7.14 TLS- oder HTTPS-Verbindung schlägt fehl

7.14.1 Ziel dieser Seite

Diese Seite beschreibt die systematische Diagnose, wenn eine TLS- oder HTTPS-Verbindung nicht aufgebaut werden kann, eine Zertifikatswarnung erscheint oder eine Anwendung erst nach dem TLS-Handshake fehlschlägt.

Ziele der Diagnose:

- DNS-, Netzwerk-, TCP-, TLS- und HTTP-Fehler voneinander trennen;
- den tatsächlich verwendeten Zielhost und Port bestimmen;
- Zertifikatsname, Gültigkeit, Vertrauenskette und Sperrstatus prüfen;
- TLS-Versionen, Cipher Suites, SNI und ALPN untersuchen;
- Serverauthentifizierung und gegenseitige TLS-Authentifizierung unterscheiden;
- Proxy, TLS-Inspection, Reverse Proxy und Load Balancer berücksichtigen;
- client-, server- und anwendungsspezifische Truststores vergleichen;
- eine kontrollierte Maßnahme mit anschließender Verifikation durchführen.

Eine erfolgreiche TCP-Verbindung zu Port `443` beweist noch nicht, dass TLS oder HTTPS funktioniert.

7.14.2 Sicherheitskennzeichnungen

Kennzeichnung	Bedeutung
LESEND	erfasst ausschließlich Zustände und Messwerte
NETZAKTIV	baut eine Verbindung auf oder ruft externe Ressourcen ab
SENSITIV	kann Zertifikate, Header, interne Namen oder Sicherheitsinformationen anzeigen
ÄNDERND	verändert Konfigurationen, Truststores oder Dienste
AUSFALLRISIKO	kann bestehende TLS-Verbindungen oder Dienste unterbrechen

Grundregeln:

- Private Schlüssel niemals in Tickets, Chats oder Diagnoseausgaben kopieren.
- Private Schlüssel nicht per E-Mail oder ungeschützt übertragen.
- `curl -k` beziehungsweise `--insecure` nicht als dauerhafte Lösung verwenden.
- Zertifikatsprüfung nicht deaktivieren, um einen Fehler zu verdecken.
- Keine veralteten TLS-Versionen dauerhaft aktivieren.
- Keine CA-Zertifikate ungeprüft als vertrauenswürdig importieren.
- Bestehende Zertifikatsbindungen vor Änderungen dokumentieren.
- Bei Zertifikatswechseln immer Rückweg und Schlüsselzuordnung prüfen.

- Diagnoseausgaben vor Weitergabe auf Tokens, Cookies, Header und interne Namen prüfen.
- Netzwerkaufzeichnungen und TLS-Schlüsselprotokolle besonders schützen.

7.14.3 Vollständiger HTTPS-Verbindungspfad

```
Anwendung
→ Proxykonfiguration
→ DNS-Auflösung
→ IPv4- oder IPv6-Zieladresse
→ Routing und Firewall
→ TCP-Verbindung
→ TLS ClientHello
→ SNI und ALPN
→ TLS ServerHello
→ Serverzertifikat und Zertifikatskette
→ Zertifikatsprüfung
→ optionales Clientzertifikat
→ verschlüsselte HTTP-Anfrage
→ Reverse Proxy oder Load Balancer
→ Backend
→ HTTP-Antwort
```

Jede Ebene kann einen eigenen Fehler verursachen.

Eine Browsermeldung über HTTPS beweist nicht automatisch, dass das Zertifikat die Ursache ist. Der Fehler kann bereits bei DNS, TCP, Proxy, TLS-Version, SNI oder im Backend entstehen.

7.14.4 Fehlerklassen unterscheiden

Fehlerklasse	Typischer Befund
DNS-Fehler	Zielname wird nicht oder falsch aufgelöst
Routing- oder Firewallfehler	Zieladresse oder Port ist nicht erreichbar
TCP-Fehler	Timeout, Verbindungsablehnung oder TCP Reset
Proxyfehler	falscher Proxy, Authentifizierung oder blockierter CONNECT-Tunnel
TLS-Protokollfehler	Client und Server finden keine gemeinsame TLS-Version
Cipher-Fehler	keine gemeinsame Cipher Suite oder Signaturalgorithmus

Fehlerklasse	Typischer Befund
Zertifikatsnamenfehler	FQDN stimmt nicht mit dem SAN überein
Gültigkeitsfehler	Zertifikat ist abgelaufen oder noch nicht gültig
Vertrauensfehler	Root-CA wird nicht vertraut
Kettenfehler	Intermediate-CA fehlt oder Kette kann nicht aufgebaut werden
Sperrprüfungsfehler	CRL- oder OCSP-Prüfung schlägt fehl
SNI-Fehler	Server liefert das Zertifikat eines falschen virtuellen Hosts
ALPN-Fehler	HTTP/1.1, HTTP/2 oder anderes Anwendungsprotokoll wird falsch ausgehandelt
mTLS-Fehler	Clientzertifikat fehlt, ist falsch oder wird nicht akzeptiert
TLS-Inspection-Fehler	Proxy ersetzt das Serverzertifikat durch ein eigenes Zertifikat
Serverkonfigurationsfehler	Zertifikat, Schlüssel, Bindung oder Listener ist fehlerhaft
Reverse-Proxy-Fehler	Frontend-TLS funktioniert, Backend-TLS schlägt fehl
HTTP-Fehler	TLS funktioniert, Server liefert aber einen HTTP-Fehlercode
Anwendungsfehler	Browser funktioniert, bestimmte Anwendung jedoch nicht
Truststore-Abweichung	Betriebssystem, Browser, Java oder Container vertraut unterschiedlichen CAs

7.14.5 Typische Symptome und Fehlermeldungen

Mögliche Meldungen:

```
Certificate verify failed
Unable to get local issuer certificate
Unable to verify the first certificate
Self-signed certificate
Certificate has expired
Certificate is not yet valid
Hostname mismatch
Unknown CA
Bad certificate
Certificate required
Handshake failure
Protocol version
No shared cipher
```

```
TLS alert
Connection reset by peer
Connection refused
Operation timed out
ERR_SSL_PROTOCOL_ERROR
ERR_CERT_AUTHORITY_INVALID
ERR_CERT_DATE_INVALID
SEC_ERROR_UNKNOWN_ISSUER
SSL_ERROR_NO_CYPHER_OVERLAP
PKIX path building failed
unable to find valid certification path
The underlying connection was closed
Could not establish trust relationship
502 Bad Gateway
503 Service Unavailable
504 Gateway Timeout
```

Die genaue Meldung, Anwendung, Uhrzeit und Clientplattform müssen dokumentiert werden.

7.14.6 Mindestinformationen erfassen

Vor der Diagnose sind festzuhalten:

- vollständige URL;
- verwendeter FQDN;
- Port;
- Client-IP-Adresse;
- Ziel-IP-Adresse;
- Zeitpunkt;
- betroffene Anwendung;
- Betriebssystem und Version;
- Browser- oder Anwendungsversion;
- Proxy- und VPN-Zustand;
- erwarteter Zertifikatsaussteller;
- erwarteter Reverse Proxy oder Load Balancer;
- funktioniert HTTP ohne TLS?
- funktioniert die Verbindung von anderen Clients?
- funktioniert sie aus anderen Netzen?
- trat der Fehler nach einer Änderung auf?
- wurde kürzlich ein Zertifikat erneuert?
- ist mTLS vorgesehen?
- handelt es sich um öffentliches oder internes PKI-Vertrauen?

Private Schlüssel, Passwörter, Sitzungscookies, API-Keys und Zugriffstokens gehören nicht in die Dokumentation.

7.14.7 TCP-, TLS- und HTTP-Fehler trennen

Testebene	Erfolgsnachweis
DNS	vorgesehene A- oder AAAA-Adresse wird geliefert
Netzwerk	Route zum vorgesehenen Ziel besteht
TCP	Drei-Wege-Handshake mit Zielport funktioniert
TLS	Handshake wird mit gemeinsamer Version und Cipher abgeschlossen
Zertifikatsprüfung	Name, Gültigkeit, Kette und Vertrauen sind korrekt
HTTP	gültige HTTP-Antwort wird empfangen
Anwendung	ursprüngliche Funktion arbeitet vollständig

Mögliche Einordnung:

```
TCP schlägt fehl
→ noch kein TLS-Problem nachgewiesen

TCP funktioniert, TLS schlägt fehl
→ TLS-Version, Cipher, SNI, Zertifikat oder mTLS prüfen

TLS funktioniert, HTTP 502 erscheint
→ Reverse Proxy oder Backend prüfen

HTTP 200 erscheint, Anwendung funktioniert nicht
→ Anwendungslogik, API, Cookie, Authentifizierung oder Inhalt prüfen
```

7.14.8 DNS und Zieladresse prüfen

Windows

```
Resolve-DnsName `
  -Name "app.example.test" `
  -Type A
```

```
Resolve-DnsName `
  -Name "app.example.test" `
  -Type AAAA
```

Linux und macOS

```
dig app.example.test A
```

```
dig app.example.test AAAA
```

Zu prüfen sind:

- richtige IPv4-Adresse;
- richtige IPv6-Adresse;
- internes oder öffentliches DNS;
- Split-DNS;
- VPN-DNS;
- CNAME-Kette;
- mehrere Zieladressen;
- veralteter Cache;
- CDN- oder Load-Balancer-Ziel;
- Hosts-Datei;
- DNS-Suffixe;
- Anwendung mit eigenem Resolver.

