2.10 OpenSSL – TLS-Verbindungen, Zertifikate und Zertifikatsketten prüfen

Ziel dieser Seite

OpenSSL ist eine Sammlung von Werkzeugen und Bibliotheken für Kryptografie, Zertifikate und TLS. In der Fehleranalyse wird der Befehl `openssl` unter anderem verwendet, um:

- TLS-Verbindungen direkt zu einem Server aufzubauen;
- TLS-Version und Cipher Suite zu erkennen;
- Server Name Indication, kurz SNI, gezielt zu testen;
- Zertifikate eines Servers anzuzeigen;
- Zertifikatsketten zu untersuchen;
- Zertifikatsnamen und Subject Alternative Names zu prüfen;
- Gültigkeitszeiträume zu kontrollieren;
- interne Zertifizierungsstellen einzubeziehen;
- STARTTLS bei SMTP, IMAP, POP3 und LDAP zu testen;
- Clientzertifikate für mTLS-Verbindungen zu verwenden;
- Zertifikate, private Schlüssel und CSRs zu untersuchen;
- zu prüfen, ob Zertifikat und privater Schlüssel zusammengehören;
- PEM-, DER- und PKCS#12-Dateien zu unterscheiden;
- Fehler zwischen TCP, TLS, Zertifikat und Anwendung einzugrenzen.

> OpenSSL kann private Schlüssel und andere hochsensible kryptografische Daten verarbeiten. Befehle mit privaten Schlüsseln dürfen nur auf autorisierten Systemen und in geschützten Verzeichnissen ausgeführt werden.

Kennzeichnungen

Kennzeichnung	Bedeutung
`[RO]`	Nur lesende lokale Prüfung
`[TEST]`	Aktiver Netzwerk- oder TLS-Test
`[PRIV]`	Erhöhte Berechtigungen können erforderlich sein
`[FILE]`	Befehl liest oder erstellt eine Datei
`[SENS]`	Ausgabe oder Datei enthält möglicherweise sensible Informationen
`[CHANGE]`	Befehl erstellt oder verändert Dateien

Kennzeichnung	Bedeutung
[DISRUPT]	Test kann Dienst, Sitzung oder produktive Daten beeinflussen

---

1. Wie wird die OpenSSL-Version geprüft?

Versions- und Buildinformationen anzeigen

Aufgabe	Windows	Linux	macOS
Programmpfad	[RO] Get-Command openssl.exe	[RO] command -v openssl	[RO] command -v openssl
Version	[RO] openssl.exe version	[RO] openssl version	[RO] openssl version
Ausführliche Version	[RO] openssl.exe version -a	[RO] openssl version -a	[RO] openssl version -a
Befehlsübersicht	[RO] openssl.exe help	[RO] openssl help	[RO] openssl help
Hilfe zu s_client	[RO] openssl.exe s_client -help	[RO] openssl s_client -help	[RO] openssl s_client -help

Ausführliche Buildinformationen

```
[RO] openssl version -a
```

Die Ausgabe kann unter anderem enthalten:

- OpenSSL- oder LibreSSL-Version;
- Erstellungsdatum;
- Zielplattform;
- Konfigurationsverzeichnis;
- Provider- und Modulpfade;
- Compileroptionen;
- CPU-Funktionen.

Betriebssystemspezifische Hinweise

Betriebssystem	Hinweis
Windows	OpenSSL ist kein allgemeines Windows-Bordmittel und muss aus einer vertrauenswürdigen Quelle stammen oder mit einer Anwendung mitgeliefert werden
Linux	Version und Konfiguration hängen von Distribution und Paketstand ab

Betriebssystem	Hinweis
macOS	Der Befehl `openssl` kann je nach Installation LibreSSL oder eine separat installierte OpenSSL-Version aufrufen
Container	Der Container kann eine andere OpenSSL-Version als der Host verwenden

OpenSSL und LibreSSL sind nicht vollständig optionskompatibel. Vor der Übernahme eines Befehls muss deshalb geprüft werden:

```
openssl version
openssl s_client -help
```

---

2. Wie wird eine grundlegende TLS-Verbindung getestet?

Verbindung mit `s_client` anzeigen

```
[TEST][SENS] openssl s_client -connect example.com:443
```

Der Befehl baut eine TCP-Verbindung zu Port 443 auf und startet anschließend einen TLS-Handshake.

Mit ausdrücklich gesetztem SNI

```
[TEST][SENS] openssl s_client -connect example.com:443 -servername example.com
```

`-servername` setzt die TLS-Erweiterung Server Name Indication.

Kompakte Ausgabe, sofern von der installierten Version unterstützt

```
[TEST][SENS] openssl s_client -connect example.com:443 -servername example.com -brief
```

Verbindung beenden

Nach dem TLS-Handshake wartet `s_client` häufig auf weitere Eingaben. Die Sitzung kann normalerweise mit folgender Tastenkombination beendet werden:

```
[Strg] + [C]
```

Neuere OpenSSL-Versionen unterstützen einen nicht interaktiven Modus:

```
[TEST][SENS] openssl s_client -connect example.com:443 -servername example.com -no-interactive
```

Vor Verwendung muss geprüft werden:

```
[RO] openssl s_client -help
```

Wichtige Ausgabeinformationen

- Ziel-IP und Zielport;
- Serverzertifikat;
- Zertifikatskette;
- TLS-Version;
- Cipher Suite;
- Schlüsselaustausch;
- Signaturalgorithmus;
- ALPN-Ergebnis;
- Zertifikatsprüfergebnis;
- Sitzungsinformationen.

---

3. Warum muss SNI ausdrücklich berücksichtigt werden?

SNI und virtuelle Hosts erklären

Mehrere HTTPS-Dienste können dieselbe IP-Adresse und denselben Port verwenden:

```
192.0.2.20:443
├── wiki.example.com
├── mail.example.com
└── api.example.com
```

Der Client teilt dem Server während des TLS-Handshakes mit, welchen Hostnamen er erreichen möchte. Das geschieht über SNI.

Wiki prüfen

```
[TEST][SENS] openssl s_client -connect 192.0.2.20:443 -servername wiki.example.com
```

API prüfen

```
[TEST][SENS] openssl s_client -connect 192.0.2.20:443 -servername api.example.com
```

Der Server kann abhängig vom SNI-Namen unterschiedliche Zertifikate zurückgeben.

Ohne SNI

```
[TEST][SENS] openssl s_client -connect 192.0.2.20:443 -noservername
```

Mögliche Folgen:

- Standardzertifikat des Reverse Proxys;
- Zertifikat eines anderen virtuellen Hosts;
- TLS-Abbruch;
- falsche Anwendung;
- irreführender Zertifikatsnamensfehler.

> Ein Test gegen eine IP-Adresse ohne passenden SNI-Namen bildet den Zugriff eines normalen Browsers auf einen DNS-Namen nicht zuverlässig nach.

---

4. Wie wird ein TLS-Server über eine bestimmte IP-Adresse getestet?

Ziel-IP und Hostname getrennt prüfen

Angenommen:

```
Hostname: wiki.example.com
Test-IP: 192.0.2.20
Port: 443
```

Direkter TLS-Test gegen diese IP mit richtigem SNI

```
[TEST][SENS] openssl s_client -connect 192.0.2.20:443 -servername wiki.example.com
```

Dadurch wird:

- die DNS-Auflösung für die Verbindung umgangen;
- die Verbindung direkt zu `192.0.2.20` aufgebaut;
- SNI auf `wiki.example.com` gesetzt.

