esse

ESSE Mobile Security – VO 4: Absicherung von Netzwerkverkehr

Raphael Kiefmann, Paul Kalauner,
Vanessa Hohenegger, Rafael Vrecar,
Thomas Grechenig

INSO – Industrial Software
Institut für Information Systems Engineering | Fakultät für Informatik | Technische Universität Wien

Agenda

- **Netzwerk-Kommunikation**
  - Client-Server-Architektur
  - SSL
  - TLS
  - Attacken auf SSL & TLS
  - HTTP
  - HTTPS
  - Proxyserver
  - OSI-Modell
- **Android**
  - Allgemeines
  - Network Security Configuration
- **iOS**
  - Allgemeines
  - App Transport Security
- **Monster-in-the-Middle Attack**
- **Absicherungsmaßnahmen**
  - Allgemeines
  - Certificate Pinning
- **Testtechniken**
- **Demo**

Netzwerk-Kommunikation

Client-Server-Architektur (Recap.)

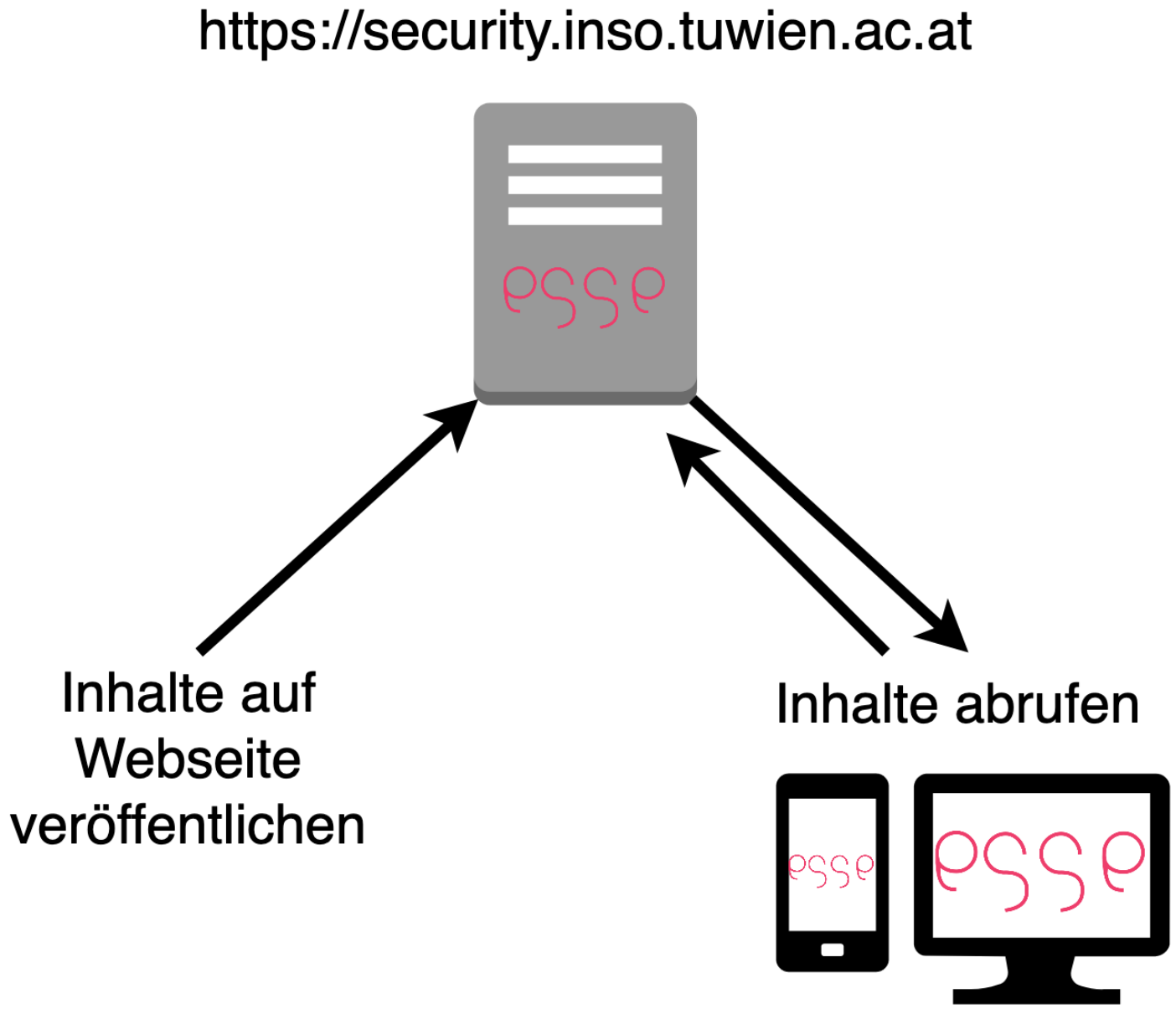
Server („backend“):

- 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.)