Wenn mehrere IP-Adressen vorhanden sind, müssen die tatsächlich verwendete Adresse und das Verhalten der einzelnen Ziele verglichen werden.

7.14.9 IPv4 und IPv6 getrennt testen

NETZAKTIV

```
curl \
  -4 \
  -v \
  https://app.example.test/
```

```
curl \
  -6 \
```

```
  -v \
  https://app.example.test/
```

Möglicher Befund:

- IPv4 funktioniert, IPv6 schlägt fehl;
- einzelne IPv4-Adresse liefert ein falsches Zertifikat;
- nur ein Load-Balancer-Node ist fehlerhaft;
- IPv6 führt zu einem anderen Reverse Proxy;
- DNS liefert eine veraltete Adresse.

Ein scheinbarer TLS-Fehler kann durch einen fehlerhaften IPv6-Pfad oder ein falsch konfiguriertes Zielsystem verursacht werden.

7.14.10 TCP-Port prüfen

Windows

```
Test-NetConnection `
  -ComputerName "app.example.test" `
  -Port 443
```

Wichtige Felder:

```
RemoteAddress
RemotePort
InterfaceAlias
SourceAddress
TcpTestSucceeded
```

Linux und macOS

```
nc \
  -vz \
  app.example.test \
  443
```

Alternativ kann bereits `curl -v` oder `openssl s_client` zum Aufbau der TCP-Verbindung verwendet werden.

Mögliche Ergebnisse:

Ergebnis	Einordnung
Timeout	Firewall, Routing, Proxy oder nicht antwortender Server
Connection refused	Ziel erreichbar, aber kein Listener oder aktive Ablehnung
TCP Reset	Firewall, Load Balancer oder Dienst beendet Verbindung
TCP erfolgreich	erst jetzt TLS-Handshake untersuchen

Ein erfolgreicher Ping beweist keine Erreichbarkeit von TCP-Port `443`.

7.14.11 Systemzeit und Zeitzone prüfen

Zertifikate besitzen einen Gültigkeitszeitraum:

```
Not Before
Not After
```

Eine falsche Systemzeit kann ein gültiges Zertifikat als abgelaufen oder noch nicht gültig erscheinen lassen.

Windows

```
Get-Date
```

```
w32tm /query /status
```

```
w32tm /query /source
```

Linux

```
date
```

```
date -u
```

```
timedatectl status
```

macOS

```
date
```

```
date -u
```

```
systemsetup -gettimezone
```

Zu prüfen sind:

- Datum;
- Uhrzeit;
- Zeitzone;
- UTC-Abweichung;
- NTP-Quelle;
- Synchronisationsstatus;
- virtuelle Maschine mit fehlerhafter Hostzeitsynchronisation;
- Containerzeit;
- Domänenzeit;
- kürzlich geänderte Uhr.

7.14.12 HTTPS mit `curl` prüfen

NETZAKTIV, SENSITIV

```
curl \
  --connect-timeout 10 \
  --max-time 30 \
  -v \
  https://app.example.test/
```

Die verbose Ausgabe kann zeigen:

- verwendete Zieladresse;
- Proxyverwendung;
- TLS-Version;
- Cipher Suite;
- Zertifikatsbetreff;
- Zertifikatsaussteller;
- ALPN-Ergebnis;

- HTTP-Anfrage;
- HTTP-Statuscode;
- Redirect;
- Verbindungsabbruch.

Verbose Ausgaben können sensible HTTP-Header enthalten und müssen vor Weitergabe geprüft werden.

Nur die HTTP-Header anzeigen:

```
curl \
  -sS \
  -D - \
  -o /dev/null \
  https://app.example.test/
```

Einige Anwendungen unterstützen keine `HEAD`-Anfrage. Deshalb ist eine normale `GET`-Anfrage mit verworfener Antwort häufig aussagekräftiger als `curl -I`.

7.14.13 Zertifikatsprüfung testweise abgrenzen

NUR ZUR DIAGNOSE, NICHT ALS LÖSUNG

```
curl \
  -v \
  -k \
  https://app.example.test/
```

Einordnung:

Normaler Test	Test mit `-k`	Einordnung
schlägt fehl	funktioniert	Zertifikatsvalidierung wahrscheinlich betroffen
schlägt fehl	schlägt ebenfalls fehl	nicht nur Zertifikatsvertrauen betroffen
funktioniert	funktioniert	kein aktueller Validierungsfehler
funktioniert nur auf einem Client	anderer Client schlägt fehl	Truststore oder Anwendung unterscheidet sich

`-k` deaktiviert die Überprüfung des Serverzertifikats. Es behebt weder DNS-, TCP-, TLS-Version-, Cipher-, SNI-, ALPN- noch mTLS-Probleme zuverlässig.

Die Option darf nicht in produktiven Skripten, Anwendungen oder dauerhaften Konfigurationen verbleiben.

7.14.14 Bestimmte IP-Adresse mit richtigem Hostnamen testen

Ein direkter Aufruf über eine IP-Adresse verändert häufig:

- den geprüften Zertifikatsnamen;
- SNI;
- HTTP-Hostheader;
- Auswahl des virtuellen Hosts.

Deshalb sollte `curl --resolve` verwendet werden:

```
curl \
  -v \
  --resolve \
  app.example.test:443:192.0.2.25 \
  https://app.example.test/
```

Damit bleiben erhalten:

- URL-Hostname;
- SNI;
- Zertifikatsnamenprüfung;
- HTTP-Hostheader.

Nur die Ziel-IP-Adresse wird für den Test festgelegt.

Dies ist besonders nützlich bei:

- mehreren Load-Balancer-Nodes;
- DNS-Umstellungen;
- fehlerhaftem Caching;
- Blue-Green-Deployments;
- alten und neuen Reverse Proxys;
- einzelnen fehlerhaften Backendpfaden.

7.14.15 TLS-Handshake mit OpenSSL prüfen

NETZAKTIV, SENSITIV

```
openssl s_client \
  -connect app.example.test:443 \
  -servername app.example.test \
  -showcerts \
  -verify_return_error \
  -verify_hostname app.example.test \
  </dev/null
```

Wichtige Ausgabebereiche:

```
CONNECTED
Certificate chain
subject
issuer
Server certificate
SSL handshake
Protocol
Cipher
ALPN protocol
Verify return code
```

Erfolgreiche OpenSSL-Verifikation:

```
Verify return code: 0 (ok)
```

Wichtige Optionen:

Option	Bedeutung
`-connect`	Zielhost und Port
`-servername`	übermittelt SNI
`-showcerts`	zeigt die vom Server gesendeten Zertifikate
`-verify_return_error`	beendet bei Validierungsfehlern
`-verify_hostname`	prüft den angegebenen Hostnamen
`</dev/null`	beendet die Eingabe nach dem Handshake

Nicht jede ältere OpenSSL-Version unterstützt alle genannten Optionen.

7.14.16 Zertifikat lokal untersuchen

Vorhandene Zertifikatsdatei im PEM-Format:

LESEND

```
openssl x509 \
  -in server.pem \
  -noout \
  -subject \
  -issuer \
  -serial \
  -dates \
  -fingerprint \
  -sha256
```

Subject Alternative Names:

```
openssl x509 \
  -in server.pem \
  -noout \
  -ext subjectAltName
```

Vollständige Zertifikatsinformationen:

```
openssl x509 \
  -in server.pem \
  -noout \
  -text
```

Zu prüfen sind:

- Subject;
- Subject Alternative Name;
- Issuer;
- Seriennummer;
- `Not Before`;
- `Not After`;
- Public-Key-Algorithmus;
- Schlüssellänge;
- Signaturalgorithmus;

- Key Usage;
- Extended Key Usage;
- Basic Constraints;
- Authority Information Access;
- CRL Distribution Points;
- SHA-256-Fingerprint.

Private Schlüssel dürfen mit diesen Ausgaben nicht verwechselt oder weitergegeben werden.

7.14.17 Hostname und Subject Alternative Name prüfen

Für die Identitätsprüfung ist der vorgesehene Dienstname entscheidend.

Beispiel:

```
Aufgerufener Name:
app.example.test

Erforderlicher SAN:
DNS:app.example.test
```

Zu prüfen sind:

- vollständiger FQDN;
- SAN-Einträge;
- Wildcard-Gültigkeit;
- interne und öffentliche Namen;
- Groß- und Kleinschreibung;
- internationalisierte Domainnamen;
- Portweiterleitungen;
- Aliase und CNAME-Ziele;
- Zugriff über IP-Adresse;
- Reverse Proxy mit abweichendem Backendnamen.

Beispiele:

```
*.example.test
```

kann typischerweise abdecken:

```
app.example.test
mail.example.test
```

aber nicht:

```
example.test
app.intern.example.test
```

Der CNAME-Zielname muss nicht zwingend im Zertifikat stehen. Entscheidend ist grundsätzlich der Name, unter dem der Client den Dienst aufruft und dessen Identität er prüft.

