

Sicherheit von Betriebssystemen – VO 10: Absicherung von Netzwerkverkehr

Paul Kalauner, Rafael Vrecar

Research Group for Industrial Software (INSO)

<https://www.inso-world.com>



RISE 
Research Industrial Systems Engineering



**FACHHOCHSCHULE
WIENER NEUSTADT**
University of Applied Sciences - Austria





Agenda

- Netzwerk-Kommunikation
- Android
- iOS
- Monster-in-the-Middle Attack
- Absicherungsmaßnahmen
- Testtechniken

Motivation

- heutzutage haben viele Apps eine Internet-Anbindung
- oftmals werden wichtige Informationen nachgeladen oder an einen Server geschickt
- es existieren mehrere Varianten, um den Netzwerkverkehr mitzuschneiden

Netzwerk-Kommunikation

Client-Server-Architektur (Recap.)

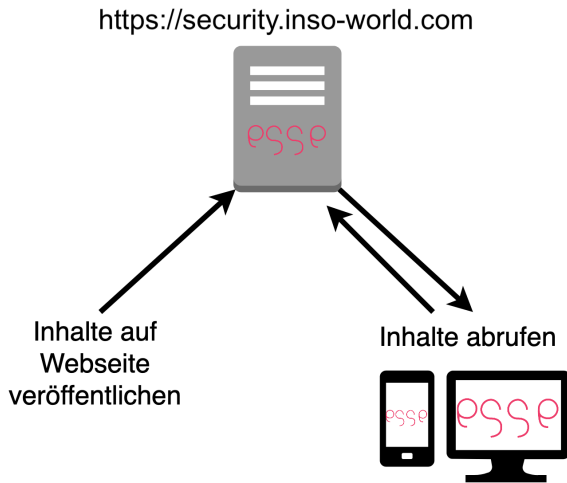
Server („backend“):

- speichert Daten & macht sie für andere Geräte öffentlich zugänglich
- führt Server-Code aus, um Anfragen der App zu bearbeiten & Daten in einer Datenbank zu speichern bzw. aus einer zu laden

Client („frontend“):

- was Benutzer:in sieht
- z.B. App-Benutzer:innenoberfläche, HTML-Webseite

Client-Server-Architektur – Grafische Darstellung (Recap.)



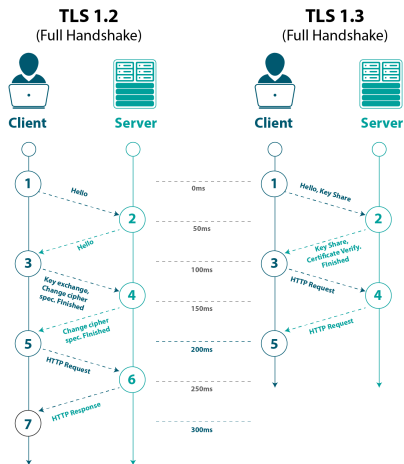
SSL (Recap.)

- = Secure Sockets Layer
- Standardtechnologie für Absicherung von Internet-Verbindungen & Schutz sensibler Daten
- verwendet Verschlüsselungsalgorithmen, um Daten während Übertragung zu kodieren
- wird nicht mehr verwendet (deprecated), wurde durch TLS ersetzt

TLS (Recap.)

- = Transport Layer Security
- aktualisierte Version von SSL, die höhere Sicherheit bietet
- aktuelle Version: seit 2018 TLS 1.3
- besteht aus TLS Handshake & TLS Record
 - TLS Handshake: sicherer Schlüsselaustausch und eine Authentisierung
 - TLS Record: verwendet im TLS Handshake ausgehandelten symmetrischen Schlüssel für sichere Datenübertragung
- für jede Verbindung wird neuer Sitzungsschlüssel (Session Key) ausgehandelt
 - die Sitzung kann auch wieder aufgenommen werden, falls sie unterbrochen wurde

TLS – Grafische Darstellung (Recap.)



Source: <https://www.appviewx.com/blogs/why-is-tls-1-3-better-and-safer-than-tls-1-2/>

Attacken auf SSL & TLS (Recap.)