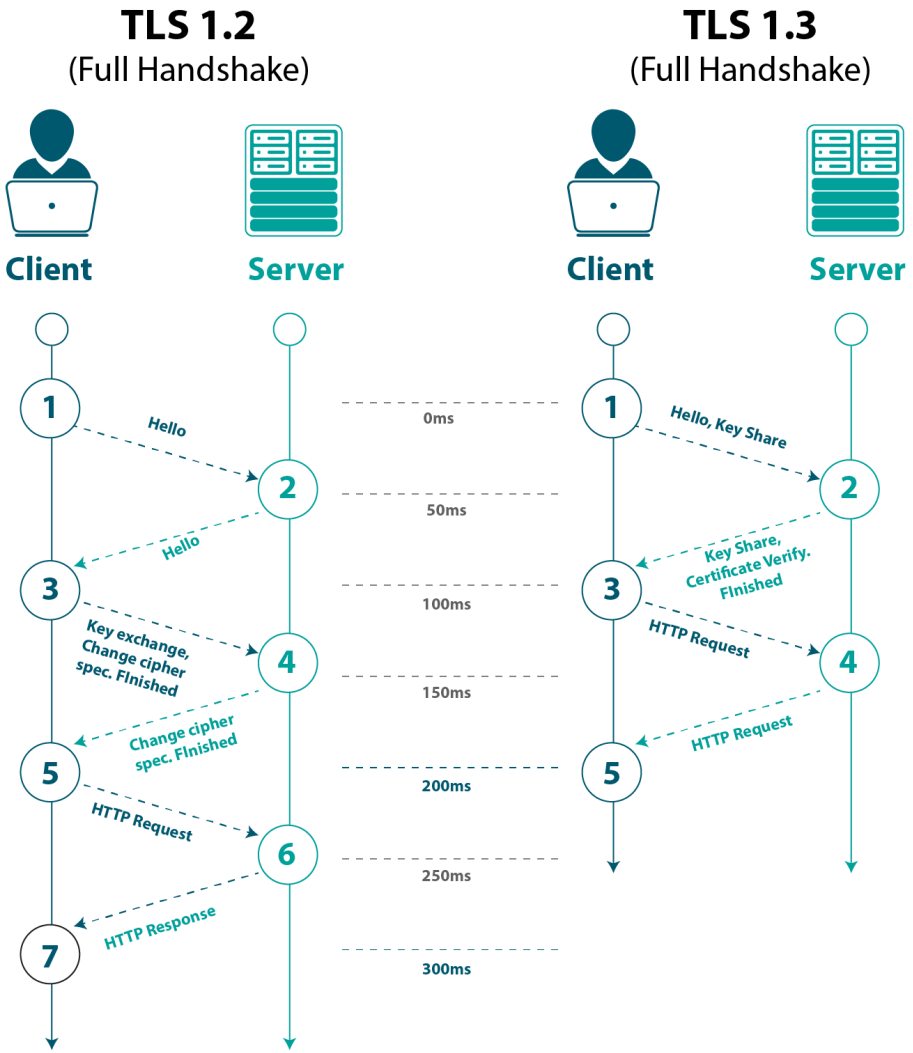
SSL (Recap.)

