1 Sichere Verbindungen

Sichere Verbindungen dienen dazu, die Schutzziele der darüberliegenden Protokollschichten zwischen Endpunkten zu gewährleisten. Je nach Schicht (Layer) im OSI-Modell kommen unterschiedliche Protokolle zum Einsatz.

Schutzziele sicherer Verbindungen

- **Vertraulichkeit**: Daten können von Dritten nicht mitgelesen werden.
- **Integrität**: Daten können nicht unbemerkt manipuliert werden.
- **Authentizität**: Die Identität der Kommunikationspartner ist gesichert.

1.0.1 Protokolle und ihre Schichten

- **Layer 7 (Application):** SSH, Signal, PGP
- **Layer 4 (Transport):** TLS (Transport Layer Security), QUIC
- **Layer 3 (Network):** IPsec
- **Layer 2 (Link):** WPA2, WPA3

Hinweis: Ungeschützte Protokolle (z.B. HTTP, DNS) können durch eine sichere Verbindung getunnelt oder abgesichert werden (z.B. HTTPS, DoT).

1.1 Transport Layer Security (TLS)

TLS ist der Nachfolger von SSL (Secure Sockets Layer) und der De-facto-Standard für sichere Kommunikation im Internet (z.B. HTTPS).

1.1.1 Schutzziele und Eigenschaften

- **Vertraulichkeit:** Verschlüsselung der Payload (Nutzdaten).
- **Integrität:** Schutz vor Veränderung, z.B. durch HMAC.
- **Authentifikation:** Optional einseitig (nur Server, Standard im Web) oder beidseitig (Mutual TLS) mittels Zertifikaten (X.509).
- **Replay-Schutz:** Verhinderung des Wiedereinspielens alter Nachrichten durch **Nonces** (Number used once).
- **Perfect Forward Secrecy (PFS):** Die Kompromittierung eines Langzeitschlüssels (z.B. Server Private Key) führt *nicht* zur Entschlüsselung aufgezeichneter vergangener Sitzungen. Dies wird durch temporäre (ephemerale) Sitzungsschlüssel (z.B. via Diffie-Hellman) erreicht.

1.1.2 Historie und Versionen

- **SSL 1.0 - 3.0:** Veraltet und unsicher (POODLE Angriff auf SSL 3.0).
- **TLS 1.0 & 1.1:** Veraltet, nutzen schwache Hashfunktionen (MD5/SHA-1). Deprecated seit 2020/2021 (RFC 8996).
- **TLS 1.2 (2008):** Aktueller Standard. Einführung von AEAD (Authenticated Encryption) und SHA-256.
- **TLS 1.3 (2018):** Neueste Version.
  - **Verbesserungen:** Entfernung unsicherer Algorithmen (kein statisches RSA/DH mehr), 1-RTT Handshake (schneller), verpflichtendes PFS.

1.1.3 Cipher Suites

Eine Cipher Suite definiert die Algorithmen für eine TLS-Sitzung.
Aufbau Beispiel: `TLS_DHE_RSA_WITH_AES_128_GCM_SHA256`

1. **Protokoll:** TLS
2. **Key Exchange (Schlüsseltausch):** DHE (Ephemeral Diffie-Hellman) → sorgt für PFS.
3. **Authentication (Authentifizierung):** RSA (Signatur des Servers).
4. **Encryption (Verschlüsselung):** AES_128_GCM (Galois/Counter Mode → AEAD).
5. **MAC/PRF (Integrität):** SHA256.

1.1.4 Der TLS Handshake (vereinfacht TLS 1.2)

Ziel: Aushandlung von Parametern, Authentifizierung und Generierung des Session Keys.

1. **ClientHello:** Sendet Random R_C, SessionID, Liste unterstützter CipherSuites.
2. **ServerHello:** Wählt CipherSuite, sendet Random R_S.
3. **Zertifikatsaustausch:** Server sendet Zertifikat (Public Key) und Key-Exchange Parameter.
4. **Key Exchange:** Client und Server berechnen das **Pre-Master Secret**.
5. **Finished:** Beide Seiten senden einen MAC über alle bisherigen Handshake-Nachrichten, um Integrität sicherzustellen (Schutz vor Downgrade-Angriffen).