7.14.18 Zertifikatskette verstehen

Eine typische Kette:

```
Serverzertifikat
→ Intermediate-CA
→ Root-CA
```

Aufgaben:

Element	Aufgabe
Serverzertifikat	identifiziert den Dienst
Intermediate-CA	verbindet Serverzertifikat mit vertrauenswürdiger CA
Root-CA	bildet den lokalen Vertrauensanker

Der Server sollte gewöhnlich senden:

```
Serverzertifikat
Intermediate-CA-Zertifikat
weitere erforderliche Intermediate-Zertifikate
```

Die Root-CA wird gewöhnlich nicht als Teil der Serverkette benötigt, weil sie bereits im Truststore des Clients vorhanden sein muss.

Typische Kettenfehler:

- Intermediate-Zertifikat fehlt;
- falsches Intermediate-Zertifikat;
- abgelaufene Intermediate-CA;
- Client vertraut der Root-CA nicht;
- mehrere mögliche Zertifizierungspfade;
- veraltete Root- oder Intermediate-CA;

- AIA-Ziel nicht erreichbar;
- Server liefert Zertifikate in falscher Reihenfolge;
- Anwendung verwendet einen eigenen Truststore;
- TLS-Inspection verwendet eine unbekannte Unternehmens-CA.

7.14.19 Zertifikatskette lokal verifizieren

Vorhandene Dateien:

```
server.pem
intermediates.pem
root-ca.pem
```

LESEND

```
openssl verify \
  -purpose sslserver \
  -CAfile root-ca.pem \
  -untrusted intermediates.pem \
  server.pem
```

Erfolgreiches Ergebnis:

```
server.pem: OK
```

Dieser Test beweist die Gültigkeit nur gegenüber den ausdrücklich angegebenen CA-Dateien. Er beweist nicht, dass der Truststore der ursprünglichen Anwendung identisch konfiguriert ist.

7.14.20 Gültigkeitszeitraum prüfen

```
openssl x509 \
  -in server.pem \
  -noout \
  -dates
```

Mögliche Ausgabe:

```
notBefore=...
notAfter=...
```

Zu prüfen sind:

- Zertifikat bereits gültig?
- Zertifikat noch gültig?
- Clientzeit korrekt?
- Serverzeit korrekt?
- Zwischenzertifikate noch gültig?
- Root-CA noch gültig?
- erneuertes Zertifikat tatsächlich aktiv?
- alter Load-Balancer-Node liefert noch altes Zertifikat?
- Anwendung verwendet weiterhin einen alten Zertifikatsspeicher?
- Dienst wurde nach Zertifikatswechsel kontrolliert neu geladen?

Ein erneuertes Zertifikat auf dem Dateisystem beweist nicht, dass der Listener dieses Zertifikat bereits verwendet.

7.14.21 Sperrprüfung, CRL und OCSP untersuchen

Mögliche Sperrmechanismen:

- Certificate Revocation List;
- Delta CRL;
- Online Certificate Status Protocol;
- OCSP Stapling;
- anwendungsspezifische Sperrprüfung.

OCSP-Stapling des Servers anfordern:

NETZAKTIV

```
openssl s_client \
  -connect app.example.test:443 \
  -servername app.example.test \
  -status \
  </dev/null
```

Unter Windows kann eine vorhandene Zertifikatsdatei geprüft werden:

NETZAKTIV, kann AIA- und Sperrlistenadressen abrufen

```
certutil -verify -urlfetch server.cer
```

Zu prüfen sind:

- CRL Distribution Point erreichbar;
- OCSP-Responder erreichbar;
- Proxyzugriff des Systemkontos;
- DNS-Auflösung der Sperrprüfungsziele;
- Gültigkeit der CRL;
- korrekte AIA-Adressen;
- Offline-Root-CA korrekt veröffentlicht;
- Firewall blockiert HTTP- oder LDAP-Abruf;
- Cache enthält veraltete Sperrinformationen;
- unterschiedliche Sperrprüfungsrichtlinien der Anwendungen.

Ein Fehler bei der Sperrprüfung darf nicht pauschal durch Abschalten der Sperrprüfung gelöst werden.

7.14.22 TLS-Versionen prüfen

Aktuelle TLS-Version aushandeln:

```
openssl s_client \
  -connect app.example.test:443 \
  -servername app.example.test \
  </dev/null
```

TLS 1.2 ausdrücklich testen:

```
openssl s_client \
  -connect app.example.test:443 \
  -servername app.example.test \
  -tls1_2 \
  </dev/null
```

TLS 1.3 ausdrücklich testen:

```
openssl s_client \
  -connect app.example.test:443 \
  -servername app.example.test \
```

```
  -tls1_3 \
  </dev/null
```

Zu prüfen sind:

- vom Client unterstützte Versionen;
- vom Server aktivierte Versionen;
- Systemrichtlinien;
- Anwendungsframework;
- Proxy oder Load Balancer;
- TLS-Inspection;
- Backend-TLS;
- veraltete Bibliothek;
- FIPS- oder Sicherheitsrichtlinie.

Das Aktivieren veralteter TLS-Versionen ist keine geeignete Dauerlösung. Stattdessen muss die veraltete Gegenstelle aktualisiert oder ersetzt werden.

7.14.23 Cipher Suites und Algorithmen prüfen

Lokal verfügbare OpenSSL-Cipher anzeigen:

```
openssl ciphers -v
```

Unter Windows:

```
Get-TlsCipherSuite |
  Select-Object `
    Name,
    Protocols,
    Cipher,
    Hash,
    Exchange
```

Zu unterscheiden sind:

- TLS-Version;
- Cipher Suite;
- Schlüsselaustausch;
- Serverzertifikatstyp;
- Signaturalgorithmus;
- unterstützte Gruppen;

- Client- und Serverrichtlinien;
- FIPS-Anforderungen;
- Sicherheitsprodukt mit eigener TLS-Policy.

Ein RSA-Serverzertifikat, eine ECDSA-Cipher und ein bestimmter Schlüsselaustausch sind unterschiedliche Bestandteile und dürfen nicht gleichgesetzt werden.

TLS 1.3 definiert Cipher Suites anders als frühere TLS-Versionen. Deshalb ist eine reine Cipherliste ohne TLS-Version und Clientkontext unvollständig.

7.14.24 SNI prüfen

SNI übermittelt den gewünschten Servernamen bereits während des TLS-Handshakes.

Mit SNI:

```
openssl s_client \
  -connect 192.0.2.25:443 \
  -servername app.example.test \
  </dev/null
```

Ohne ausdrückliches SNI:

```
openssl s_client \
  -connect 192.0.2.25:443 \
  </dev/null
```

Wenn unterschiedliche Zertifikate erscheinen, verwendet der Server namensbasierte virtuelle TLS-Hosts.

Zu prüfen sind:

- korrekter SNI-Name;
- Default-Zertifikat des Listeners;
- virtuelle Hosts;
- Load-Balancer-Regeln;
- Reverse-Proxy-Konfiguration;
- Backend-SNI;
- Anwendung unterstützt SNI;
- direkter IP-Aufruf;
- falscher `proxy_ssl_name`;
- falscher Backendhostname.

Ein Test ohne SNI kann ein anderes Zertifikat liefern als die reale Anwendung und darf deshalb nicht allein bewertet werden.

7.14.25 ALPN, HTTP/1.1 und HTTP/2 prüfen

ALPN wird während des TLS-Handshakes verwendet, um das Anwendungsprotokoll auszuhandeln.

```
openssl s_client \
  -connect app.example.test:443 \
  -servername app.example.test \
  -alpn h2,http/1.1 \
  </dev/null
```

Mit `curl`:

```
curl \
  --http1.1 \
  -v \
  https://app.example.test/
```

Falls die verwendete curl-Version HTTP/2 unterstützt:

```
curl \
  --http2 \
  -v \
  https://app.example.test/
```

Zu prüfen sind:

- ausgewähltes ALPN-Protokoll;
- Unterstützung von HTTP/1.1;
- Unterstützung von HTTP/2;
- Reverse Proxy;
- Load Balancer;
- TLS-Inspection;
- Backendprotokoll;
- fehlerhafte HTTP/2-Konfiguration;
- Anwendung erwartet ein bestimmtes Protokoll.

HTTP/3 verwendet QUIC über UDP und muss getrennt von klassischem HTTPS über TCP untersucht werden.

7.14.26 Gegenseitige TLS-Authentifizierung prüfen

Bei normalem TLS authentifiziert sich der Server gegenüber dem Client.

Bei Mutual TLS authentifizieren sich beide Seiten:

```
Serverzertifikat
+
Clientzertifikat
```

Test mit autorisiertem Clientzertifikat:

NETZAKTIV, SENSITIV

```
openssl s_client \
  -connect app.example.test:443 \
  -servername app.example.test \
  -cert client.pem \
  -key client.key \
  -verify_return_error \
  </dev/null
```

Zu prüfen sind:

- fordert der Server ein Clientzertifikat an?
- besitzt der Client das richtige Zertifikat?
- ist der private Schlüssel vorhanden?
- kann die Anwendung auf den privaten Schlüssel zugreifen?
- ist das Clientzertifikat gültig?
- enthält es den vorgesehenen Verwendungszweck?
- vertraut der Server der ausstellenden Client-CA?
- ist die Clientzertifikatskette vollständig?
- ist das Zertifikat gesperrt?
- ist die Zertifikatszuordnung korrekt?
- verändert ein Proxy die mTLS-Verbindung?
- endet mTLS am Load Balancer oder am Backend?

Die Datei `client.key` ist besonders schützenswert. Pfade, Berechtigungen und Diagnoseausgaben müssen entsprechend behandelt werden.