Für eine vollständige Hostnamenprüfung sollte zusätzlich verwendet werden:

```
[TEST][SENS] openssl s_client -connect 192.0.2.20:443 -servername wiki.example.com -verify_hostname wiki.example.com -verify_return_error
```

Vergleich mit curl

```
[TEST][SENS] curl --resolve wiki.example.com:443:192.0.2.20 https://wiki.example.com/
```

OpenSSL untersucht hauptsächlich TLS und Zertifikate. curl führt zusätzlich eine HTTP-Anfrage aus.

---

5. Was bedeutet eine erfolgreiche TLS-Verbindung?

Handshake, Zertifikat und Anwendung unterscheiden

Ein erfolgreicher TLS-Handshake beweist zunächst:

- TCP-Verbindung wurde hergestellt;
- Client und Server fanden eine gemeinsame TLS-Version;
- Client und Server fanden geeignete kryptografische Parameter;
- der Server konnte ein Zertifikat oder anderes benötigtes Material bereitstellen;
- verschlüsselte Kommunikation konnte grundsätzlich beginnen.

Er beweist nicht automatisch:

- dass der Zertifikatsname richtig ist;
- dass die Zertifikatskette vertrauenswürdig ist;
- dass das Zertifikat nicht abgelaufen ist;
- dass die Anwendung funktioniert;
- dass HTTP einen erfolgreichen Statuscode liefert;

- dass eine Benutzeranmeldung möglich ist;
- dass eine Sperrprüfung erfolgreich war.

Diagnoseebenen

```
DNS
 ↓
TCP
 ↓
TLS-Handshake
 ↓
Zertifikatsprüfung
 ↓
HTTP oder anderes Anwendungsprotokoll
 ↓
Anwendung
```

Für HTTPS sollte nach dem OpenSSL-Test zusätzlich eine HTTP-Prüfung erfolgen:

```
[TEST][SENS] curl -v https://wiki.example.com/
```

---

6. Warum kann `s_client` trotz Zertifikatsfehler eine Verbindung fortsetzen?

Wichtige Besonderheit der Zertifikatsprüfung anzeigen

`openssl s_client` ist ein Diagnosewerkzeug. Standardmäßig kann es nach Zertifikatsprüffehlern fortfahren, damit weitere Probleme sichtbar werden.

Eine Ausgabe wie:

```
verify error:num=20:unable to get local issuer certificate
```

bedeutet deshalb nicht zwingend, dass der TLS-Handshake sofort abgebrochen wird.

Bei Zertifikatsfehlern wirklich abbrechen

```
[TEST][SENS] openssl s_client -connect example.com:443 -servername example.com -verify_return_error
```

Zusätzlich Hostnamen prüfen

```
[TEST][SENS] openssl s_client -connect example.com:443 -servername example.com -verify_hostname example.com -verify_return_error
```

Erwartetes positives Ergebnis

```
Verification: OK
```

oder in der ausführlichen Ausgabe:

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

Wichtig

```
TLS-Handshake abgeschlossen
≠
Zertifikat erfolgreich validiert
```

Für eine belastbare Prüfung müssen mindestens berücksichtigt werden:

- `-verify_return_error`;
- der erwartete Hostname;
- der richtige Vertrauensspeicher;
- die vollständige Zertifikatskette.

---

7. Wie wird der Zertifikatsname geprüft?

Hostname und IP-Adresse validieren

DNS-Hostname prüfen

```
[TEST][SENS] openssl s_client -connect example.com:443 -servername example.com -verify_hostname example.com -verify_return_error
```

Verbindung gegen IP, aber DNS-Namen prüfen

```
[TEST][SENS] openssl s_client -connect 192.0.2.20:443 -servername wiki.example.com -verify_hostname wiki.example.com -verify_return_error
```

IP-Adresse als Zertifikatsidentität prüfen

```
[TEST][SENS] openssl s_client -connect 192.0.2.20:443 -verify_ip 192.0.2.20 -verify_return_error
```

Ein Zertifikat muss die IP-Adresse dafür als IP-Eintrag im Subject Alternative Name enthalten. Ein DNS-Name oder ein Common Name mit einer IP-ähnlichen Zeichenfolge ist dafür nicht automatisch ausreichend.

Zu unterscheiden

Option	Aufgabe
`-connect`	Tatsächliches Netzwerkziel
`-servername`	Im TLS-Handshake gesendeter SNI-Name
`-verify_hostname`	Erwartete DNS-Identität im Zertifikat
`-verify_ip`	Erwartete IP-Identität im Zertifikat

Diese Werte können bei gezielten Tests unterschiedlich sein.

---

8. Wie wird die vom Server gesendete Zertifikatsliste angezeigt?

Zertifikate mit `-showcerts` anzeigen

```
[TEST][SENS] openssl s_client -connect example.com:443 -servername example.com -showcerts
```

`-showcerts` zeigt die Zertifikate so an, wie der Server sie übermittelt hat.

Typische Reihenfolge:

1. Server-/Leaf-Zertifikat
2. Zwischenzertifikat
3. weiteres Zwischenzertifikat

Ein Root-Zertifikat wird von einem TLS-Server normalerweise nicht mitgesendet, weil es bereits im Vertrauensspeicher des Clients vorhanden sein soll.

Wichtiger Hinweis

Die Ausgabe von `-showcerts` ist:

- die vom Server gesendete Zertifikatsliste;
- nicht automatisch eine erfolgreich validierte Kette;
- nicht zwingend vollständig;
- nicht zwingend in einer brauchbaren Vertrauenskette angeordnet;
- kein Beweis, dass alle Zertifikate vertrauenswürdig sind.

Typischer Fehler

```
Server sendet nur Leaf-Zertifikat
→ Client kann Aussteller möglicherweise nicht finden
→ unable to get local issuer certificate
```

---

9. Wie wird ein Serverzertifikat in eine Datei übernommen?

Leaf-Zertifikat extrahieren

Unter Linux und macOS kann das erste vom Server ausgegebene Zertifikat an `openssl x509` weitergegeben werden:

```
[TEST][FILE][SENS] openssl s_client -connect example.com:443 -servername example.com </dev/null 2>/dev/null | openssl x509 -outform PEM > server-cert.pem
```

Anschließend prüfen:

```
[RO][FILE][SENS] openssl x509 -in server-cert.pem -noout -subject -issuer -dates
```

Wichtig

Diese Pipeline übernimmt normalerweise das erste PEM-Zertifikat aus der Ausgabe und damit üblicherweise das Leaf-Zertifikat. Sie speichert nicht automatisch die vollständige Zertifikatskette.

Für die vollständige Kette sollte die Ausgabe von `-showcerts` kontrolliert und jedes Zertifikat eindeutig getrennt gespeichert werden.

Unter Windows

Shell-Pipelines und Umleitungen unterscheiden sich zwischen:

- PowerShell;
- Eingabeaufforderung;
- Git Bash;
- WSL.

Daher sollte unter Windows zunächst die Ausgabe kontrolliert in eine Datei geschrieben und das gewünschte PEM-Zertifikat anschließend eindeutig extrahiert werden. Binärdaten und private Schlüssel dürfen nicht unkontrolliert durch Textkonvertierungen einer Shell verändert werden.