1.1.5 Angriffe auf TLS

Downgrade-Angriffe Ein Man-in-the-Middle (MitM) täuscht vor, dass eine Seite nur veraltete Versionen unterstützt, um die Verbindung auf unsichere Standards (z.B. SSL 3.0) herabzustufen.

- **POODLE:** Nutzt Fallback auf SSL 3.0 aus.
- **SLOTH:** Nutzt Kollisionen in schwachen Hashfunktionen (MD5/SHA-1) im Handshake (Transcript Collision), um sich als Kommunikationspartner auszugeben.
- **Gegenmaßnahme in TLS 1.3:** MD5/SHA-1 verboten, Signatur über gesamten Handshake.

Heartbleed (Implementierungsfehler) Kein kryptographischer Fehler, sondern ein **Buffer-Over-Read** in der OpenSSL-Bibliothek.

- **Funktionsweise:** Die Heartbeat-Extension erlaubt das Senden einer "Keep-Alive" Nachricht (z.B. String "Bird", Länge 4). Der Server soll dasselbe zurücksenden.
- **Der Fehler:** Der Angreifer sendet "Bird", gibt aber als Länge 64kB an. Der Server prüft die Länge nicht und kopiert 64kB aus seinem Arbeitsspeicher zurück an den Angreifer.
- **Folge:** Angreifer konnten Private Keys, Passwörter und Session-Daten auslesen.

1.2 IPsec (Internet Protocol Security)

IPsec sichert IP-Pakete auf Layer 3 ab. Es ist transparent für Anwendungen (im Gegensatz zu TLS). Es wird oft für VPNs (Virtual Private Networks) genutzt.

1.2.1 Architektur und Datenbanken

- **SA (Security Association):** Ein "Vertrag" zwischen zwei Partnern über Algorithmen und Schlüssel (unidirektional).
- **SPD (Security Policy Database):** Regelt, was mit Paketen passiert (Verwerfen, Durchlassen, Verschlüsseln).

1.2.2 Modi: Transport vs. Tunnel

Transportmodus

Einsatz: End-to-End Kommunikation (Host-zu-Host).
- Nur die **Payload** (z.B. TCP/UDP Segment) wird verschlüsselt/authentifiziert.
- Der originale IP-Header bleibt erhalten und sichtbar.
- *Nachteil:* Verbindungsmetadaten (Quell-/Ziel-IP) sind sichtbar.

```
[IP Header] [IPsec Header] [Verschlüsselte Payload]
```

Tunnelmodus

Einsatz: Site-to-Site VPN (Gateway-zu-Gateway) oder Remote Access VPN.
- Das **gesamte IP-Paket** (inkl. Header) wird verschlüsselt und in ein neues IP-Paket verpackt.
- Ermöglicht virtuelle Netzwerke (z.B. Zugriff auf internes Uni-Netz von zuhause).
- Versteckt die inneren IP-Adressen.

```
[Neuer IP Header (Gateway)] [IPsec Header] [Verschlüsselt:  Orig.  IP Header + Payload]
```

1.2.3 Protokolle: AH und ESP

1. **Authentication Header (AH):**
   - Bietet Integrität und Authentizität.
   - **Keine Verschlüsselung** (Vertraulichkeit).
   - Signiert auch Teile des IP-Headers → **Inkompatibel mit NAT** (da NAT IP-Adressen ändert und somit die Signatur bricht).
2. **Encapsulating Security Payload (ESP):**
   - Bietet Vertraulichkeit (Verschlüsselung), Integrität und Authentizität.
   - Schützt (im Tunnelmodus) das innere Paket komplett.
   - Standard für VPNs.

1.2.4 IKEv2 (Internet Key Exchange)

Dient dem Schlüsselaustausch und der Authentifizierung für IPsec (ähnlich dem TLS Handshake, läuft über UDP 500/4500).