- = 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

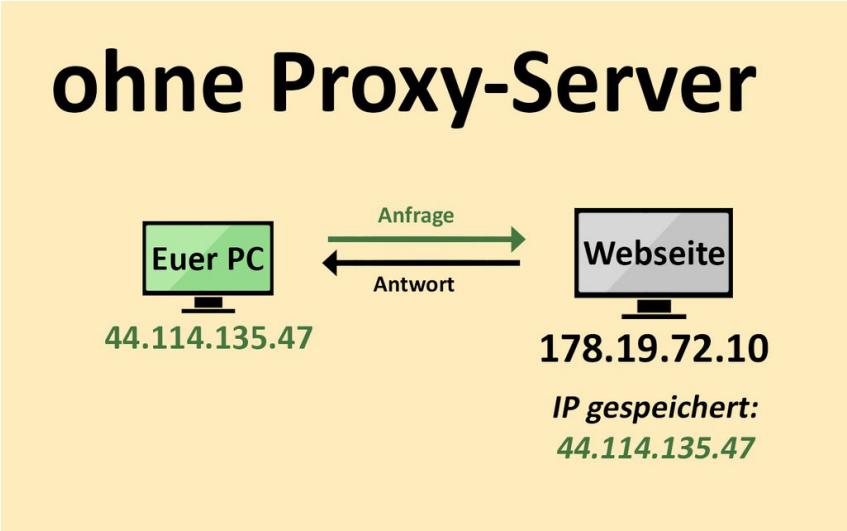
- = Hypertext Transfer Protocol
- zustandsloses Protokoll, zur Datenübertragung in IP-Netzwerken
- hauptsächlich für Übertragung von Internetseiten & Daten zwischen Webserver & Webbrowser verwendet
- Application Layer
- unverschlüsselt

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

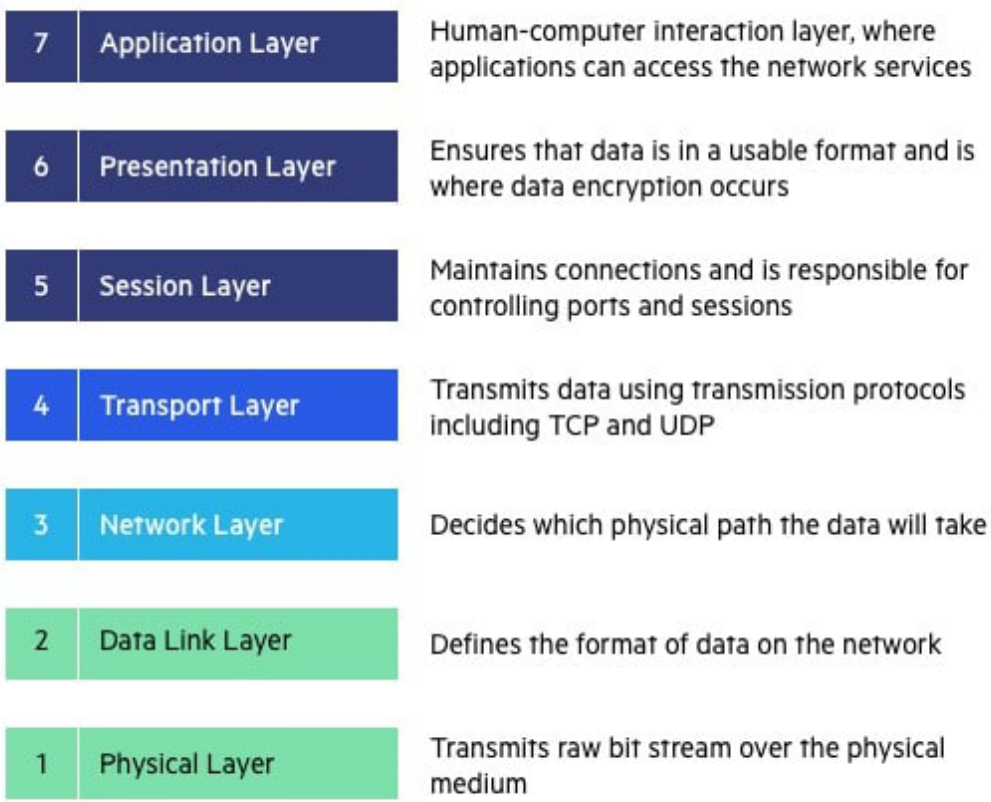
- = *„Server, der als Vermittler und Filter zwischen einem internen Servernetzwerk und dem Internet dient“*



© 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-erklaert/) (Bild)