---

10. Wie wird ein lokales Zertifikat vollständig angezeigt?

Zertifikatsinhalt mit `openssl x509` anzeigen

```
[RO][FILE][SENS] openssl x509 -in server-cert.pem -noout -text
```

Kompakte Kerninformationen

```
[RO][FILE][SENS] openssl x509 -in server-cert.pem -noout -subject -issuer -serial -dates
```

SHA-256-Fingerabdruck

```
[RO][FILE][SENS] openssl x509 -in server-cert.pem -noout -fingerprint -sha256
```

Public-Key-Information

```
[RO][FILE][SENS] openssl x509 -in server-cert.pem -noout -pubkey
```

Wichtige Zertifikatsfelder

Feld	Bedeutung

Subject	Identität des Zertifikatsinhabers
Issuer	Ausstellende Zertifizierungsstelle
Serial Number	Seriennummer des Zertifikats
Not Before	Beginn der Gültigkeit
Not After	Ende der Gültigkeit
Public Key Algorithm	Typ des öffentlichen Schlüssels
Signature Algorithm	Signaturalgorithmus der CA
Subject Alternative Name	Gültige DNS-Namen, IP-Adressen oder andere Identitäten
Key Usage	Zulässige grundlegende Schlüsselverwendungen
Extended Key Usage	Beispielsweise TLS Web Server Authentication
Basic Constraints	Kennzeichnung als Endzertifikat oder CA-Zertifikat
Authority Information Access	Hinweise auf Aussteller und OCSP
CRL Distribution Points	Verweise auf Sperrlisten

---

11. Wie werden Subject Alternative Names angezeigt?

SAN-Felder prüfen

```
[RO][FILE][SENS] openssl x509 -in server-cert.pem -noout -ext subjectAltName
```

Beispiel:

```
X509v3 Subject Alternative Name:
    DNS:example.com, DNS:www.example.com, DNS:api.example.com
```

Weitere Erweiterungen

```
[RO][FILE][SENS] openssl x509 -in server-cert.pem -noout -ext keyUsage
```

```
[RO][FILE][SENS] openssl x509 -in server-cert.pem -noout -ext extendedKeyUsage
```

```
[RO][FILE][SENS] openssl x509 -in server-cert.pem -noout -ext basicConstraints
```

```
[RO][FILE][SENS] openssl x509 -in server-cert.pem -noout -ext authorityInfoAccess
```

Wichtig

Moderne TLS-Clients prüfen die Identität hauptsächlich anhand von Subject Alternative Name. Der Common Name allein sollte nicht als verlässlicher Ersatz für fehlende SAN-Einträge betrachtet werden.

---

12. Wie wird die Gültigkeitsdauer eines Zertifikats geprüft?

Start-, Ablaufdatum und Restlaufzeit anzeigen

Gültigkeitszeitraum anzeigen

```
[RO][FILE][SENS] openssl x509 -in server-cert.pem -noout -dates
```

Ausgabe:

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

Nur Ablaufdatum

```
[RO][FILE][SENS] openssl x509 -in server-cert.pem -noout -enddate
```

Prüfen, ob das Zertifikat mindestens noch 30 Tage gültig ist

```
[RO][FILE][SENS] openssl x509 -in server-cert.pem -noout -checkend 2592000
```

Berechnung:

```
30 Tage × 24 Stunden × 60 Minuten × 60 Sekunden
= 2.592.000 Sekunden
```

Exitcode auswerten

Linux und macOS:

```
openssl x509 -in server-cert.pem -noout -checkend 2592000
echo $?
```

PowerShell:

```
openssl.exe x509 -in server-cert.pem -noout -checkend 2592000
$LASTEXITCODE
```

Exitcode	Bedeutung
0	Zertifikat ist über den angegebenen Zeitraum hinaus gültig
ungleich 0	Zertifikat läuft innerhalb des Zeitraums ab oder ist bereits abgelaufen

Zusätzlich muss die lokale Systemzeit korrekt sein. Eine falsche Uhr kann gültige Zertifikate als noch nicht gültig oder abgelaufen erscheinen lassen.

---

13. Wie wird eine Zertifikatskette lokal geprüft?

Kettenprüfung mit `openssl verify` anzeigen

Angenommene Dateien:

```
server-cert.pem
intermediate-ca.pem
root-ca.pem
```

Kette prüfen

```
[RO][FILE][SENS] openssl verify -CAfile root-ca.pem -untrusted intermediate-ca.pem server-cert.pem
```

Erwartetes Ergebnis:

```
server-cert.pem: OK
```

Zusätzlich Hostnamen prüfen

```
[RO][FILE][SENS] openssl verify -CAfile root-ca.pem -untrusted intermediate-ca.pem -verify_hostname wiki.example.com server-cert.pem
```

Für TLS-Serverzweck prüfen

```
[RO][FILE][SENS] openssl verify -CAfile root-ca.pem -untrusted intermediate-ca.pem -purpose sslserver server-cert.pem
```

Bedeutung

Option	Aufgabe
`-CAfile root-ca.pem`	Vertrauensanker
`-untrusted intermediate-ca.pem`	Nicht direkt vertrautes Zwischenzertifikat zur Kettenbildung
`-verify_hostname`	Erwarteten DNS-Namen prüfen
`-purpose sslserver`	Eignung für TLS-Server prüfen

Das Zwischenzertifikat wird mit `-untrusted` zur Kettenbildung bereitgestellt. Dadurch wird es nicht automatisch selbst zum Vertrauensanker.

---

14. Welche typischen Fehler der Zertifikatskette gibt es?

Verify-Fehler einordnen

Fehlermeldung	Mögliche Bedeutung
`unable to get local issuer certificate`	Aussteller oder Zwischenzertifikat fehlt beziehungsweise ist nicht auffindbar
`unable to verify the first certificate`	Kette kann vom Leaf-Zertifikat nicht zu einem vertrauten Root aufgebaut werden
`self-signed certificate`	Zertifikat ist selbstsigniert und nicht als vertrauenswürdig hinterlegt

Fehlermeldung	Mögliche Bedeutung
`self-signed certificate in certificate chain`	Selbstsigniertes Zertifikat in der Kette ist nicht als Vertrauensanker akzeptiert
`certificate has expired`	Zertifikat liegt nach `notAfter`
`certificate is not yet valid`	Systemzeit liegt vor `notBefore`
`hostname mismatch`	Erwarteter Name passt nicht zum Zertifikat
`invalid CA certificate`	Zertifikat ist nicht korrekt als CA verwendbar
`path length constraint exceeded`	CA-Pfadlängenbegrenzung verletzt
`unsupported certificate purpose`	Zertifikat ist nicht für den geprüften Zweck vorgesehen
`certificate signature failure`	Signatur konnte nicht korrekt bestätigt werden

Prüfreihenfolge

1. Leaf-Zertifikat untersuchen.
2. Issuer des Leaf-Zertifikats bestimmen.
3. Passendes Zwischenzertifikat prüfen.
4. Issuer und Subject der Kettenglieder vergleichen.
5. CA-Eigenschaften kontrollieren.
6. Vertrauensanker eindeutig festlegen.
7. Gültigkeitszeiten aller Zertifikate prüfen.
8. Hostnamen und Verwendungszweck prüfen.

---

15. Wie wird eine interne Zertifizierungsstelle verwendet?

Eigene CA für einen einzelnen Test angeben

TLS-Verbindung mit eigener CA-Datei prüfen

```
[TEST][FILE][SENS] openssl s_client -connect wiki.example.com:443 -servername wiki.example.com -verify_hostname wiki.example.com -verify_return_error -CAfile company-root-ca.pem
```

Lokales Zertifikat prüfen

```
[RO][FILE][SENS] openssl verify -CAfile company-root-ca.pem -untrusted company-intermediate-ca.pem -verify_hostname wiki.example.com server-cert.pem
```

Wichtig

Die Angabe von `-CAfile` gilt für diesen OpenSSL-Aufruf. Sie installiert die CA nicht automatisch in:

- Windows-Zertifikatspeicher;
- macOS-Schlüsselbund;
- Linux-Systemvertrauensspeicher;
- Browser-Vertrauensspeicher;
- Java-KeyStore;
- Container-Images.