7.14.27 Proxykonfiguration prüfen

Windows-WinHTTP-Proxy

```
netsh winhttp show proxy
```

Linux und macOS

```
env |
grep -i proxy
```

Mögliche Variablen:

```
HTTP_PROXY
HTTPS_PROXY
NO_PROXY
http_proxy
https_proxy
no_proxy
```

Zu prüfen sind:

- verwendet die Anwendung überhaupt den Systemproxy?
- verwendet sie WinHTTP, WinINet oder eine eigene Bibliothek?
- stimmt der Proxyhostname?
- stimmt der Proxyport?
- funktioniert Proxy-DNS?
- ist Proxy-Authentifizierung erforderlich?
- erlaubt der Proxy die CONNECT-Methode zum Zielport?
- enthält `NO_PROXY` den vorgesehenen internen Namen?
- wird der FQDN oder die IP-Adresse verglichen?
- gilt die Proxyregel für das Dienstkonto?
- erreicht der Proxy CRL-, OCSP- und AIA-Ziele?
- verursacht eine PAC-Datei einen anderen Pfad?
- verwendet die Anwendung einen fest eingebauten Proxy?

Ein erfolgreicher Browsertest beweist nicht, dass ein Windows-Dienst, Java-Prozess oder Container dieselbe Proxykonfiguration verwendet.

7.14.28 TLS-Inspection erkennen

Bei TLS-Inspection beendet ein Sicherheitsgerät die TLS-Verbindung und baut anschließend eine neue TLS-Verbindung zum Ziel auf.

Mögliche Hinweise:

- Zertifikatsaussteller ist eine Unternehmens- oder Security-CA;
- Zertifikat unterscheidet sich je nach Netzwerk;
- Browser funktioniert, Container oder Java-Anwendung nicht;
- nur verwaltete Clients vertrauen dem Zertifikat;
- Zertifikat-Fingerprint ändert sich hinter dem Proxy;
- mTLS funktioniert nicht über den Inspecting Proxy;
- bestimmte TLS-Versionen oder Cipher werden verändert;
- OCSP- oder Zertifikatsinformationen unterscheiden sich.

Vergleich:

1. Zertifikat vom betroffenen Client erfassen.
2. Zertifikat aus einem anderen autorisierten Netzwerk erfassen.
3. Subject, Issuer, SAN, Seriennummer und Fingerprint vergleichen.
4. Proxy- und Sicherheitsrichtlinie prüfen.
5. Truststore der betroffenen Anwendung prüfen.

TLS-Inspection darf nicht ohne Freigabe umgangen werden. Eine notwendige Ausnahme muss über den vorgesehenen Sicherheitsprozess erfolgen.

7.14.29 Windows-Zertifikatsspeicher prüfen

Stores anzeigen:

LESEND

```
Get-ChildItem `
  -Path Cert:\CurrentUser
```

```
Get-ChildItem `
  -Path Cert:\LocalMachine
```

Persönliche Zertifikate des Computers:

```
Get-ChildItem `
  -Path Cert:\LocalMachine\My |
Select-Object `
  Subject,
  Issuer,
```

```
  Thumbprint,
  NotBefore,
  NotAfter,
  HasPrivateKey
```

Vertrauenswürdige Root-CAs:

```
Get-ChildItem `
  -Path Cert:\LocalMachine\Root |
Select-Object `
  Subject,
  Issuer,
  Thumbprint,
  NotAfter
```

Intermediate-CAs:

```
Get-ChildItem `
  -Path Cert:\LocalMachine\CA |
Select-Object `
  Subject,
  Issuer,
  Thumbprint,
  NotAfter
```

Zu unterscheiden sind:

- `CurrentUser`;
- `LocalMachine`;
- Personal;
- Trusted Root Certification Authorities;
- Intermediate Certification Authorities;
- WebHosting;
- anwendungseigener Zertifikatsspeicher.

Ein Zertifikat im Benutzerstore steht einem Dienstkonto oder Computersystem nicht automatisch zur Verfügung.

7.14.30 Windows-Schannel-Ereignisse prüfen

LESEND

```
Get-WinEvent `
  -FilterHashtable @{
    LogName = "System"
    ProviderName = "Schannel"
    StartTime = (Get-Date).AddHours(-6)
  } |
Select-Object `
  TimeCreated,
  Id,
  LevelDisplayName,
  Message
```

Zu korrelieren sind:

- exakter Fehlerzeitpunkt;
- betroffene Anwendung;
- Client- oder Serverrolle;
- TLS-Alert;
- Zertifikatsproblem;
- fehlende gemeinsame Algorithmen;
- Clientzertifikatsanforderung;
- Richtlinienänderung;
- Systemupdate;
- Zertifikatswechsel.

Schannel-Ereignisse allein enthalten nicht immer den vollständigen Fehlerpfad. Sie müssen mit Anwendung, CAPI2, Proxy, Netzwerk und Serverseite korreliert werden.

7.14.31 Windows-CAPI2-Protokoll prüfen

Pfad in der Ereignisanzeige:

```
Anwendungs- und Dienstprotokolle
→ Microsoft
→ Windows
→ CAPI2
→ Operational
```

Vorhandene Ereignisse ausgeben:

```
Get-WinEvent `
  -LogName "Microsoft-Windows-CAPI2/Operational" `
  -MaxEvents 100 |
Select-Object `
  TimeCreated,
  Id,
  LevelDisplayName,
  Message
```

CAPI2 kann unter anderem Hinweise liefern zu:

- Aufbau der Zertifikatskette;
- ausgewähltem Vertrauenspfad;
- fehlender Root- oder Intermediate-CA;
- Sperrprüfung;
- AIA;
- CRL;
- Chain Policy;
- nicht vertrauenswürdigem Root-Zertifikat.

Das Protokoll ist möglicherweise nicht aktiviert. Eine Aktivierung ist eine Konfigurationsänderung und muss zeitlich begrenzt sowie dokumentiert werden.

7.14.32 Windows-TLS-Richtlinien erfassen

Verfügbare Cipher Suites:

```
Get-TlsCipherSuite
```

Vorhandene Schannel-Protokolleinstellungen:

```
Get-ChildItem `
  -Path "HKLM:\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols" `
  -Recurse `
  -ErrorAction SilentlyContinue
```

Zu prüfen sind:

- lokale Richtlinie;
- Gruppenrichtlinie;
- Schannel-Registrywerte;

- .NET-Einstellungen;
- IIS-Konfiguration;
- Anwendung mit eigener TLS-Bibliothek;
- Betriebssystemversion;
- installierte Updates;
- FIPS-Richtlinie;
- Sicherheitsbaseline.

Registrywerte dürfen nicht nach allgemeinen Internetanleitungen verändert werden. Betriebssystemversion, Anwendung und Microsoft-Dokumentation müssen zusammenpassen.

7.14.33 Linux- und macOS-Truststores prüfen

OpenSSL-Version:

```
openssl version -a
```

OpenSSL-Konfigurationsverzeichnis:

```
openssl version -d
```

Je nach Linux-Distribution können CA-Zertifikate unter anderem verwaltet werden über:

```
/etc/ssl/certs
/etc/pki/ca-trust
/usr/local/share/ca-certificates
```

Die tatsächlich verwendeten Pfade hängen von Distribution, TLS-Bibliothek und Anwendung ab.

macOS-Systemschlüsselbund anzeigen:

```
security find-certificate \
  -a \
  -Z \
  /Library/Keychains/System.keychain
```

Zu prüfen sind:

- System-Truststore;
- Benutzer-Truststore;
- anwendungseigener Truststore;

- OpenSSL-CA-Datei;
- Container-CA-Bundle;
- Browser-Truststore;
- Unternehmensprofil;
- MDM-verteilte Zertifikate;
- abgelaufene oder doppelte CA-Zertifikate.

Ein erfolgreicher Test mit Apple-Systemvertrauen beweist nicht, dass eine separat installierte OpenSSL- oder Java-Version denselben Truststore verwendet.

7.14.34 Java-Truststore prüfen

Java-Version:

```
java -version
```

Java-Pfad und Eigenschaften:

```
java \
  -XshowSettings:properties \
  -version
```

Standard-CA-Store anzeigen:

```
keytool \
  -list \
  -cacerts
```

Bestimmten Truststore anzeigen:

```
keytool \
  -list \
  -keystore <Truststore>
```

Zu prüfen sind:

- tatsächlich verwendete Java-Version;
- `java.home`;
- Truststorepfad;
- Truststoreformat;

- Truststorepasswort;
- Root- und Intermediate-CA;
- Zertifikatsalias;
- Gültigkeitszeitraum;
- Anwendung mit eigenem Truststore;
- Container besitzt anderen Java-Truststore;
- JVM-Startparameter;
- alte JVM ohne aktuelle CA-Liste.

Ein Import in den Betriebssystem-Truststore behebt keinen Java-Fehler, wenn die Anwendung ausschließlich ihren eigenen Truststore verwendet.

7.14.35 Container separat prüfen

Der Host und ein Container können unterschiedliche besitzen:

- DNS-Konfigurationen;
- Proxyvariablen;
- Systemzeiten;
- CA-Bundles;
- OpenSSL-Versionen;
- TLS-Bibliotheken;
- Java-Truststores;
- IPv4- oder IPv6-Pfade;
- Zertifikatsdateien.