- BEAST (Browser Exploit Against SSL/TLS)
- BREACH (Browser Reconnaissance & Exfiltration via Adaptive Compression of Hypertext)
- POODLE (Padding Oracle On Downgraded Legacy Encryption)
- SLOTH (Security Losses from Obsolete and Truncated Transcript Hashes)
- Heartbleed in OpenSSL

HTTP

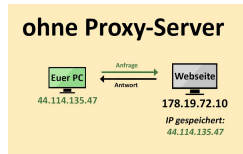
- = Hypertext Transfer Protocol
- zustandsloses Protokoll, zur Datenübertragung in IP-Netzwerken
- hauptsächlich für Übertragung von Internetseiten & Daten zwischen Webserver & Webbrowser verwendet
- Application Layer
- unverschlüsselt

HTTPS

- = Hyper Text Transfer Protocol *Secure*
- sichere Erweiterung von HTTP
- durch SSL/TLS-Zertifikat abgesichert
- Datei von Zertifizierungsstelle (CA) z.B. *Let's Encrypt* wird verwendet, um Besitz eines öffentlichen Schlüssels zu zertifizieren
- Webseiten, die SSL/TLS-Zertifikat installiert & konfiguriert haben, können HTTPS verwenden, um sichere Verbindung zu Server aufzubauen
- Daten werden verschlüsselt, bevor sie versendet werden

Proxyserver

- = „Server, der als Vermittler und Filter zwischen einem internen Servernetzwerk und dem Internet dient“



© Giga.de

(Vergleiche <https://www.duden.de/rechtschreibung/Proxyserver>) (Begriff) und

(Vergleiche <https://www.giga.de/extra/server/specials/was-ist-ein-proxy-server-und-warum-ist-er-so-wichtig-einfach-erklart/>) (Bild)

OSI-Modell

| | | |
|---|--------------------|--|
| 7 | Application Layer | Human-computer interaction layer, where applications can access the network services |
| 6 | Presentation Layer | Ensures that data is in a usable format and is where data encryption occurs |
| 5 | Session Layer | Maintains connections and is responsible for controlling ports and sessions |
| 4 | Transport Layer | Transmits data using transmission protocols including TCP and UDP |
| 3 | Network Layer | Decides which physical path the data will take |
| 2 | Data Link Layer | Defines the format of data on the network |
| 1 | Physical Layer | Transmits raw bit stream over the physical medium |

© Imperva

(Vergleiche <https://www.imperva.com/learn/application-security/osi-model/>)

Allgemeines

- Netzwerkzugriffe müssen explizit erlaubt und kontrolliert werden
- Nutzung sicherer Netzwerkprotokolle und -bibliotheken
- Unterstützung für Verschlüsselung, Timeouts und Verbindungsmanagement
- High-Level-HTTP-Clients vereinfachen sichere Implementierungen
- HTTPS für Webkommunikation verwenden
- Verschlüsselte und authentifizierte Netzwerkkommunikation einsetzen
- Absicherung auch auf Socket- und Transportebene
- Mobile Geräte nutzen häufig unsichere Netzwerke
- Unverschlüsselte Kommunikation birgt hohes Risiko

Android

Android: Allgemeines

- Netzwerkzugriffe erfordern explizite Berechtigungen im Manifest
- Sichere Android-APIs für Netzwerkkommunikation nutzen
- **HttpsURLConnection**: TLS, Timeouts, IPv6, Connection Pooling
- **Retrofit**: Komfortable HTTP-Client-Bibliothek
- **SSLSocket** für TLS-gesicherte Verbindungen
- Unsichere Inter-Process Communication (IPC) vermeiden
 - Keine IPC über **localhost** oder offene Netzwerkports
 - **INADDR_ANY** vermeiden (global erreichbar)
 - Android-IPC mit Authentifizierung nutzen (z. B. Services)
- Externen Daten nicht vertrauen
 - HTTP-Daten können manipuliert sein
 - Eingaben aus WebViews validieren
 - Intent-Daten aus externen Quellen prüfen