Ein erfolgreicher OpenSSL-Test mit eigener CA-Datei beweist deshalb nicht, dass alle Anwendungen dieser CA vertrauen.

---

16. Wie werden TLS 1.2 und TLS 1.3 getrennt getestet?

Protokollversionen anzeigen

Ausschließlich TLS 1.2

```
[TEST][SENS] openssl s_client -connect example.com:443 -servername example.com -tls1_2
```

Ausschließlich TLS 1.3

```
[TEST][SENS] openssl s_client -connect example.com:443 -servername example.com -tls1_3
```

Minimalversion festlegen, sofern unterstützt

```
[TEST][SENS] openssl s_client -connect example.com:443 -servername example.com -min_protocol TLSv1.2
```

Maximalversion festlegen

```
[TEST][SENS] openssl s_client -connect example.com:443 -servername example.com -max_protocol TLSv1.2
```

Interpretation

TLS 1.2	TLS 1.3	Mögliche Aussage
Erfolgreich	Erfolgreich	Beide Versionen werden unterstützt

TLS 1.2	TLS 1.3	Mögliche Aussage
Erfolgreich	Fehler	Server, Proxy oder Client unterstützt TLS 1.3 nicht oder blockiert es
Fehler	Erfolgreich	Server erlaubt möglicherweise nur TLS 1.3
Beide Fehler	TCP, SNI, Zertifikat, Cipher oder Dienst prüfen	

Ein Versionsfehler kann auch durch einen zwischengeschalteten Proxy, Load Balancer oder TLS-Inspection-Dienst verursacht werden.

---

17. Wie werden Cipher Suites geprüft?

Cipher-Informationen anzeigen

Verfügbare Cipher Suites des lokalen OpenSSL-Builds

```
[RO] openssl ciphers -v
```

Nur TLS-1.2-kompatible Auswahl untersuchen

```
[RO] openssl ciphers -v -tls1_2
```

Nur TLS-1.3-Cipher-Suites anzeigen, sofern unterstützt

```
[RO] openssl ciphers -v -tls1_3
```

Bestimmte TLS-1.2-Cipher-Auswahl testen

```
[TEST][SENS] openssl s_client -connect example.com:443 -servername example.com -tls1_2 -cipher 'ECDHE-RSA-AES128-GCM-SHA256'
```

Bestimmte TLS-1.3-Cipher-Suite testen

```
[TEST][SENS] openssl s_client -connect example.com:443 -servername example.com -tls1_3 -ciphersuites 'TLS_AES_128_GCM_SHA256'
```

Wichtiger Unterschied

Option	Gilt hauptsächlich für
`-cipher`	TLS 1.2 und älter
`-ciphersuites`	TLS 1.3

Die tatsächlich verfügbaren Cipher Suites hängen von:

- OpenSSL-Version;
- aktivierten Providern;
- Sicherheitsstufe;
- Betriebssystempaket;
- Build-Konfiguration;
- kryptografischer Richtlinie

ab.

> Eine einzelne funktionierende Cipher Suite beweist nicht, dass die gesamte TLS-Konfiguration sicher oder vollständig kompatibel ist.

---

18. Wie wird ALPN für HTTP/2 und HTTP/1.1 geprüft?

ALPN-Aushandlung anzeigen

HTTP/2 und HTTP/1.1 anbieten

```
[TEST][SENS] openssl s_client -connect example.com:443 -servername example.com -alpn 'h2,http/1.1'
```

Mögliche Ausgabe:

```
ALPN protocol: h2
```

oder:

```
ALPN protocol: http/1.1
```

Nur HTTP/2 anbieten

```
[TEST][SENS] openssl s_client -connect example.com:443 -servername example.com -alpn h2
```

Interpretation

Ergebnis	Bedeutung
`h2`	Server und Client einigten sich auf HTTP/2
`http/1.1`	HTTP/1.1 wurde gewählt
Keine ALPN-Aushandlung	Server, Proxy oder OpenSSL-Build unterstützt beziehungsweise verwendet ALPN nicht
TLS-Abbruch	Protokoll-, Cipher-, SNI- oder Richtlinienproblem möglich

ALPN bestimmt das Anwendungsprotokoll innerhalb der TLS-Verbindung. OpenSSL `s_client` führt danach nicht automatisch eine vollständige HTTP/2-Anfrage aus.

Für den Anwendungstest:

```
[TEST][SENS] curl --http2 -v https://example.com/
```

---

19. Wie wird OCSP Stapling geprüft?

OCSP-Statusantwort anzeigen

```
[TEST][SENS] openssl s_client -connect example.com:443 -servername example.com -status
```

Die Option fordert während des TLS-Handshakes eine gestapelte OCSP-Antwort an.

Mögliche Ausgabe:

```
OCSP response:
...
Cert Status: good
```

Mögliche Ergebnisse

Beobachtung	Einordnung
Gültige Antwort mit `good`	Server lieferte eine auswertbare OCSP-Statusantwort
`revoked`	Zertifikat wird als gesperrt gemeldet
`unknown`	OCSP-Responder kennt den Status nicht eindeutig
`OCSP response: no response sent`	Server lieferte kein OCSP Stapling
Fehlerhafte Antwort	OCSP-Antwort, Signatur oder Aktualität prüfen

Kein OCSP Stapling bedeutet nicht automatisch, dass das Zertifikat gesperrt oder die TLS-Verbindung ungültig ist. Ob Stapling erforderlich ist, hängt von Zertifikat, Anwendung und Richtlinie ab.

---

20. Wie wird STARTTLS bei Maildiensten geprüft?

SMTP, IMAP und POP3 anzeigen

Bei STARTTLS beginnt die Verbindung unverschlüsselt und wird anschließend innerhalb des Anwendungsprotokolls auf TLS umgestellt.

SMTP auf Port 25

```
[TEST][SENS] openssl s_client -starttls smtp -connect mail.example.com:25 -servername mail.example.com
```

SMTP Submission auf Port 587

```
[TEST][SENS] openssl s_client -starttls smtp -connect mail.example.com:587 -servername
mail.example.com
```

IMAP mit STARTTLS

```
[TEST][SENS] openssl s_client -starttls imap -connect mail.example.com:143 -servername
mail.example.com
```

POP3 mit STARTTLS

```
[TEST][SENS] openssl s_client -starttls pop3 -connect mail.example.com:110 -servername
mail.example.com
```

LMTP mit STARTTLS, sofern unterstützt

```
[TEST][SENS] openssl s_client -starttls lmtp -connect mail.example.com:24 -servername mail.example.com
```

Direktes TLS ohne STARTTLS

Dienst	Typischer Port	Aufruf
SMTPS	465	`openssl s_client -connect mail.example.com:465 -servername mail.example.com`
IMAPS	993	`openssl s_client -connect mail.example.com:993 -servername mail.example.com`
POP3S	995	`openssl s_client -connect mail.example.com:995 -servername mail.example.com`

STARTTLS und direktes TLS dürfen nicht verwechselt werden.

