CDN SSL/TLS security protects website traffic using HTTPS while a content delivery network sits between visitors and the origin server. In a typical setup there are two separate encrypted connections: **visitor to CDN edge** and **CDN edge to origin**. Both

must be configured correctly to provide end-to-end protection across the delivery path.

“SSL certificate” remains a common search term, but modern services use TLS. SSLv2 and SSLv3 are obsolete. Product dashboards may still label the section SSL/TLS for familiarity.

01 How TLS works through a CDN

1. A visitor requests `https://example.com` and reaches a nearby CDN edge.
2. The edge presents a certificate valid for that hostname and negotiates TLS with the visitor.
3. The CDN terminates and decrypts the request so it can apply caching, routing, web-application firewall, and other configured services.
4. For an uncached request, the CDN creates a separate connection to the origin.
5. With secure origin settings, that connection uses HTTPS and validates the origin certificate and hostname.
6. The CDN receives the origin response, applies configured controls, and encrypts its response to the visitor.

The CDN can see plaintext application data where TLS terminates. If that is outside the site’s trust or compliance model, use appropriate contractual, architectural, or application-layer controls rather than assuming TLS hides content from the CDN provider.

02 Visitor, edge, and origin certificates

Certificate	Presented to	Requirement
Edge certificate	Website visitors	Covers every public hostname and chains to a browser-trusted authority
Origin certificate	The CDN	Valid, unexpired, hostname-matching, and trusted by the CDN’s origin policy
Client certificate for mTLS	CDN edge or origin	Used only when clients or CDN requests must authenticate with certificates

An origin-only certificate issued by a CDN provider may be trusted by that CDN but not by ordinary browsers. Do not expose such an origin directly and expect visitors to trust it.

03 Encryption modes and the dangerous gap

CDN vendors use different names, but modes generally range from no encryption, through edge-only encryption, to HTTPS with strict origin validation. Edge-only or “flexible” configurations leave CDN-to-origin traffic unencrypted and can create redirect confusion. A mode that encrypts but does not validate the origin certificate may still connect to an impostor endpoint.

Use the vendor’s strict verified mode where possible. Cloudflare’s Full (strict), for example, requires an unexpired origin certificate from an accepted issuer with a matching hostname. AWS CloudFront can separately require HTTPS for viewers and for custom origins.

04 Secure CDN TLS checklist

- Issue edge certificates for all production hostnames and automate renewal.
- Redirect HTTP to HTTPS at the edge, then confirm APIs and non-browser clients behave correctly.
- Require HTTPS from CDN to origin with certificate and hostname validation.
- Allow modern TLS versions and a provider-maintained security policy; retain older protocols only for a measured compatibility need.
- Test the full certificate chain, SNI, alternate hostnames, and expiry monitoring.
- Fix mixed content so scripts, images, fonts, and API calls also use HTTPS.
- Enable HSTS only after every required subdomain supports HTTPS; increase duration gradually before considering `includeSubDomains` or preload.
- Restrict the origin to CDN egress ranges, authenticated origin pulls, mTLS, private connectivity, or a provider tunnel.
- Protect origin DNS and remove alternate hostnames that bypass CDN controls.

05 What TLS does and does not protect

TLS protects confidentiality and integrity in transit and authenticates the server according to certificate validation. It does not remove malware from an origin, fix application vulnerabilities, make cached private content safe, or stop attacks using valid HTTPS. A padlock means the connection to the named endpoint is encrypted; it does not certify that the website is honest.

06 Common failures

- **Redirect loop:** the origin believes the request used HTTP while the visitor used HTTPS. Configure trusted forwarded-protocol handling.
- **Certificate error:** missing intermediate, expired certificate, or hostname mismatch at the origin.
- **Origin bypass:** attackers connect directly to the server and evade CDN security controls.
- **Cache leak:** personalized or authenticated content is cached with an unsafe key.
- **Mixed content:** an HTTPS page loads active content over HTTP.

07 Frequently asked questions

Does a CDN provide an SSL certificate?

Most managed CDNs can issue or attach an edge certificate, but the domain owner must validate control and cover every required hostname. The origin usually needs its own certificate too.

Should I use mTLS between CDN and origin?

It can strongly authenticate CDN requests where supported, but it complements rather than replaces origin certificate validation, access control, patching, and application security.

NEED MORE HELP?

We are here when the guide is not enough.

Visit the Gridinsoft Support Center for current product guidance or submit a support request with the details of your case.

Open this guide online at support.gridinsoft.com