- **Phase 1 (IKE_SA_INIT):** Aushandeln der Krypto-Parameter, Diffie-Hellman Austausch.
- **Phase 2 (IKE_AUTH):** Authentifizierung der Identitäten, Aufbau der IPsec SA.

1.3 Secure Shell (SSH)

SSH (Layer 7) dient primär dem sicheren Fernzugriff auf Kommandozeilen (Remote Shell), ersetzt unsichere Protokolle wie Telnet.

1.3.1 Funktionsweise und Authentifizierung

- **Trust on First Use (TOFU):** Der Client kennt den Server beim ersten Verbinden meist nicht. Der Server schickt seinen Public Key. Der User muss den Fingerprint bestätigen. Danach wird der Key gespeichert (in `known_hosts`).
- **Client-Authentifizierung:**
  - *Passwort:* Unsicher, sollte deaktiviert werden.

– *Public Key (empfohlen):* Der Client besitzt ein Schlüsselpaar. Der Public Key wird auf dem Server (`authorized_keys`) hinterlegt. Der Server stellt eine Challenge, die der Client mit dem Private Key signiert.

1.3.2 Härtung (Hardening)

- Root-Login verbieten.
- Nur Key-Based Authentication erlauben (keine Passwörter).
- Port ändern (Security by Obscurity, hilft aber gegen Massen-Scans).

1.4 Onion Routing / Tor

Tor dient der Anonymisierung von Verbindungen (Verschleierung von Wer kommuniziert mit Wem).

1.4.1 Funktionsprinzip

- Daten werden über mehrere Knoten (Relays) geleitet (typisch 3 Hops: Entry, Middle, Exit).
- **Zwiebelschalen-Prinzip:** Das Paket wird vom Sender mehrfach verschlüsselt. Jeder Knoten entfernt eine Schicht (entschlüsselt mit seinem Key) und kennt nur den direkten Vorgänger und den direkten Nachfolger. Kein Knoten kennt die gesamte Route (Sender bis Ziel).

1.4.2 Tor Onion Services (Hidden Services)

Ermöglicht das Anbieten von Diensten (z.B. Webseiten), ohne dass der Server seinen Standort (IP) preisgibt.

Verbindungsaufbau zu einem Onion Service:

1. **Setup:** Der Server wählt *Introduction Points* (Tor Nodes) und veröffentlicht diese zusammen mit seinem Public Key in einer *Distributed Hash Table* (Directory).
2. **Discovery:** Der Client erhält die Onion-Adresse (z.B. via Website), lädt die Infos aus dem Directory und verifiziert die Signatur.
3. **Rendezvous-Wahl:** Der Client wählt einen zufälligen *Rendezvous Point* und teilt diesem ein "One-Time-Secret" mit.
4. **Kontaktaufnahme:** Der Client sendet eine Nachricht (verschlüsselt mit dem Public Key des Servers) inkl. des gewählten Rendezvous Points und des Secrets an einen *Introduction Point* des Servers.
5. **Verbindung:** Der Server verbindet sich zum *Rendezvous Point* und sendet das Secret. Wenn die Secrets übereinstimmen, werden die Verbindungen zusammengeschaltet.

Ergebnis: Weder Client noch Server kennen die IP des anderen. Die Verbindung läuft über 6 Hops (3 vom Client, 3 vom Server).

1.4.3 Tor Adressen (v3)

- Format: `[base32-encoded-key].onion`
- Enthält den kompletten Public Key (Ed25519), eine Version und eine **Prüfsumme** (Checksum).
- *Vorteil Checksum:* Tippfehler werden sofort erkannt, bevor eine Verbindung versucht wird; Validierung ohne Directory möglich.

1.4.4 Grenzen der Anonymität

- **Globaler Angreifer:** Wer Eintritts- und Austrittsknoten überwacht, kann über Zeitkorrelation (Timing Analysis) Sender und Empfänger verknüpfen.
- **Metadaten:** Browser-Fingerprinting oder Nutzerverhalten können zur De-Anonymisierung führen.

- **Exit Nodes:** Der Exit Node sieht den unverschlüsselten Traffic zum Ziel (wenn kein End-to-End TLS genutzt wird).