---

21. Wie wird STARTTLS bei LDAP und Datenbanken geprüft?

LDAP-, PostgreSQL- und MySQL-Beispiele anzeigen

LDAP mit STARTTLS

```
[TEST][SENS] openssl s_client -starttls ldap -connect ldap.example.com:389 -servername ldap.example.com
```

LDAPS mit direktem TLS

```
[TEST][SENS] openssl s_client -connect ldap.example.com:636 -servername ldap.example.com
```

PostgreSQL mit TLS-Aushandlung

```
[TEST][SENS] openssl s_client -starttls postgres -connect db.example.com:5432 -servername db.example.com
```

MySQL mit TLS-Aushandlung, sofern von der installierten OpenSSL-Version unterstützt

```
[TEST][SENS] openssl s_client -starttls mysql -connect db.example.com:3306 -servername db.example.com
```

Nicht jede OpenSSL- oder LibreSSL-Version unterstützt alle `-starttls`-Protokolle. Verfügbarkeit prüfen:

```
[RO] openssl s_client -help
```

Ein erfolgreicher TLS-Handshake beweist nicht, dass anschließend eine Datenbank- oder LDAP-Anmeldung erfolgreich ist.

---

22. Wie wird eine Verbindung über einen HTTP-Proxy getestet?

HTTP-CONNECT-Proxy anzeigen

```
[TEST][SENS] openssl s_client -proxy proxy.example.com:8080 -connect example.com:443 -servername example.com
```

OpenSSL sendet dabei eine HTTP-CONNECT-Anfrage an den Proxy und baut durch den Tunnel die TLS-Verbindung zum Zielserver auf.

Proxybenutzer angeben

```
[TEST][SENS] openssl s_client -proxy proxy.example.com:8080 -proxy_user max.mustermann -proxy_pass stdin -connect example.com:443 -servername example.com
```

Das Passwort wird über die angegebene Passwortquelle eingelesen.

Wichtiger Sicherheitshinweis

Die von `s_client` unterstützte Proxy-Basisauthentifizierung überträgt die Zugangsdaten vor dem TLS-Tunnel in leicht rückwandelbarer Base64-Form an den Proxy. Sie ist nur in entsprechend geschützten und autorisierten Umgebungen zu verwenden.

Direkt- und Proxytest vergleichen

Direkt:

```
[TEST][SENS] openssl s_client -connect example.com:443 -servername example.com
```

Über Proxy:

```
[TEST][SENS] openssl s_client -proxy proxy.example.com:8080 -connect example.com:443 -servername example.com
```

Unterschiedliche Zertifikate können auf TLS-Inspection oder unterschiedliche Netzwerkpfade hinweisen.

---

23. Wie wird eine mTLS-Verbindung mit Clientzertifikat geprüft?

Clientzertifikat und privaten Schlüssel verwenden

Clientzertifikat und Schlüssel getrennt

```
[TEST][FILE][SENS] openssl s_client -connect api.example.com:443 -servername api.example.com -cert client-cert.pem -key client-key.pem
```

Zusätzlich Serverzertifikat und Hostname verifizieren

```
[TEST][FILE][SENS] openssl s_client -connect api.example.com:443 -servername api.example.com -cert client-cert.pem -key client-key.pem -CAfile company-root-ca.pem -verify_hostname api.example.com -verify_return_error
```

Clientzertifikatskette angeben

```
[TEST][FILE][SENS] openssl s_client -connect api.example.com:443 -servername api.example.com -cert client-cert.pem -key client-key.pem -cert_chain client-chain.pem -build_chain
```

Die Optionen `-cert_chain` und `-build_chain` sind versionsabhängig.

Mögliche Fehler

- Clientzertifikat fehlt;
- privater Schlüssel passt nicht zum Zertifikat;
- Clientzertifikat abgelaufen;

- Extended Key Usage erlaubt keine Clientauthentifizierung;
- ausstellende Client-CA wird vom Server nicht vertraut;
- Zwischenzertifikat fehlt;
- falsches Dateiformat;
- verschlüsselter Schlüssel kann nicht geöffnet werden.

> Private Schlüssel dürfen niemals in Tickets, BookStack-Seiten, Chatnachrichten oder Diagnoseausgaben eingefügt werden.

---

24. Wie wird ein privater Schlüssel geprüft, ohne ihn auszugeben?

Schlüsselstruktur sicher prüfen

Allgemeinen privaten Schlüssel prüfen

```
[RO][FILE][SENS] openssl pkey -in server-key.pem -check -noout
```

Bei einem verschlüsselten Schlüssel wird normalerweise eine Passphrase abgefragt.

Nur öffentliche Schlüsselkomponente ausgeben

```
[RO][FILE][SENS] openssl pkey -in server-key.pem -pubout
```

Schlüsselinformation ohne private Bestandteile

```
[RO][FILE][SENS] openssl pkey -in server-key.pem -pubout -text_pub -noout
```

Nicht unkontrolliert verwenden

```
openssl pkey -in server-key.pem -text
```

Dieser Befehl kann private Schlüsselbestandteile ausgeben. Solche Ausgaben dürfen nicht in Logs, Terminalsitzungsaufzeichnungen oder Tickets gelangen.

Dateiberechtigungen prüfen

Linux:

```
[RO][FILE][SENS] stat server-key.pem
```

macOS:

```
[RO][FILE][SENS] stat -x server-key.pem
```

Windows:

```
[RO][FILE][SENS] Get-Acl .\server-key.pem
```

---

25. Wie wird geprüft, ob Zertifikat und privater Schlüssel zusammengehören?

Öffentliche Schlüssel vergleichen

Die zuverlässige allgemeine Methode besteht darin, aus beiden Dateien den öffentlichen Schlüssel zu gewinnen und dessen DER-Darstellung zu hashen.

Hash des öffentlichen Schlüssels aus dem Zertifikat

```
[RO][FILE][SENS] openssl x509 -in server-cert.pem -pubkey -noout | openssl pkey -pubin -outform DER | openssl dgst -sha256
```

Hash des öffentlichen Schlüssels aus dem privaten Schlüssel

```
[RO][FILE][SENS] openssl pkey -in server-key.pem -pubout -outform DER | openssl dgst -sha256
```

Wenn beide SHA-256-Werte identisch sind, besitzen Zertifikat und privater Schlüssel denselben öffentlichen Schlüssel.

Beispiel

```
SHA2-256(stdin)= abcdef...
SHA2-256(stdin)= abcdef...
```

Warum nicht nur den RSA-Modulus vergleichen?

Der klassische Modulusvergleich funktioniert nur für bestimmte Schlüsselarten wie RSA. Der Vergleich des normalisierten öffentlichen Schlüssels funktioniert auch für andere von OpenSSL unterstützte Schlüsseltypen, beispielsweise EC-Schlüssel.

---

26. Wie wird eine Certificate Signing Request geprüft?

CSR-Inhalt und Signatur anzeigen

Gesamten CSR-Inhalt anzeigen

```
[RO][FILE][SENS] openssl req -in request.csr -noout -text
```

Subject anzeigen

```
[RO][FILE][SENS] openssl req -in request.csr -noout -subject
```

CSR-Signatur prüfen

```
[RO][FILE][SENS] openssl req -in request.csr -noout -verify
```

Öffentlichen Schlüssel des CSR ausgeben

```
[RO][FILE][SENS] openssl req -in request.csr -noout -pubkey
```

Zu prüfen

- Subject;
- gewünschte DNS-Namen;
- Subject Alternative Names;
- Schlüsseltyp;
- Schlüssellänge beziehungsweise Kurve;
- Signaturalgorithmus;
- erfolgreiche CSR-Signaturprüfung.