OSI-Modell



© Imperva

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

Android

Allgemeines 1/3

- Manifest muss Berechtigungen enthalten, um Netzwerkoperationen in App durchführen zu können
- HttpsURLConnection-Client (`https://developer.android.com/reference/javax/net/ssl/HttpsURLConnection`) unterstützt: TLS, Streaming-Uploads und -Downloads, konfigurierbare Timeouts, IPv6 und Verbindungspooling
- Retrofit (`https://square.github.io/retrofit`): HTTP client library
- Android empfiehlt für sensible Daten geeignete Protokolle zu verwendet, wie z. B. HttpsURLConnection für sicheren Webverkehr
- Überall, wo der Server HTTPS unterstützt, sollte HTTPS verwendet werden, da sich mobile Geräte häufig mit ungesicherten Netzwerken verbinden, z. B. mit öffentlichen Wi-Fi-Hotspots

Allgemeines 2/3

- Authentifizierte, verschlüsselte Kommunikation auf Socket-Ebene kann mit der Klasse SSLSocket (`https://developer.android.com/reference/javax/net/ssl/SSLSocket`) leicht implementiert werden.
- Angesichts der Häufigkeit, mit der sich Android-Geräte über Wi-Fi mit ungesicherten drahtlosen Netzwerken verbinden, wird die Verwendung eines sicheren Netzwerks für alle Anwendungen, die über das Netzwerk kommunizieren, dringend empfohlen.

Allgemeines 3/3

- Einige Anwendungen verwenden Localhost-Netzwerkschnittstellen für den Umgang mit sensiblen IPC. Sie sollten diesen Ansatz nicht verwenden, da diese Schnittstellen für andere Anwendungen auf dem Gerät zugänglich sind. Verwenden Sie stattdessen einen Android-IPC-Mechanismus, bei dem eine Authentifizierung möglich ist, z. B. mit einem Dienst. Das Binden an INADDR_ANY ist schlechter als die Verwendung von Loopback, da Ihre Anwendung dann Anfragen von überall her erhalten kann.
- Stellen Sie sicher, dass Sie Daten, die von HTTP oder anderen unsicheren Protokollen heruntergeladen werden, nicht vertrauen. Dies schließt die Validierung von Eingaben in WebView und alle Antworten auf Intents ein, die über HTTP ausgegeben werden.

Network Security Configuration Hauptfunktionen

- Benutzer:innendefinierte Vertrauensanker: Anpassen, welchen Zertifizierungsstellen (CA) für die sicheren Verbindungen einer App vertraut wird. Beispielsweise das Vertrauen in bestimmte selbstsignierte Zertifikate oder das Einschränken der öffentlichen CAs, denen die App vertraut.
- Nur-Debugging-Überschreibungen: Sicheres Debuggen sicherer Verbindungen in einer App ohne zusätzliches Risiko für die installierte Basis.
- Opt-in für Klartext-Datenverkehr: Schützen Apps vor versehentlicher Verwendung von Klartext-Datenverkehr.

Network Security Configuration Zertifikate

- Certificate Pinning: Beschränken sichere Verbindung einer App auf bestimmte Zertifikate
- vertrauenswürdige Zertifikate können auch von Benutzer:innen hinzugefügt werden
  - sinnvoll in bestimmten Umgebungen in denen zum Beispiel selbstsignierte Zertifikate zum Einsatz kommen
- vor Android Nougat haben Apps allen Benutzer:innen-Zertifikaten vertraut. Mit Android Nougat musste explizit zugestimmt werden, dass Zertifikaten vertraut wird

Network Security Configuration

- Apps können ihre Netzwerksicherheitseinstellungen in einer sicheren, deklarativen Konfigurationsdatei (dem Manifest) – ohne App-Code zu ändern – anpassen.
- XML Datei
  - Eintrag in App-Manifest einfügen, um auf diese Datei zu verweisen.
- Diese Einstellungen können für bestimmte Domänen und für eine bestimmte App konfiguriert werden.