Network Security Configuration – Hauptfunktionen

- Steuerung, welchen Zertifizierungsstellen (CAs) vertraut wird
- Einschränkung auf ausgewählte oder selbstsignierte Zertifikate
- Separate Einstellungen für Debug- und Produktionsumgebung
- Schutz vor unbeabsichtigtem Klartext-Datenverkehr

Network Security Configuration – Zertifikate

- Certificate Pinning zur Einschränkung auf bestimmte Zertifikate
- Unterstützung für benutzerdefinierte Zertifikate
- Sinnvoll in kontrollierten Umgebungen (z. B. interne Netze)
- Explizite Freigabe von Benutzerzertifikaten erforderlich

Network Security Configuration – Konfiguration

- Netzwerksicherheitsregeln deklarativ konfigurierbar
- Keine Änderungen am Anwendungscode erforderlich
- Zentrale Konfiguration über externe XML-Datei
- Feingranulare Regeln pro App und Domäne möglich

Network Security Configuration – Beispiel

```
1 <network-security-config>
2
3   <base-config cleartextTrafficPermitted="false">
4     <trust-anchors>
5       <certificates src="system"/>
6     </trust-anchors>
7   </base-config>
8
9   <domain-config cleartextTrafficPermitted="false">
10    <domain>example.com</domain>
11    <pin-set>
12      <pin digest="SHA-256">
13        AbCdEfGhIjKlMnOpQrStUvWxYz1234567890abcd=
14      </pin>
15    </pin-set>
16  </domain-config>
17
18  <debug-overrides>
19    <trust-anchors>
20      <certificates src="user" />
21    </trust-anchors>
22  </debug-overrides>
23
24 </network-security-config>
```

Zertifikate in Android

Hauptfunktionen:

- Geräte verfügen über einen vordefinierten Satz vertrauenswürdiger Zertifikate
- Unter Android sind diese in `/system/etc/security/cacerts/` zu finden
- Systemzertifikate sind fest im Betriebssystem verankert
- Änderungen durch Nutzer:innen normalerweise nicht möglich
- Schutz durch schreibgeschützte Systembereiche

iOS

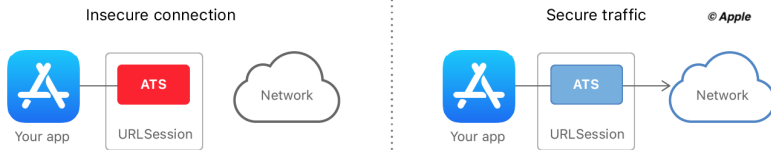
iOS: Allgemeines

- iOS unterstützt TLS-Versionen 1.0, 1.1, 1.2 und 1.3
- TLS unterstützt AES-128, AES-256 & bevorzugt Cipher-Suites mit *Forward Secrecy*
- u. a. „Safari“, „Kalender“, „Mail“ verwenden TLS für einen verschlüsselten Kommunikationskanal
- High-Level-APIs (z.B. **CFNetwork**) erleichtern, TLS in Apps zu verwenden
- Low-Level-APIs (z.B. **Network.framework**) bieten präzise Einstellungsmöglichkeiten

App Transport Security

- App Transport Security (ATS) = Netzwerksicherheitsfunktion
- System erzwingt ATS, wenn *URL Loading System* verwendet wird
- Apps müssen Netzwerkverbindungen durch das TLS-Protokoll mit zuverlässigen Zertifikaten und Chiffren sichern
- blockiert Verbindungen, welche Mindestsicherheitsanforderungen nicht erfüllen
- Ausnahmen können hinzugefügt werden, um Anforderungen zu lockern (*nicht empfehlenswert*)

App Transport Security – Grafische Darstellung



Monster-in-the-Middle Attack

Monster-in-the-Middle Attack

- auch bekannt als „Man-in-the-Middle attack“ (MiTM)
- Methode, um Kommunikation zwischen zwei Systemen abzufangen
- wenn Verbindung abgefangen wurde, fungiert Angreifer:in als Proxy ⇒ Daten können gelesen, eingefügt & verändert werden