Containerkonfiguration:

```
docker inspect \
  <Containername>
```

Proxyvariablen:

```
docker exec \
  <Containername> \
  env
```

Zeit im Container:

```
docker exec \
  <Containername> \
  date -u
```

Falls `curl` im Image vorhanden ist:

```
docker exec \
  <Containername> \
  curl \
  -v \
  https://app.example.test/
```

Vergleich:

```
Test auf dem Host
gegen
Test im Container
```

Wenn nur der Container fehlschlägt, sind insbesondere CA-Bundle, Proxy, DNS, Netzwerknamespace und Anwendungslaufzeit zu prüfen.

Produktive Images dürfen nicht spontan durch Installation zusätzlicher Diagnosepakete verändert werden. Falls Werkzeuge fehlen, ist ein freigegebener Diagnosecontainer oder ein reproduzierbares Testimage zu verwenden.

7.14.36 Serverzertifikat und privaten Schlüssel zuordnen

Zertifikat und privater Schlüssel müssen dasselbe Schlüsselpaar bilden.

Öffentlichen Schlüssel des Zertifikats hashen:

LESEND

```
openssl x509 \
  -in server.pem \
  -pubkey \
  -noout |
openssl pkey \
  -pubin \
  -outform DER |
openssl sha256
```

Öffentlichen Anteil des privaten Schlüssels hashen:

SENSITIV – Zugriff auf privaten Schlüssel erforderlich

```
openssl pkey \
  -in server.key \
  -pubout \
  -outform DER |
openssl sha256
```

Die beiden Hashwerte müssen übereinstimmen.

Dieser Test gibt nicht den privaten Schlüssel aus. Der Zugriff auf die Schlüsseldatei bleibt dennoch sicherheitskritisch.

Zu prüfen sind zusätzlich:

- Dateipfad;
- Dateiberechtigungen;
- Dateibesitzer;
- Verschlüsselung des Schlüssels;
- Kennwortzugriff;
- Dienstkonto;
- Hardware Security Module;
- Secret- oder Vault-Bereitstellung;
- Zertifikatsbindung;
- tatsächlich geladene Datei.

7.14.37 Nginx prüfen

Konfigurationssyntax:

LESEND

```
sudo nginx -t
```

Dienststatus:

```
systemctl \
  status nginx \
  --no-pager
```

Protokolle:

```
journalctl \
  -u nginx \
  --since "-2 hours"
```

Zu prüfen sind:

- `listen 443 ssl`;
- richtiger virtueller Host;
- `server_name`;
- Zertifikatspfad;
- Schlüsselpfad;
- vollständige Zertifikatskette;
- Dateiberechtigungen;
- SNI;
- TLS-Protokolle;
- Cipher-Konfiguration;
- Clientzertifikatsanforderung;
- OCSP-Konfiguration;
- Reverse-Proxy-Backend;
- Backend-SNI;
- Backend-Zertifikatsprüfung;
- kontrollierter Reload nach Änderung.

Bei einer kombinierten Zertifikatsdatei steht gewöhnlich zuerst das Serverzertifikat, gefolgt von den erforderlichen Intermediate-Zertifikaten.

7.14.38 Apache HTTP Server prüfen

Konfigurationssyntax:

LESEND

```
sudo apachectl configtest
```

Alternativ, abhängig von der Installation:

```
sudo httpd -t
```

Dienststatus:

```
systemctl \
  status apache2 \
  --no-pager
```

oder:

```
systemctl \
  status httpd \
  --no-pager
```

Zu prüfen sind:

- SSL-Modul;
- virtueller Host;
- Port und Listener;
- `ServerName`;
- Zertifikatsdatei;
- Schlüsseldatei;
- Zertifikatskette;
- Dateiberechtigungen;
- TLS-Versionen;
- Cipher;
- Clientzertifikate;
- Proxy zum Backend;
- Protokolle;
- kontrollierter Reload.

Der genaue Dienstname und die Konfigurationspfade hängen von Distribution und Installation ab und dürfen nicht angenommen werden.

7.14.39 IIS-Bindungen prüfen

Falls das WebAdministration-Modul vorhanden ist:

```
Import-Module WebAdministration
```

HTTPS-Bindungen:

```
Get-WebBinding `
  -Protocol https
```

SSL-Bindungen:

```
Get-ChildItem `
  -Path IIS:\SslBindings
```

Zu prüfen sind:

- IP-Adresse;
- Port;
- Hostname;
- SNI-Einstellung;
- Zertifikat-Thumbprint;
- Zertifikatsspeicher;
- privater Schlüssel;
- Berechtigung des Dienstkontos;
- abgelaufenes Zertifikat;
- falsche Bindung;
- mehrere Sites auf demselben Listener;
- zentrale Zertifikatsspeicherung;
- Clientzertifikatsanforderung.

Ein gültiges Zertifikat im Windows-Zertifikatsspeicher beweist nicht, dass es an die richtige IIS-Site gebunden ist.

7.14.40 Reverse Proxy und Load Balancer getrennt prüfen

Mögliche TLS-Abschnitte:

```
Client
→ TLS zum Load Balancer
→ TLS zum Reverse Proxy
→ TLS zum Backend
```

Oder:

```
Client
→ TLS-Terminierung am Load Balancer
→ unverschlüsseltes HTTP zum Backend
```

Für jeden Abschnitt getrennt erfassen:

- Zielhostname;

- Ziel-IP-Adresse;
- Port;
- SNI;
- Zertifikatsname;
- ausstellende CA;
- Truststore;
- TLS-Version;
- Cipher;
- Clientzertifikat;
- Healthcheck;
- Timeout;
- Backendstatus.

Typische Fehler:

- Frontendzertifikat korrekt, Backendzertifikat abgelaufen;
- Load Balancer sendet falsches SNI;
- Backendzertifikat gilt nicht für den internen Namen;
- Proxy vertraut der internen CA nicht;
- Healthcheck verwendet HTTP statt HTTPS;
- Healthcheck verwendet IP-Adresse statt FQDN;
- Backend erwartet mTLS;
- nur ein Load-Balancer-Node besitzt das neue Zertifikat;
- Zertifikatsverteilung ist unvollständig;
- Proxy prüft Backendzertifikat nicht oder mit falschem Namen.

Ein erfolgreicher externer TLS-Test prüft nicht automatisch die Verbindung vom Reverse Proxy zum Backend.

7.14.41 HTTP-Ebene nach erfolgreichem TLS prüfen

Wenn der TLS-Handshake erfolgreich ist, muss der HTTP-Statuscode ausgewertet werden.

Status	Mögliche Einordnung
`200`	Anfrage grundsätzlich erfolgreich
`301` oder `302`	Redirectziel prüfen
`400`	fehlerhafte Anfrage oder Hostheader
`401`	Authentifizierung erforderlich oder fehlgeschlagen
`403`	Zugriff verweigert
`404`	Pfad oder virtueller Host falsch
`405`	HTTP-Methode nicht erlaubt
`408`	Request Timeout

Status	Mögliche Einordnung
`421`	Anfrage beim falschen Ursprung oder falscher Verbindung
`429`	Rate Limit
`500`	Anwendungs- oder Serverfehler
`502`	Proxy erreicht Backend nicht korrekt
`503`	Dienst oder Backend nicht verfügbar
`504`	Backend antwortet nicht rechtzeitig

Redirects verfolgen:

```
curl \
  -v \
  -L \
  https://app.example.test/
```

Zu prüfen sind:

- Redirect auf anderen Hostnamen;
- Redirect auf abgelaufenes Zertifikat;
- Wechsel zwischen internem und öffentlichem Namen;
- HTTP-zu-HTTPS-Schleife;
- falscher Port;
- falsches Schema;
- Proxyheader;
- Hostheader;
- Backendstatus;
- Anwendungspfad.

Ein HTTP-Fehlercode beweist, dass mindestens eine TLS-Verbindung bereits erfolgreich aufgebaut wurde.

7.14.42 Netzwerkaufzeichnung einsetzen

Eine Netzwerkaufzeichnung ist sinnvoll, wenn unklar bleibt:

- ob TCP aufgebaut wird;
- wer die Verbindung beendet;
- ob ein Reset erscheint;
- welches Ziel angesprochen wird;
- ob Retransmissions auftreten;
- ob ein TLS ClientHello gesendet wird;

- welche TLS-Versionen angeboten werden;
- ob SNI übermittelt wird;
- ob ein TLS Alert erscheint;
- ob der Server antwortet;
- ob mehrere Verbindungsversuche erfolgen;
- ob ein Proxy beteiligt ist.

Bei TLS 1.3 sind größere Teile des Handshakes verschlüsselt als bei älteren Versionen. Mit Encrypted ClientHello kann auch der bisher sichtbare Servername geschützt werden.

Für die Entschlüsselung moderner TLS-Sitzungen werden in der Regel autorisiert erfasste Sitzungsschlüssel benötigt. Ein Server-Private-Key allein reicht insbesondere bei modernen Schlüsselaustauschverfahren nicht zur nachträglichen Entschlüsselung.

TLS-Schlüsselprotokolle und entschlüsselte Aufzeichnungen enthalten hochsensible Daten und müssen besonders geschützt, zeitlich begrenzt und anschließend sicher entfernt werden.