Ein CSR ist noch kein Zertifikat. Er enthält:

- Identitätsantrag;

- öffentlichen Schlüssel;
- angeforderte Erweiterungen;
- Signatur des zugehörigen privaten Schlüssels.

Er enthält nicht den privaten Schlüssel.

---

27. Wie wird geprüft, ob CSR und privater Schlüssel zusammengehören?

CSR-Schlüsselvergleich anzeigen

Öffentlichen Schlüssel aus dem CSR hashen

```
[RO][FILE][SENS] openssl req -in request.csr -pubkey -noout | openssl pkey -pubin -outform DER | openssl dgst -sha256
```

Öffentlichen Schlüssel aus dem privaten Schlüssel hashen

```
[RO][FILE][SENS] openssl pkey -in server-key.pem -pubout -outform DER | openssl dgst -sha256
```

Sind beide Werte identisch, gehört der private Schlüssel zum CSR.

Nach Ausstellung zusätzlich vergleichen

```
CSR ↔ privater Schlüssel
Zertifikat ↔ privater Schlüssel
```

Dadurch wird geprüft, ob die Zertifizierungsstelle tatsächlich das Zertifikat für den vorgesehenen Schlüssel ausgestellt hat.

---

28. Wie werden PEM- und DER-Zertifikate unterschieden und konvertiert?

Zertifikatsformate anzeigen

PEM

PEM ist textbasiert und enthält Markierungen wie:

```
-----BEGIN CERTIFICATE-----
...
-----END CERTIFICATE-----
```

DER

DER ist ein binäres ASN.1-Format und besitzt keine lesbaren BEGIN-/END-Markierungen.

PEM-Zertifikat anzeigen

```
[RO][FILE][SENS] openssl x509 -in certificate.pem -inform PEM -noout -text
```

DER-Zertifikat anzeigen

```
[RO][FILE][SENS] openssl x509 -in certificate.der -inform DER -noout -text
```

DER nach PEM konvertieren

```
[CHANGE][FILE][SENS] openssl x509 -in certificate.der -inform DER -out certificate.pem -outform PEM
```

PEM nach DER konvertieren

```
[CHANGE][FILE][SENS] openssl x509 -in certificate.pem -inform PEM -out certificate.der -outform DER
```

Die Dateiendung allein beweist das Format nicht. Dateien mit `.cer` oder `.crt` können PEM oder DER enthalten.

---

29. Wie wird eine PKCS#12-Datei untersucht?

PFX/P12-Inhalt sicher prüfen

PKCS#12-Dateien verwenden häufig die Endungen:

```
.p12
.pfx
```

Sie können enthalten:

- Zertifikat;
- Zwischenzertifikate;
- Root-Zertifikate;
- privaten Schlüssel;
- Metadaten.

Inhalt anzeigen, aber keine privaten Schlüssel ausgeben

```
[RO][FILE][SENS] openssl pkcs12 -in bundle.p12 -info -noout
```

Nur Zertifikate anzeigen, keine privaten Schlüssel

```
[RO][FILE][SENS] openssl pkcs12 -in bundle.p12 -nokeys
```

Nur das Client- beziehungsweise Leaf-Zertifikat extrahieren

```
[CHANGE][FILE][SENS] openssl pkcs12 -in bundle.p12 -clcerts -nokeys -out client-cert.pem
```

CA-Zertifikate extrahieren

```
[CHANGE][FILE][SENS] openssl pkcs12 -in bundle.p12 -cacerts -nokeys -out ca-chain.pem
```

Privaten Schlüssel extrahieren

```
[CHANGE][FILE][SENS] openssl pkcs12 -in bundle.p12 -nocerts -out client-key-encrypted.pem
```

Dieser Vorgang erzeugt eine Datei mit privatem Schlüssel und ist besonders sensibel. Die Ausgabedatei sollte verschlüsselt und mit streng beschränkten Dateirechten gespeichert werden.

Unverschlüsselten privaten Schlüssel exportieren

Optionen wie `-nodes` beziehungsweise das neuere `-noenc` können unverschlüsselte private Schlüssel erzeugen. Das sollte nur in begründeten Ausnahmefällen, mit dokumentierter Freigabe und geeigneten Dateirechten erfolgen.

---

30. Wie werden Zertifikatsfingerabdrücke verglichen?

SHA-256-Fingerprints anzeigen

Zertifikatsfingerabdruck

```
[RO][FILE][SENS] openssl x509 -in server-cert.pem -noout -fingerprint -sha256
```

Hash der gesamten Datei

```
[RO][FILE][SENS] openssl dgst -sha256 server-cert.pem
```

Diese beiden Werte sind nicht dasselbe:

Prüfung	Gehashte Daten
`openssl x509 -fingerprint -sha256`	DER-codiertes Zertifikat
`openssl dgst -sha256 server-cert.pem`	Tatsächliche Datei einschließlich PEM-Darstellung

Zwei PEM-Dateien können dasselbe Zertifikat enthalten, aber wegen unterschiedlicher Zeilenenden oder Formatierung unterschiedliche Dateihashes besitzen. Der Zertifikatsfingerabdruck bleibt dagegen identisch.

Betriebssystemspezifischer Dateihash

Windows:

```
[RO][FILE][SENS] Get-FileHash .\server-cert.pem -Algorithm SHA256
```

Linux:

```
[RO][FILE][SENS] sha256sum server-cert.pem
```

macOS:

```
[RO][FILE][SENS] shasum -a 256 server-cert.pem
```

---

31. Wie werden detaillierte TLS-Zustände und Nachrichten untersucht?

Erweiterte Diagnoseoptionen anzeigen

TLS-Zustandswechsel anzeigen

```
[TEST][SENS] openssl s_client -connect example.com:443 -servername example.com -state
```

TLS-Protokollnachrichten anzeigen

```
[TEST][SENS] openssl s_client -connect example.com:443 -servername example.com -msg
```

Umfangreicher Trace, sofern unterstützt

```
[TEST][SENS] openssl s_client -connect example.com:443 -servername example.com -trace
```

TLS-Erweiterungen anzeigen

```
[TEST][SENS] openssl s_client -connect example.com:443 -servername example.com -tlsextdebug
```

Wichtiger Hinweis

`-debug`, `-msg` und `-trace` können sehr umfangreiche Hexadezimal- und Protokollausgaben erzeugen. Sie sollten erst verwendet werden, wenn:

- der normale `s_client`-Test nicht ausreicht;
- der Fehler reproduzierbar ist;
- die Ausgabe geschützt gespeichert wird;
- keine geheimen Anwendungsdaten eingegeben werden.

---

32. Wie werden IPv4 und IPv6 getrennt getestet?

IPv4-/IPv6-TLS-Vergleich anzeigen

IPv4 erzwingen

```
[TEST][SENS] openssl s_client -4 -connect example.com:443 -servername example.com
```

IPv6 erzwingen

```
[TEST][SENS] openssl s_client -6 -connect example.com:443 -servername example.com
```

Direkte IPv6-Adresse

```
[TEST][SENS] openssl s_client -connect '[2001:db8::20]:443' -servername example.com
```

Bei einer IPv6-Adresse in `host:port`-Schreibweise muss die Adresse in eckige Klammern gesetzt werden.