- Apps oft anfällig, weil ...
 - falsche Konfigurationen bzw. Nicht-Berücksichtigen offizieller Dokumentation
 - Akzeptanz nicht vertrauenswürdiger TLS-Zertifikate
 - Verwendung schwächerer TLS-Modi

Absicherungsmaßnahmen

Allgemeines

- Developer:in:
 - offizielle Entwickler:innenrichtlinien (z.B. ATS, Network Security Configuration) befolgen
 - State-of-the-Art verwenden (z.B. TLS)
 - regelmäßiges Updaten (Services & Dependencies aktuell halten)
- User:in:
 - nicht vertrauenswürdige Zertifikate ablehnen
 - Apps nur aus offiziellen Quellen (z.B. App Store) installieren

(Vergleiche <https://developer.android.com/topic/security/best-practices>)

Certificate Pinning 1/3

- Host wird mit Zertifikat oder öffentlichem Schlüssel verknüpft & an Gerät „angeheftet“
- kann MitM-Angriffe verhindern
- ab iOS 14: erwartete Schlüssel in **Info.plist** hinterlegbar
 - Schutzmechanismus greift nur für Verbindungen über *URL Loading System*
 - Einschränkungen greifen nicht, wenn Low-Level-Methoden (z.B. *Network Framework*) verwendet werden
- seit Android 7.0 lassen sich Sicherheitseinstellungen & Zertifikate/Schlüssel auf ähnliche Weise per Konfiguration festlegen
- Android rät von Verwendung (z.B. aufgrund von künftigen Änderungen an Serverkonfiguration) ab

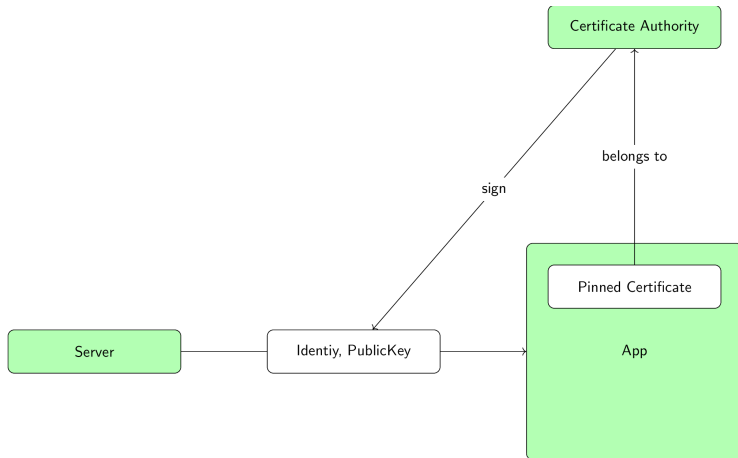
Certificate Pinning 2/3

- standardmäßig werden die im Betriebssystem hinterlegten Zertifikate genutzt
- oftmals greifen Applikationen aber aus unterschiedlichen Gründen auf eigene Zertifikate zurück
 - Zertifikate sind selbstsigniert
 - den vorhandenen Zertifikaten wird nicht getraut, etc.
- es bestehen verschiedene Möglichkeiten vertrauenswürdige Zertifikate zu hinterlegen
 - Ablage als externe Ressource
 - Herunterladen zum Start der App
 - Manuelles Hinzufügen während des Starts, etc.

Certificate Pinning 3/3

- ein Beispiel für die Verwendung von eigenen Zertifikaten ist die Sonos App
- bei Programmstart werden initial einige Pfade kontrolliert und anschließend alle Zertifikate eingelesen
- der erste Request lädt anschließend eine Liste an vertrauenswürdigen CAs herunter
 - geschieht in einer geteilten Bibliothek, dadurch schwer nachvollziehbar wie die Liste weiterverarbeitet wird

Certificate Pinning – Grafische Darstellung



Testtechniken

Testtechniken: HTTP(S)-Verkehr abfangen

z.B. mit Burp (<https://portswigger.net/burp>):