Zertifikate in Android

Hauptfunktionen:

- Auf Android Geräten ist im Normalfall ein Satz an vertrauenswürdigen Zertifikaten hinterlegt
  - Die Zertifikate sind in **/system/etc/security/cacerts/** zu finden
- die **/system** Partition ist als Härtungsmaßnahme `read-only` gemountet
  - Die Zertifikate können somit auf einem normalen Gerät nicht abgeändert werden

iOS

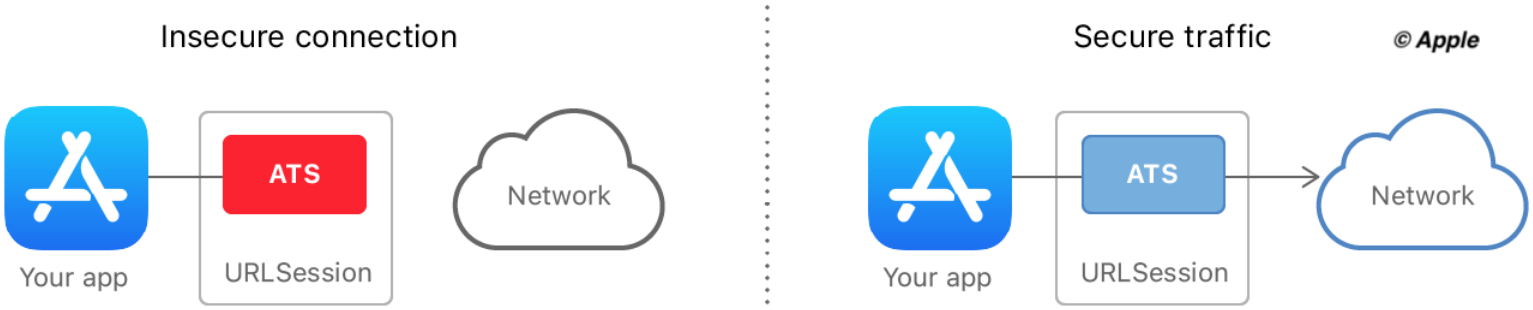
iOS: Allgemeines

- 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
  $\Rightarrow$ Daten können gelesen, eingefügt & verändert werden

- Apps oft anfällig, weil ...
  - falsche Konfigurationen bzw. Nicht-Berücksichtigen offizieller Dokumentation
  - Akzeptanz nicht vertrauenswürdiger TSL-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