7.14.43 Hypothese und Gegenbeweis formulieren

Beispiel:

```
Hypothese:
Der Reverse Proxy sendet beim Verbindungsaufbau zum Backend
keinen passenden SNI-Namen und erhält deshalb das
Default-Zertifikat eines anderen virtuellen Hosts.

Erwarteter Befund:
Der OpenSSL-Test gegen die Backend-IP liefert ohne SNI
ein anderes Zertifikat als mit dem vorgesehenen SNI-Namen.

Gegenbeweis:
Mit und ohne SNI wird dasselbe korrekte Zertifikat geliefert
und die Backendverbindung funktioniert.

Testmethode:
OpenSSL-Verbindungen mit identischer Ziel-IP und
unterschiedlicher SNI-Übermittlung vergleichen.

Erfolgskriterium:
Das abweichende Zertifikat und der fehlerhafte
Proxy-SNI-Pfad sind reproduzierbar nachgewiesen.
```

Eine belastbare Hypothese enthält:

- betroffene Verbindungsebene;
- vermutete Ursache;
- erwarteten messbaren Befund;
- Gegenbeweis;
- Testmethode;
- Risiko;
- Erfolgskriterium.

7.14.44 Kontrollierte Maßnahmen

Maßnahme	Voraussetzung	Risiko
Zertifikat erneuern	Ablauf oder falsche Gültigkeit bestätigt	falsche Bindung oder unvollständige Verteilung
vollständige Kette konfigurieren	fehlendes Intermediate bestätigt	falsche Reihenfolge oder falsche CA
Zertifikatsbindung korrigieren	falscher Listener oder virtueller Host bestätigt	andere Site kann betroffen sein
SNI korrigieren	falsche Zertifikatsauswahl nachgewiesen	weitere virtuelle Hosts betroffen
Backend-SNI korrigieren	Proxy sendet falschen Namen	Backendrouting ändert sich
Truststore ergänzen	vorgesehene CA eindeutig bestätigt	zusätzliche CA erhält Vertrauensstatus
Systemzeit korrigieren	Zeitabweichung bestätigt	Kerberos, Tokens und Logs betroffen
TLS-Policy korrigieren	fehlende gemeinsame sichere Version bestätigt	ältere Clients können ausfallen
Cipher-Policy korrigieren	fehlende sichere Schnittmenge bestätigt	Kompatibilität und Sicherheit betroffen
Clientzertifikat erneuern	mTLS-Zertifikat fehlerhaft	Identitätszuordnung kann ausfallen
Proxyregel korrigieren	falscher Proxyweg bestätigt	weitere Anwendungen betroffen
TLS-Inspection-Ausnahme	Inspection als Ursache bestätigt und freigegeben	Sicherheitskontrolle wird verändert
Reverse-Proxy-Backend korrigieren	Backend-TLS-Fehler bestätigt	Dienstunterbrechung möglich
Dienst kontrolliert neu laden	neue Konfiguration geprüft	kurze Unterbrechung oder Ladefehler
Load-Balancer-Nodes synchronisieren	abweichende Zertifikate bestätigt	mehrere produktive Nodes betroffen

Vor der Maßnahme sind zu dokumentieren:

- Ausgangszustand;
- aktives Zertifikat;

- SHA-256-Fingerprint;
- Zertifikatsbindung;
- Servername;
- SNI;
- TLS-Version;
- Cipher;
- Truststore;
- Proxyweg;
- Risiko;
- Rückweg;
- Erfolgskriterium.

7.14.45 Nicht zulässige Schnelllösungen

```
Zertifikatsprüfung dauerhaft deaktivieren
curl -k in produktive Skripte übernehmen
verify=false in Anwendungen eintragen
NODE_TLS_REJECT_UNAUTHORIZED=0 setzen
Java-Hostnameprüfung deaktivieren
beliebige Root-CA importieren
selbst signiertes Zertifikat ungeprüft vertrauen
private Schlüssel weitergeben
private Schlüssel in Containerimages einbauen
alte TLS-Versionen pauschal aktivieren
unsichere Cipher dauerhaft aktivieren
Sperrprüfung ohne Risikoanalyse abschalten
Truststore vollständig ersetzen
alle Zertifikate aus Stores löschen
Schannel-Registrywerte nach ungeprüfter Anleitung ändern
Reverse Proxy ohne Konfigurationstest neu starten
Zertifikat nur auf einem Load-Balancer-Node austauschen
TLS-Inspection heimlich umgehen
```

Eine funktionierende Verbindung ohne Zertifikatsprüfung ist kein Nachweis einer sicheren oder korrekten Lösung.

7.14.46 Vollständiger Diagnoseablauf

1. **Exakte Fehlermeldung aufnehmen**
   Wortlaut, Anwendung, Client und Uhrzeit dokumentieren.

2. **Vollständige URL bestimmen**
   Schema, FQDN, Port und Pfad erfassen.
3. **Fehlerumfang bestimmen**
   Einzelnen Client, Anwendung, Standort oder alle Benutzer unterscheiden.
4. **Letzte Änderungen erfassen**
   Zertifikatswechsel, Update, Proxy-, DNS- oder Firewalländerung prüfen.
5. **Systemzeit prüfen**
   Datum, UTC, Zeitzone und Synchronisation bestätigen.
6. **DNS-Auflösung prüfen**
   A, AAAA und CNAME auswerten.
7. **Tatsächliche Zieladresse bestimmen**
   IPv4, IPv6, Load Balancer oder Proxy identifizieren.
8. **TCP-Port prüfen**
   Timeout, Ablehnung und erfolgreichen Handshake unterscheiden.
9. **Proxyweg bestimmen**
   Direktverbindung, expliziten Proxy, PAC oder TLS-Inspection unterscheiden.
10. **Normalen HTTPS-Test durchführen**
    Zertifikatsprüfung nicht deaktivieren.
11. **TLS und HTTP trennen**
    Handshakefehler und HTTP-Statuscode unterscheiden.
12. **OpenSSL-Handshake durchführen**
    SNI und Hostnamenprüfung ausdrücklich angeben.
13. **TLS-Version dokumentieren**
    Ausgehandelte Version erfassen.
14. **Cipher dokumentieren**
    Ausgehandelte Cipher Suite erfassen.
15. **ALPN dokumentieren**
    HTTP/1.1, HTTP/2 oder anderes Protokoll bestimmen.
16. **Serverzertifikat dokumentieren**
    Subject, Issuer, Seriennummer und Fingerprint erfassen.
17. **SAN prüfen**
    Aufgerufenen FQDN mit Zertifikat vergleichen.
18. **Gültigkeitszeitraum prüfen**
    Server- und Intermediate-Zertifikate berücksichtigen.
19. **Zertifikatskette prüfen**
    Fehlende oder falsche Intermediate-CA bestimmen.
20. **Root-Vertrauen prüfen**
    Tatsächlichen Truststore der Anwendung untersuchen.
21. **Sperrprüfung prüfen**
    CRL, OCSP, AIA und Netzwerkzugriff berücksichtigen.
22. **SNI prüfen**
    Test mit und ohne vorgesehenen Servernamen vergleichen.
23. **Einzelne Zieladressen prüfen**
    Load-Balancer-Nodes mit `--resolve` vergleichen.
24. **IPv4 und IPv6 vergleichen**
    Unterschiedliche Pfade und Zertifikate bestimmen.

25. **mTLS prüfen**
    Clientzertifikat, Schlüssel, Kette und Serververtrauen untersuchen.
26. **TLS-Inspection prüfen**
    Zertifikatsaussteller und Fingerprint zwischen Netzen vergleichen.
27. **Anwendungsspezifischen Truststore prüfen**
    Betriebssystem, Browser, Java und Container unterscheiden.
28. **Serverbindung prüfen**
    Listener, virtuellen Host, Zertifikat und Schlüssel zuordnen.
29. **Zertifikat und Schlüssel vergleichen**
    Öffentliche Schlüssel sicher hashen.
30. **Reverse Proxy und Backend trennen**
    Frontend- und Backend-TLS einzeln testen.
31. **Serverprotokolle auswerten**
    Handshake, Zertifikat, mTLS und Backendfehler korrelieren.
32. **Bei Bedarf Netzwerkaufzeichnung durchführen**
    Nur autorisiert, begrenzt und datenschutzkonform.
33. **Hypothese und Gegenbeweis formulieren**
    Ursache vor der Änderung messbar festlegen.
34. **Eine kontrollierte Maßnahme durchführen**
    Risiko, Rückweg und Freigabe beachten.
35. **Konfiguration vor Reload prüfen**
    Beispielsweise `nginx -t` oder `apachectl configtest`.
36. **Identischen TLS-Test wiederholen**
    Version, Cipher, Zertifikat und Verify-Ergebnis vergleichen.
37. **HTTP-Antwort prüfen**
    Statuscode, Redirect und Header auswerten.
38. **Ursprüngliche Anwendung testen**
    Nicht nur `curl` oder OpenSSL verifizieren.
39. **Weitere Clients und Netze prüfen**
    Repräsentative Systeme vergleichen.
40. **Temporäre Diagnoseänderungen zurücknehmen**
    Logging, Capture, Testzertifikate und Ausnahmen entfernen.
41. **Ursache dokumentieren**
    Fehlerhafte Ebene und technischer Nachweis festhalten.
42. **Prävention festlegen**
    Monitoring, automatische Erneuerung und Konfigurationstest verbessern.