- System-Proxy auf dem mobilen Gerät konfigurieren ⇒ HTTP(S)-Verkehr über einen „Abfang“-Proxy auf eigenen Host-Computer umleiten
- verfügbare serverseitigen APIs abbilden
- Einblicke in Kommunikationsprotokoll gewinnen
- Anfragen wiederholen & manipulieren, um auf serverseitige Schwachstellen zu testen

Testtechniken: Abfangen auf Netzwerk-Ebene

z.B. mit Wireshark (<https://www.wireshark.org>):

- App verwendet Standardbibliotheken & Kommunikation über HTTP ⇒ dynamische Analyse mit „Abhör“-Proxys
- Datenverkehr über den Host-Computer leiten & mit-sniffen

Zusammenfassung

- Moderne Apps kommunizieren häufig über Netzwerke und übertragen dabei sensible Daten
- Client-Server-Architekturen bilden die Grundlage der Netzwerkkommunikation
- TLS ermöglicht verschlüsselte und authentifizierte Verbindungen
- HTTPS schützt HTTP-Verkehr durch den Einsatz von TLS und Zertifikaten
- Plattformen wie Android und iOS bieten Mechanismen zur Absicherung des Netzwerkverkehrs (z.B. ATS, Network Security Configuration, Certificate Pinning)

Literaturverzeichnis 1/6

- Pohlmann, N. (2019). Transport Layer Security (TLS)/Secure Socket Layer (SSL). In: Cyber-Sicherheit. Springer Vieweg, Wiesbaden.
https://doi.org/10.1007/978-3-658-25398-1_11
- Vijay Kumar Velu. Mobile Application Penetration Testing. Birmingham: Packt Publishing Ltd, 2016. ISBN: 978-1-78588-337-8.
- Fluffy.es – <https://fluffy.es/introduction-to-client-server>
- Websecurity.Digicert.com – <https://www.websecurity.digicert.com/de/de/security-topics/what-is-ssl-tls-https>
- Hostinger.com –
<https://www.hostinger.com/tutorials/what-is-ssl-tls-https>

Literaturverzeichnis 2/6

- Digicert.com – <https://www.digicert.com/how-tls-ssl-certificates-work>
- IETF.org – <https://datatracker.ietf.org/doc/html/rfc8446.html>
- BREACH Attack – <http://www.breachattack.com>
- POODLE – <https://heise.de/-2425250>
- SLOTH – <http://dx.doi.org/10.14722/ndss.2016.23418>
- Heartbleed – <https://heartbleed.com>

Literaturverzeichnis 3/6

- Heise.de (1) – https://www.heise.de/select/ix/2018/8/1533435196337811/ix.0818.110-116.qxp_table_3639.html
- W3.org – <https://www.w3.org/Protocols/>
- Protonmail.com – <https://protonmail.com/blog/tls-ssl-certificate/>
- Letsencrypt.org – <https://letsencrypt.org>
- Mobile-Security.Gitbook.io – <https://mobile-security.gitbook.io/mobile-security-testing-guide/general-mobile-app-testing-guide/0x04f-testing-network-communication>

Literaturverzeichnis 4/6

- Heise.de (2) – <https://heise.de/-5048975>
- OWASP.org – https://owasp.org/www-community/attacks/Manipulator-in-the-middle_attack
- Developer.Android.com
 - <https://developer.android.com/training/articles/security-ssl>
 - <https://developer.android.com/training/articles/security-tips>
 - <https://developer.android.com/training/basics/network-ops/connecting>
 - <https://developer.android.com/training/articles/security-config>

Literaturverzeichnis 5/6

- Developer.Apple.com
 - https://developer.apple.com/documentation/security/preventing_insecure_network_connections
 - <https://developers.google.com/admob/ios/app-transport-security>
 - https://developer.apple.com/documentation/foundation/url_loading_system
 - <https://support.apple.com/de-at/guide/security/sec100a75d12/web>

Literaturverzeichnis 6/6

- Portswigger.net

- `https:`

- `//portswigger.net/support/configuring-an-ios-device-to-work-with-burp`

- `https://portswigger.net/support/`

- `installing-burp-suites-ca-certificate-in-an-ios-device`

Vielen Dank!

<https://establishing-security.at/>