Interpretation

IPv4	IPv6	Mögliche Ursache
Erfolgreich	Fehler	IPv6-DNS, Routing, Firewall, MTU oder Zielkonfiguration
Fehler	Erfolgreich	IPv4-Routing, NAT, Firewall oder Zielkonfiguration
Unterschiedliches Zertifikat	Unterschiedlicher Load Balancer oder Server	
Unterschiedliche TLS-Version	Unterschiedliche Proxy- oder Serverkonfiguration	

---

33. Wie wird ein TLS-Fehler systematisch eingegrenzt?

Diagnosekette anzeigen

1. DNS prüfen

Windows:

```
[RO] Resolve-DnsName example.com
```

Linux:

```
[RO] dig example.com
```

macOS:

```
[RO] dig example.com
```

2. TCP-Port prüfen

Windows:

```
[TEST] Test-NetConnection example.com -Port 443
```

Linux und macOS:

```
[TEST] nc -vz example.com 443
```

3. TLS mit SNI prüfen

```
[TEST][SENS] openssl s_client -connect example.com:443 -servername example.com
```

4. Zertifikatsprüfung erzwingen

```
[TEST][SENS] openssl s_client -connect example.com:443 -servername example.com -verify_hostname example.com -verify_return_error
```

5. Zertifikatsliste anzeigen

```
[TEST][SENS] openssl s_client -connect example.com:443 -servername example.com -showcerts
```

6. TLS-Versionen getrennt prüfen

```
[TEST][SENS] openssl s_client -connect example.com:443 -servername example.com -tls1_2
```

```
[TEST][SENS] openssl s_client -connect example.com:443 -servername example.com -tls1_3
```

7. Anwendung prüfen

```
[TEST][SENS] curl -v https://example.com/
```

8. Logs vergleichen

- Reverse Proxy;
- Webserver;
- Load Balancer;
- Firewall;
- Anwendung;
- Zertifikatsverwaltung;
- Systemzeit.

---

34. Wie werden typische OpenSSL-Fehler eingeordnet?

Fehlertabelle anzeigen

Fehler oder Beobachtung	Mögliche Ursache
`Connection refused`	Ziel erreichbar, aber kein Listener am Port
`Connection timed out`	Firewall, Routing, Rückweg oder Ziel nicht erreichbar
`no peer certificate available`	TLS-Handshake brach vor Zertifikatsübertragung ab
`wrong version number`	Falsches Protokoll am Port, beispielsweise direktes TLS gegen einen Klartextport
`unknown protocol`	Ziel spricht kein erwartetes TLS-Protokoll
`handshake failure`	Keine gemeinsamen Parameter, Richtlinie, Clientzertifikat oder Serverkonfiguration
`no shared cipher`	Keine gemeinsame Cipher Suite
`unsupported protocol`	TLS-Version nicht unterstützt oder deaktiviert
`certificate verify failed`	Vertrauenskette, Zeit, Zweck oder Name fehlerhaft
`unable to get local issuer certificate`	Zwischenzertifikat oder Aussteller fehlt
`self-signed certificate`	Nicht vertrautes selbstsigniertes Zertifikat
`certificate has expired`	Zertifikat abgelaufen
`certificate is not yet valid`	Systemzeit falsch oder Gültigkeit noch nicht begonnen
`hostname mismatch`	Zertifikat gilt nicht für erwarteten Namen
`bad certificate`	Gegenstelle lehnt Zertifikat ab
`certificate required`	Server verlangt ein Clientzertifikat
`tlsv1 alert protocol version`	Gegenstelle akzeptiert angebotene TLS-Version nicht

Fehler oder Beobachtung	Mögliche Ursache
`unexpected eof while reading`	Gegenstelle oder Zwischenkomponente beendet Verbindung ohne regulären TLS-Abschluss
Falsches Zertifikat	SNI fehlt, falscher Load Balancer oder falsches Ziel
`Verify return code: 0 (ok)` trotz falscher Anwendung	TLS-Zertifikat okay, Anwendungsproblem bleibt möglich

---

35. Welche Fehlinterpretationen müssen vermieden werden?

Praxisfallen anzeigen

Fehlinterpretation	Richtige Einordnung
TLS-Handshake erfolgreich bedeutet Zertifikat gültig	`s_client` kann nach Verify-Fehlern fortfahren
`CONNECTED` bedeutet TLS vollständig erfolgreich	Es bestätigt zunächst die hergestellte Transportverbindung
`-showcerts` zeigt eine validierte Kette	Es zeigt nur die vom Server gesendeten Zertifikate
Root-Zertifikat fehlt in Serverausgabe	Root-CA wird normalerweise nicht mitgesendet
Common Name reicht immer aus	Moderne Clients prüfen hauptsächlich SAN
Test über IP entspricht Test über DNS-Namen	SNI und Hostnamenprüfung können abweichen
`-servername` prüft automatisch den Zertifikatsnamen	SNI und Hostnamenvalidierung sind getrennte Funktionen
`-verify_hostname` setzt automatisch das Netzwerkziel	Das Netzwerkziel wird durch `-connect` bestimmt
Selbstsigniert bedeutet automatisch unsicher	Es bedeutet zunächst, dass kein externer Vertrauenspfad vorliegt
`Verify return code: 0` beweist funktionierende Website	Nur die geprüfte TLS-/Zertifikatsebene ist erfolgreich
Gleiches Subject bedeutet gleiches Zertifikat	Seriennummer, Public Key und Fingerprint können verschieden sein
Gleicher Dateihash ist der einzige Zertifikatsvergleich	PEM-Formatierung kann Dateihash verändern
PKCS#12 ist nur ein Zertifikat	Datei kann zusätzlich private Schlüssel und Ketten enthalten
Private Schlüssel können zur Diagnose angezeigt werden	Private Schlüssel dürfen nicht offengelegt werden
`-tls1_2` bedeutet mindestens TLS 1.2	Bei `s_client` erzwingt die Option den Test mit TLS 1.2
Fehlendes OCSP Stapling bedeutet gesperrt	Server hat lediglich keine Stapling-Antwort geliefert

---

36. Wie sieht ein sicherer OpenSSL-Diagnoseablauf aus?

Empfohlene Schrittfolge anzeigen

Vorbereitung

1. Erwarteten Hostnamen, Ziel-IP und Port dokumentieren.
2. Klären, ob direktes TLS oder STARTTLS verwendet wird.
3. OpenSSL- beziehungsweise LibreSSL-Version prüfen.
4. Erwartete Zertifizierungsstelle und Zertifikatsnamen ermitteln.
5. Systemzeit und Zeitzone kontrollieren.
6. Private Schlüssel und Passwortquellen schützen.

Verbindung

7. DNS-Auflösung prüfen.
8. TCP-Port prüfen.
9. TLS-Verbindung mit richtigem SNI aufbauen.
10. Tatsächlich ausgehandelte TLS-Version und Cipher Suite dokumentieren.
11. ALPN bei HTTP-Diensten prüfen.

Zertifikat

12. Zertifikatsprüffehlermeldungen beachten.
13. Prüfung mit `-verify_return_error` erzwingen.
14. Hostnamen mit `-verify_hostname` prüfen.
15. Serverzertifikate mit `-showcerts` anzeigen.
16. Leaf- und Zwischenzertifikate getrennt untersuchen.
17. SAN, Issuer, Subject und Gültigkeit prüfen.
18. Kette mit `openssl verify` lokal nachvollziehen.

Vertiefung

19. TLS 1.2 und TLS 1.3 getrennt testen.
20. IPv4 und IPv6 vergleichen.
21. Direkt- und Proxyverbindung vergleichen.
22. Bei mTLS Clientzertifikat und Schlüsselzuordnung prüfen.
23. Bei STARTTLS das richtige Anwendungsprotokoll angeben.
24. Nur bei Bedarf `-state`, `-msg` oder `-trace` verwenden.