7.14.47 Befundmatrix

Befund	Mögliche Einordnung	Nächster Nachweis
DNS schlägt fehl	Namensauflösung	direkte A- und AAAA-Abfrage
TCP-Port `443` nicht erreichbar	Routing, Firewall oder Listener	Netzwerkpfad und Serverseite
TCP funktioniert, kein ServerHello	TLS-Policy, Proxy oder Server	Handshake und Serverlogs

Befund	Mögliche Einordnung	Nächster Nachweis
`curl -k` funktioniert, normaler Test nicht	Zertifikatsvalidierung	Name, Kette, Zeit und Vertrauen
Hostname mismatch	falsches Zertifikat oder falscher Name	SAN und SNI prüfen
Zertifikat abgelaufen	Erneuerung oder falscher Node	alle Zieladressen vergleichen
Zertifikat noch nicht gültig	Systemzeit oder falsches Zertifikat	Zeit und Gültigkeitsbeginn
unbekannte CA	Root fehlt oder TLS-Inspection	Issuer und Truststore
lokale CA im Browser vertraut, Java nicht	separater Java-Truststore	`keytool -list`
Kette unvollständig	Intermediate fehlt	`-showcerts` und `openssl verify`
nur ein Client betroffen	lokaler Truststore, Zeit oder Proxy	anderen Client vergleichen
nur ein Load-Balancer-Node betroffen	Zertifikatsverteilung unvollständig	`curl --resolve`
ohne SNI falsches Zertifikat	namensbasierter virtueller Host	Test mit korrektem SNI
mit SNI ebenfalls falsches Zertifikat	falsche Bindung oder Serverregel	Listenerkonfiguration
TLS 1.2 funktioniert, TLS 1.3 nicht	TLS-1.3-Pfad oder Intermediär	Proxy und Serverpolicy
TLS 1.3 funktioniert, alter Client nicht	veralteter Client	Clientbibliothek und TLS-Version
keine gemeinsame Cipher	Policy- oder Algorithmuskonflikt	Client- und Serverlisten vergleichen
Browser funktioniert, Dienst nicht	Proxy, Store oder Systemkonto	Dienstkontext prüfen
Host funktioniert, Container nicht	CA-Bundle, Proxy oder DNS	Test im Container
Frontend-TLS funktioniert, HTTP 502	Backend nicht erreichbar oder Backend-TLS	Proxy- und Backendlogs
HTTP 504	Backendtimeout	Backendpfad und Antwortzeit
Clientzertifikat erforderlich	mTLS	Zertifikat und Schlüssel prüfen
`unknown ca` am Server bei mTLS	Server vertraut Client-CA nicht	Server-Truststore
nur hinter Security-Proxy fehlerhaft	TLS-Inspection	Zertifikatsaussteller vergleichen
OCSP- oder CRL-Fehler	Sperrprüfungsziel nicht erreichbar	AIA/CDP und Proxy prüfen
neuer Fingerprint nur auf manchen Nodes	unsynchronisierte Zertifikate	alle Nodes einzeln prüfen
HTTP 200, Anwendung fehlerhaft	oberhalb von TLS	Anwendung, Authentifizierung und Inhalt

---

7.14.48 Typische Diagnosefehler

- HTTPS sofort als reines Zertifikatsproblem behandeln.
- DNS und TCP nicht zuerst prüfen.
- Ping als Port- oder TLS-Nachweis verwenden.

- Direkten IP-Aufruf als vollständigen HTTPS-Test verwenden.
- SNI beim OpenSSL-Test vergessen.
- Hostnamenprüfung nicht ausdrücklich durchführen.
- Nur das Serverzertifikat, aber nicht die Kette prüfen.
- Nur das Ablaufdatum des Leaf-Zertifikats prüfen.
- Intermediate- und Root-Zertifikat verwechseln.
- Root-CA unnötig mit der Serverkette ausliefern.
- Subject prüfen, aber SAN ignorieren.
- Wildcard-Zertifikat zu weit auslegen.
- Systemzeit nicht prüfen.
- `curl -k` als Lösung verwenden.
- Browser und Anwendung denselben Truststore unterstellen.
- Benutzer- und Computerzertifikatsspeicher verwechseln.
- Java-Truststore ignorieren.
- Container-CA-Bundle ignorieren.
- Proxykontext des Dienstkontos nicht prüfen.
- TLS-Inspection nicht berücksichtigen.
- IPv4 und IPv6 nicht getrennt testen.
- Nur einen Load-Balancer-Node prüfen.
- Frontend- und Backend-TLS verwechseln.
- mTLS nicht von normalem TLS unterscheiden.
- Clientzertifikat ohne privaten Schlüssel verwenden.
- Zertifikat und privaten Schlüssel nicht zuordnen.
- Zertifikatsdatei ersetzen, aber Dienst nicht kontrolliert neu laden.
- Dienst neu starten, ohne vorher die Konfiguration zu testen.
- TLS-Version und Cipher Suite gleichsetzen.
- HTTP-Statuscode als TLS-Fehler behandeln.
- Sperrprüfung ohne Ursachenanalyse deaktivieren.
- Private Schlüssel in Diagnoseausgaben aufnehmen.
- Verbose Ausgaben mit Tokens oder Cookies ungeschützt weitergeben.
- Mehrere TLS-Einstellungen gleichzeitig verändern.
- Nur das Diagnosewerkzeug, aber nicht die Anwendung verifizieren.

7.14.49 Verifikation

Nach einer Maßnahme müssen mindestens folgende Punkte geprüft werden:

- DNS liefert die vorgesehenen Adressen;
- IPv4 und IPv6 funktionieren wie vorgesehen;
- TCP-Port ist erreichbar;
- richtiger Proxyweg wird verwendet;
- TLS-Handshake wird abgeschlossen;
- vorgesehene TLS-Version wird ausgehandelt;
- sichere gemeinsame Cipher Suite wird verwendet;
- SNI enthält den richtigen Namen;

- ALPN liefert das vorgesehene Protokoll;
- Serverzertifikat besitzt den richtigen SAN;
- Zertifikat ist aktuell gültig;
- vollständige Zertifikatskette wird geliefert;
- Client vertraut der vorgesehenen Root-CA;
- Sperrprüfung funktioniert;
- bei mTLS wird das richtige Clientzertifikat verwendet;
- privater Schlüssel ist vorhanden und geschützt;
- Zertifikat und Schlüssel gehören zusammen;
- alle Load-Balancer-Nodes liefern dasselbe Zertifikat;
- Reverse Proxy vertraut dem Backend;
- Backend-SNI ist korrekt;
- keine neuen Schannel-, CAPI2- oder Serverfehler entstehen;
- HTTP-Antwortcode entspricht dem Sollzustand;
- Redirects führen zum vorgesehenen Host;
- ursprüngliche Anwendung funktioniert;
- repräsentative Clients funktionieren;
- Container und Dienste funktionieren in ihrem tatsächlichen Kontext;
- temporäre Ausnahmen wurden entfernt;
- Fingerprint, Ablaufdatum und Erneuerungsweg wurden dokumentiert.

Ein erfolgreicher Test mit deaktivierter Zertifikatsprüfung ist keine gültige Verifikation.

7.14.50 Prävention und Monitoring

Zu überwachen sind:

- Ablaufdatum des Serverzertifikats;
- Ablaufdatum der Intermediate-CA;
- Ablaufdatum der Root-CA;
- Zertifikatsname und SAN;
- SHA-256-Fingerprint;
- vollständige Zertifikatskette;
- Erreichbarkeit des HTTPS-Endpunkts;
- TLS-Handshake;
- TLS-Version;
- Zertifikatsaussteller;
- OCSP- und CRL-Erreichbarkeit;
- automatische Zertifikatserneuerung;
- Verteilung auf alle Nodes;
- Reload des betroffenen Dienstes;
- Frontend- und Backend-TLS;
- mTLS-Clientzertifikate;
- Proxy- und Inspection-Zertifikate;
- Systemzeit;

- Konfigurationssyntax;
- HTTP-Statuscode und Antwortzeit.

Empfohlener Zertifikatsprozess:

```
Erneuerung anfordern
→ Zertifikat und Kette validieren
→ Zertifikat und Schlüssel zuordnen
→ Testsystem oder einzelnen Node aktualisieren
→ Konfiguration prüfen
→ Dienst kontrolliert neu laden
→ externen TLS-Test durchführen
→ alle Nodes aktualisieren
→ Monitoring und Fingerprint prüfen
→ altes Zertifikat kontrolliert entfernen
```

Die Erneuerung gilt erst als erfolgreich, wenn der produktive Listener das neue Zertifikat tatsächlich ausliefert.

