Diagnose an incomplete HTTPS certificate chain without disabling verification

A website may open on your laptop while a webhook, game plugin or fresh Linux installation rejects its HTTPS certificate. The browser may already know an intermediate certificate that the server forgot to send. Collect the served chain before replacing certificates or changing application security settings.

1. Record which connection fails

Write down the exact hostname, client error and time. Check whether the failure affects every client or one runtime. If the domain uses a reverse proxy, a successful public test checks that proxy; it does not prove the VPS origin serves the same certificate. These examples use `example.com`; replace it with your own hostname.

2. Ask for the chain and verify the hostname

```
timeout 15s openssl s_client -connect example.com:443 -servername example.com  -
verify_hostname example.com -verify_return_error -showcerts </dev/null
```

The 15-second timeout bounds this diagnostic; expiry alone does not identify a certificate fault. The server-name option requests the correct virtual host. Read the verification result as well as the certificate list. A missing issuer or an unable-to-build-chain error is a lead, not proof that the server alone is wrong: the client's trust store may also be old. An expired certificate or hostname mismatch needs a different correction.

3. Compare the origin without changing DNS

If you administer the origin and direct access is allowed, make one test using its real address. The documentation-only address below must be replaced. Keep the public hostname in the URL so both SNI and hostname verification remain correct.

```
curl --connect-timeout 5 --max-time 15  --resolve example.com:443:192.0.2.10 -I
https://example.com/
```

A timeout from a restricted origin is expected if only the proxy may reach it. Do not open access just for this test. Cloudflare Origin CA certificates also are not normally trusted by public clients; that alone does not mean they are invalid for Cloudflare-to-origin use.

4. Review the active certificate file

On Nginx, inspect the relevant server configuration privately and confirm which file `ssl_certificate` uses. A normal public chain file contains the site certificate followed by the required intermediate certificates. A file named `cert.pem` is not automatically a full chain. Use the bundle provided by your certificate issuer or ACME client; do not download an arbitrary issuer certificate from an error-page link.

Back up the affected configuration before a correction. Have the web-server administrator validate the configuration and arrange a controlled reload. Do not paste private keys, a full configuration dump, or account tokens into a public ticket.

Verify and escalate

Repeat the same verified test from the originally failing client and a second trusted machine. Success means the intended hostname verifies, the application connects, and the result survives a new connection. Never make `curl -k` or a disabled Java trust check the permanent solution. Send CloudExa support the hostname, UTC time, verification error and certificate issuer/expiry if the origin remains unclear.

Related: test Certbot renewal and configure your domain and HTTPS.

Official references

- OpenSSL s_client options
- Nginx certificate-chain configuration
- curl connection and verification options

Revision #1
Created 2026-11-10 10:51:41 UTC by CloudExa Hosting Admin
Updated 2026-11-10 10:51:41 UTC by CloudExa Hosting Admin