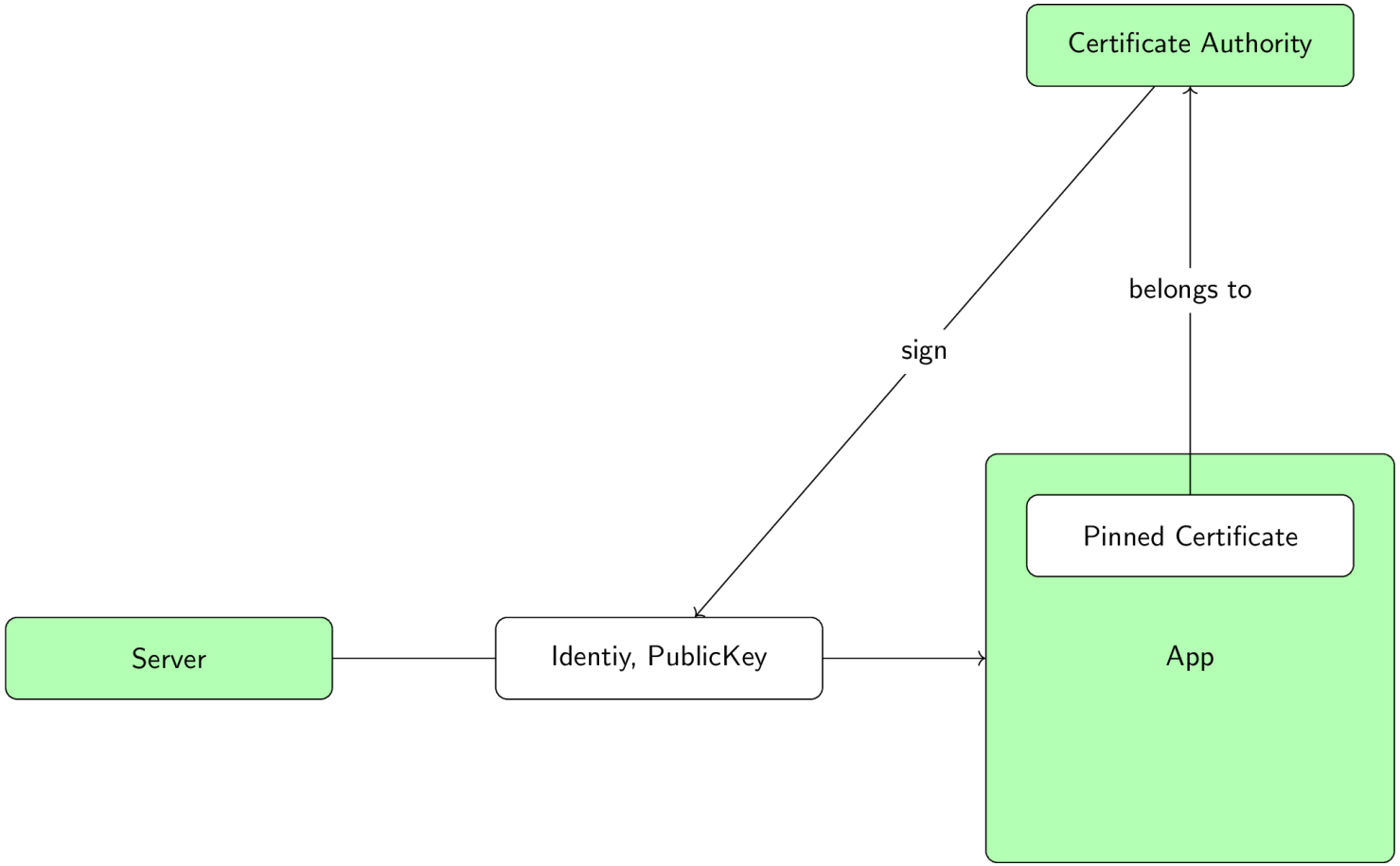
Certificate Pinning 2/3

- 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 hinterlegten
  - 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 $\Rightarrow$ 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
  $\Rightarrow$ dynamische Analyse mit „Abhör“-Proxys
- Datenverkehr über den Host-Computer leiten & mit-sniffen
- Hooking oder Code Injection auf iPhones mit Jailbreak

Demo

Demo

- Burp konfigurieren
  (https://portswigger.net/support/configuring-an-ios-device-to-work-with-burp)

- Burp-Suite CA-Zertifikat installieren
  (https://portswigger.net/support/installing-burp-suites-ca-certificate-in-an-ios-device)

- Certificate Pinning deaktivieren
  (https://github.com/sensepost/objection)

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 – abgerufen: 27.04.2022
- Websecurity.Digicert.com – https://www.websecurity.digicert.com/de/de/security-topics/what-is-ssl-tls-https – abgerufen: 27.04.2022
- Hostinger.com – https://www.hostinger.com/tutorials/what-is-ssl-tls-https – abgerufen: 27.04.2022

Literaturverzeichnis 2/6

- Digicert.com – https://www.digicert.com/how-tls-ssl-certificates-work – abgerufen: 27.04.2022
- IETF.org – https://datatracker.ietf.org/doc/html/rfc8446.html – abgerufen: 27.04.2022
- BREACH Attack – http://www.breachattack.com – abgerufen: 27.04.2022
- POODLE – https://heise.de/-2425250 – abgerufen: 27.04.2022
- SLOTH – http://dx.doi.org/10.14722/ndss.2016.23418 – abgerufen: 27.04.2022
- Heartbleed – https://heartbleed.com – abgerufen: 27.04.2022

Literaturverzeichnis 3/6

- Heise.de (1) – https://www.heise.de/select/ix/2018/8/1533435196337811/ix.0818.110-116.qxp_table_3639.html – abgerufen: 27.04.2022
- W3.org – https://www.w3.org/Protocols/ – abgerufen: 27.04.2022
- Protonmail.com – https://protonmail.com/blog/tls-ssl-certificate/ – abgerufen: 27.04.2022
- Letsencrypt.org – https://letsencrypt.org – abgerufen: 27.04.2022
- Mobile-Security.Gitbook.io – https://mobile-security.gitbook.io/mobile-security-testing-guide/general-mobile-app-testing-guide/0x04f-testing-network-communication – abgerufen: 27.04.2022

Literaturverzeichnis 4/6

- Heise.de (2) – https://heise.de/-5048975 – abgerufen: 27.04.2022
- OWASP.org – https://owasp.org/www-community/attacks/Manipulator-in-the-middle_attack – abgerufen: 27.04.2022
- Developer.Android.com – abgerufen (alle): 27.04.2022
  - 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 – abgerufen (alle): 27.04.2022
  - 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 – abgerufen (beide): 27.04.2022
  - 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/