7.14.51 Dokumentationsvorlage

```
Störung:
<exakte Fehlermeldung>

Zeitpunkt:
<Datum und Uhrzeit>

Betroffener Client:
<Hostname, IP-Adresse und Betriebssystem>

Betroffene Anwendung:
<Browser, Dienst, Container oder Anwendung>

URL:
<vollständige URL>

FQDN:
<Servername>
```

Port:

<Port>

Aufgelöste Adressen:

<IPv4 und IPv6>

Verwendete Zieladresse:

<IP-Adresse>

Proxyweg:

<direkt, Proxy, PAC oder TLS-Inspection>

TCP-Ergebnis:

<Ergebnis>

TLS-Version:

<ausgehandelte Version>

Cipher Suite:

<ausgehandelte Cipher>

SNI:

<übermittelter Name>

ALPN:

<ausgehandeltes Protokoll>

Zertifikat-Subject:

<Wert>

Subject Alternative Names:

<Werte>

Issuer:

<Wert>

Seriennummer:

<Wert>

SHA-256-Fingerprint:

<Wert>

Gültigkeit:

<Not Before und Not After>

Zertifikatskette:

<Leaf, Intermediate und Root>

Truststore:

<Betriebssystem, Browser, Java oder Anwendung>

Sperrprüfung:

<CRL- und OCSP-Befund>

mTLS:

<erforderlich, Clientzertifikat und Ergebnis>

Reverse-Proxy-Pfad:

<Frontend und Backend>

HTTP-Ergebnis:

<Statuscode, Redirect und Antwortzeit>

Nachgewiesene Ursache:

<technischer Befund>

Gegenbeweis ausgeschlossen durch:

<Test und Ergebnis>

Durchgeführte Maßnahme:

<genaue Änderung>

Risiko und Rückweg:

<Beschreibung>

Verifikation:

<identischer TLS-Test und Anwendungstest>

Prävention:

<Monitoring, Erneuerung oder Standardisierung>

7.14.52 Checkliste

- [ ] exakte Fehlermeldung dokumentiert
- [ ] Datum und Uhrzeit erfasst
- [ ] vollständige URL erfasst
- [ ] FQDN erfasst
- [ ] Port erfasst
- [ ] betroffene Anwendung erfasst
- [ ] Clientbetriebssystem erfasst
- [ ] Systemzeit geprüft
- [ ] Zeitzone geprüft
- [ ] DNS-A-Eintrag geprüft
- [ ] DNS-AAAA-Eintrag geprüft
- [ ] CNAME-Kette geprüft
- [ ] tatsächliche Zieladresse bestimmt
- [ ] IPv4 getestet
- [ ] IPv6 getestet
- [ ] TCP-Port geprüft
- [ ] Timeout und Ablehnung unterschieden
- [ ] Proxykonfiguration geprüft
- [ ] PAC-Datei berücksichtigt
- [ ] `NO_PROXY` berücksichtigt
- [ ] TLS-Inspection berücksichtigt
- [ ] normalen HTTPS-Test durchgeführt
- [ ] Zertifikatsprüfung nicht dauerhaft deaktiviert
- [ ] OpenSSL-Handshake durchgeführt
- [ ] SNI ausdrücklich angegeben
- [ ] Hostnamenprüfung durchgeführt
- [ ] TLS-Version dokumentiert
- [ ] Cipher Suite dokumentiert
- [ ] ALPN dokumentiert
- [ ] Serverzertifikat erfasst

- [ ] Subject geprüft
- [ ] SAN geprüft
- [ ] Wildcard-Gültigkeit geprüft
- [ ] Issuer geprüft
- [ ] Seriennummer dokumentiert
- [ ] SHA-256-Fingerprint dokumentiert
- [ ] Gültigkeitsbeginn geprüft
- [ ] Ablaufdatum geprüft
- [ ] Serverzertifikatskette geprüft
- [ ] Intermediate-CA geprüft
- [ ] Root-CA geprüft
- [ ] tatsächlichen Truststore bestimmt
- [ ] CRL geprüft
- [ ] OCSP geprüft
- [ ] AIA-Erreichbarkeit geprüft
- [ ] Sperrprüfungsrichtlinie berücksichtigt
- [ ] einzelne Load-Balancer-Nodes verglichen
- [ ] Test mit `curl --resolve` durchgeführt
- [ ] mTLS-Anforderung geprüft
- [ ] Clientzertifikat geprüft
- [ ] Client-CA-Vertrauen geprüft
- [ ] privater Schlüssel vorhanden
- [ ] Zertifikat und Schlüssel zugeordnet
- [ ] Windows-Schannel-Ereignisse geprüft
- [ ] CAPI2 bei Bedarf geprüft
- [ ] Windows-Zertifikatsspeicher geprüft
- [ ] Linux- oder macOS-Truststore geprüft
- [ ] Java-Truststore bei Bedarf geprüft
- [ ] Container-Truststore bei Bedarf geprüft
- [ ] Reverse-Proxy-Frontend geprüft
- [ ] Reverse-Proxy-Backend geprüft
- [ ] Backend-SNI geprüft
- [ ] Webserverkonfiguration getestet
- [ ] Serverprotokolle ausgewertet

- [ ] HTTP-Statuscode dokumentiert
- [ ] Redirects geprüft
- [ ] Hypothese formuliert
- [ ] Gegenbeweis festgelegt
- [ ] Risiko und Rückweg dokumentiert
- [ ] nur eine kontrollierte Maßnahme durchgeführt
- [ ] identischen TLS-Test wiederholt
- [ ] ursprüngliche Anwendung getestet
- [ ] weitere Clients geprüft
- [ ] alle Load-Balancer-Nodes geprüft
- [ ] temporäre Ausnahmen entfernt
- [ ] sensible Diagnoseinformationen geschützt
- [ ] Ursache dokumentiert
- [ ] Präventionsmaßnahme festgelegt

7.14.53 Schnellreferenz

Aufgabe	Befehl
Windows-DNS	`Resolve-DnsName app.example.test`
Linux/macOS-DNS	`dig app.example.test A`
Windows-TCP-Test	`Test-NetConnection app.example.test -Port 443`
HTTPS-Diagnose	`curl -v https://app.example.test/`
IPv4-Test	`curl -4 -v https://app.example.test/`
IPv6-Test	`curl -6 -v https://app.example.test/`
bestimmte IP mit richtigem Hostnamen	`curl -v --resolve app.example.test:443:192.0.2.25 https://app.example.test/`
TLS-Handshake	`openssl s_client -connect app.example.test:443 -servername app.example.test </dev/null`
Zertifikatskette anzeigen	`openssl s_client -connect app.example.test:443 -servername app.example.test -showcerts </dev/null`
Hostname verifizieren	`openssl s_client -connect app.example.test:443 -servername app.example.test -verify_hostname app.example.test </dev/null`
TLS 1.2 testen	`openssl s_client -connect app.example.test:443 -servername app.example.test -tls1_2 </dev/null`
TLS 1.3 testen	`openssl s_client -connect app.example.test:443 -servername app.example.test -tls1_3 </dev/null`
ALPN testen	`openssl s_client -connect app.example.test:443 -servername app.example.test -alpn h2,http/1.1 </dev/null`

Aufgabe	Befehl
OCSP-Stapling anfordern	`openssl s_client -connect app.example.test:443 -servername app.example.test -status </dev/null`
Zertifikatsdaten	`openssl x509 -in server.pem -noout -subject -issuer -serial -dates`
SAN anzeigen	`openssl x509 -in server.pem -noout -ext subjectAltName`
Zertifikatskette lokal prüfen	`openssl verify -purpose sslserver -CAfile root-ca.pem -untrusted intermediates.pem server.pem`
Windows-Zertifikatsprüfung	`certutil -verify -urlfetch server.cer`
Windows-Zertifikatsspeicher	`Get-ChildItem Cert:\LocalMachine`
Windows-Cipher Suites	`Get-TlsCipherSuite`
WinHTTP-Proxy	`netsh winhttp show proxy`
Schannel-Ereignisse	`Get-WinEvent -FilterHashtable @{LogName="System"; ProviderName="Schannel"}`
Java-Truststore	`keytool -list -cacerts`
Docker-Test	`docker exec <Container> curl -v https://app.example.test/`
Nginx-Konfigurationstest	`nginx -t`
Apache-Konfigurationstest	`apachectl configtest`
IIS-HTTPS-Bindungen	`Get-WebBinding -Protocol https`

7.14.54 Quellen

Offizielle Microsoft-Dokumentation

- Microsoft Learn – Overview of TLS and Schannel SSP
- Microsoft Learn – TLS registry settings
- Microsoft Learn – Troubleshooting SSL-related IIS server-certificate issues
- Microsoft Learn – Certificate Provider
- Microsoft Learn – Certutil
- Microsoft Learn – Valid root CA certificates are untrusted
- Microsoft Learn – Diagnostic logging for PKI and SSL certificate issues

Offizielle OpenSSL-Dokumentation

- OpenSSL – s_client
- OpenSSL – x509
- OpenSSL – verify
- OpenSSL – Certificate verification options

- OpenSSL – Command overview

Offizielle curl-Dokumentation

- curl – Command-line manual
- curl – TLS certificate verification
- curl – SSL ciphers
- curl – Tutorial
- curl – HTTP scripting

Offizielle Webserver-Dokumentation

- Nginx – Configuring HTTPS servers
- Nginx – HTTP SSL module
- Nginx – HTTP proxy module
- Apache HTTP Server – Configuration files and configtest
- Apache HTTP Server – apachectl

Offizielle Java-Dokumentation

- Oracle – keytool
- Oracle – JSSE Reference Guide

Offizielle Wireshark-Dokumentation

- Wireshark – User’s Guide
- Wireshark – TShark manual
- Wireshark – Exporting TLS session keys

Standards

- RFC 5280 – Internet X.509 Public Key Infrastructure Certificate and CRL Profile
- RFC 6066 – TLS Extensions: Server Name Indication
- RFC 7301 – TLS Application-Layer Protocol Negotiation
- RFC 8446 – TLS 1.3
- RFC 9110 – HTTP Semantics
- RFC 9113 – HTTP/2
- RFC 9325 – Recommendations for Secure Use of TLS and DTLS
- RFC 9525 – Service Identity in TLS

Für diese Seite wurden keine Community-Berichte, Hersteller-Social-Media-Aussagen oder eigenen Laborergebnisse als Nachweis verwendet.