Validierung

25. HTTPS zusätzlich mit curl prüfen.
26. Server-, Proxy- und Firewall-Logs vergleichen.
27. Bei Netzwerkverdacht Paketmitschnitt erstellen.
28. Nach einer Änderung exakt denselben Test wiederholen.
29. Diagnoseausgaben vor Weitergabe auf sensible Daten prüfen.

30. Temporär extrahierte Schlüssel- oder Zertifikatsdateien sicher behandeln.

---

37. Kurzreferenz – häufige OpenSSL-Befehle

Befehlstabelle anzeigen

Aufgabe	Befehl
Version	[RO] openssl version -a
TLS-Verbindung	[TEST][SENS] openssl s_client -connect example.com:443 -servername example.com
Kompakte Ausgabe	[TEST][SENS] openssl s_client -connect example.com:443 -servername example.com -brief
Hostname prüfen	[TEST][SENS] openssl s_client -connect example.com:443 -servername example.com -verify_hostname example.com -verify_return_error
Zertifikatsliste	[TEST][SENS] openssl s_client -connect example.com:443 -servername example.com -showcerts
Eigene CA	[TEST][FILE][SENS] openssl s_client -connect example.com:443 -servername example.com -CAfile root-ca.pem -verify_return_error
TLS 1.2	[TEST][SENS] openssl s_client -connect example.com:443 -servername example.com -tls1_2
TLS 1.3	[TEST][SENS] openssl s_client -connect example.com:443 -servername example.com -tls1_3
ALPN	[TEST][SENS] openssl s_client -connect example.com:443 -servername example.com -alpn 'h2,http/1.1'
OCSP Stapling	[TEST][SENS] openssl s_client -connect example.com:443 -servername example.com -status
SMTP STARTTLS	[TEST][SENS] openssl s_client -starttls smtp -connect mail.example.com:587 -servername mail.example.com
IMAP STARTTLS	[TEST][SENS] openssl s_client -starttls imap -connect mail.example.com:143 -servername mail.example.com
LDAP STARTTLS	[TEST][SENS] openssl s_client -starttls ldap -connect ldap.example.com:389 -servername ldap.example.com
Zertifikat anzeigen	[RO][FILE][SENS] openssl x509 -in server-cert.pem -noout -text
Kerndaten	[RO][FILE][SENS] openssl x509 -in server-cert.pem -noout -subject -issuer -serial -dates
SAN anzeigen	[RO][FILE][SENS] openssl x509 -in server-cert.pem -noout -ext subjectAltName
SHA-256-Fingerprint	[RO][FILE][SENS] openssl x509 -in server-cert.pem -noout -fingerprint -sha256
30 Tage Restlaufzeit	[RO][FILE][SENS] openssl x509 -in server-cert.pem -noout -checkend 2592000

Aufgabe	Befehl
Kette prüfen	`[RO][FILE][SENS] openssl verify -CAfile root-ca.pem -untrusted intermediate-ca.pem server-cert.pem`
Privaten Schlüssel prüfen	`[RO][FILE][SENS] openssl pkey -in server-key.pem -check -noout`
CSR prüfen	`[RO][FILE][SENS] openssl req -in request.csr -noout -text -verify`
PKCS#12 untersuchen	`[RO][FILE][SENS] openssl pkcs12 -in bundle.p12 -info -noout`
DER nach PEM	`[CHANGE][FILE][SENS] openssl x509 -in certificate.der -inform DER -out certificate.pem -outform PEM`

---

38. Kurzreferenz – wichtige `s_client`-Optionen

Optionstabelle anzeigen

Option	Bedeutung
`-connect host:port`	Netzwerkziel festlegen
`-servername name`	TLS-SNI setzen
`-noservername`	Kein SNI senden
`-verify_hostname name`	DNS-Namen des Zertifikats prüfen
`-verify_ip adresse`	IP-Identität des Zertifikats prüfen
`-verify_return_error`	Bei Zertifikatsprüffehlern abbrechen
`-CAfile datei`	Vertrauenswürdige CA-Datei verwenden
`-showcerts`	Vom Server gesendete Zertifikate anzeigen
`-brief`	Kompakte Verbindungsübersicht
`-no-interactive`	Nicht interaktiver Modus, sofern unterstützt
`-4`	IPv4 erzwingen
`-6`	IPv6 erzwingen
`-tls1_2`	TLS 1.2 erzwingen
`-tls1_3`	TLS 1.3 erzwingen
`-min_protocol`	Minimale Protokollversion
`-max_protocol`	Maximale Protokollversion
`-cipher`	Cipher-Auswahl für TLS 1.2 und älter
`-ciphersuites`	Cipher-Suites für TLS 1.3
`-alpn`	ALPN-Protokolle anbieten

Option	Bedeutung
`-status`	OCSP-Stapling-Antwort anfordern
`-starttls`	STARTTLS für ein Anwendungsprotokoll
`-proxy`	HTTP-CONNECT-Proxy verwenden
`-cert`	Clientzertifikat angeben
`-key`	Privaten Clientschlüssel angeben
`-cert_chain`	Clientzertifikatskette angeben
`-state`	TLS-Zustände anzeigen
`-msg`	TLS-Protokollnachrichten anzeigen
`-trace`	Ausführlichen TLS-Trace anzeigen
`-tlsextdebug`	TLS-Erweiterungen anzeigen

Merksätze

- OpenSSL- und LibreSSL-Version sowie lokale Optionen müssen zuerst geprüft werden.
- `-connect` bestimmt das Netzwerkziel, `-servername` das SNI und `-verify_hostname` den zu prüfenden Zertifikatsnamen.
- Ein abgeschlossener TLS-Handshake beweist nicht automatisch eine erfolgreiche Zertifikatsprüfung.
- `s_client` kann ohne `-verify_return_error` nach Zertifikatsfehlern fortfahren.
- `Verify return code: 0 (ok)` ist wesentlich aussagekräftiger als nur `CONNECTED`.
- `-showcerts` zeigt die vom Server gesendeten Zertifikate, aber keine automatisch validierte Kette.
- Das Root-Zertifikat wird normalerweise nicht vom TLS-Server mitgesendet.
- Moderne Clients prüfen DNS-Namen hauptsächlich anhand des Subject Alternative Name.
- STARTTLS und direktes TLS verwenden unterschiedliche Verbindungsabläufe.
- `-tls1_2` und `-tls1_3` erzwingen jeweils eine bestimmte TLS-Version.
- TLS-1.2-Cipher werden mit `-cipher`, TLS-1.3-Cipher-Suites mit `-ciphersuites` gewählt.
- Eine interne CA mit `-CAfile` gilt nur für den jeweiligen OpenSSL-Aufruf.
- Zertifikat, CSR und privater Schlüssel können über ihre öffentlichen Schlüssel sicher miteinander verglichen werden.
- Private Schlüssel dürfen niemals in BookStack, Tickets, Chats oder Diagnoseprotokollen veröffentlicht werden.
- Eine erfolgreiche TLS-Prüfung ersetzt nicht den anschließenden Anwendungstest mit curl oder einem protokollspezifischen Client.

Quellen

- Offizielle OpenSSL-Dokumentation

- OpenSSL `s_client`
- OpenSSL `x509`
- OpenSSL `verify`
- OpenSSL `pkey`
- OpenSSL `req`
- OpenSSL `pkcs12`
- OpenSSL `ciphers`
- OpenSSL `version`
- OpenSSL Verification Options
- OpenSSL Passphrase Options