esse

ESSE Einführung in Security – VO 11: Mobile Security

Paul Kalauner, Raphael Kiefmann, Rafael Vrecar, Vanessa Hohenegger, Christian Schanes

24W

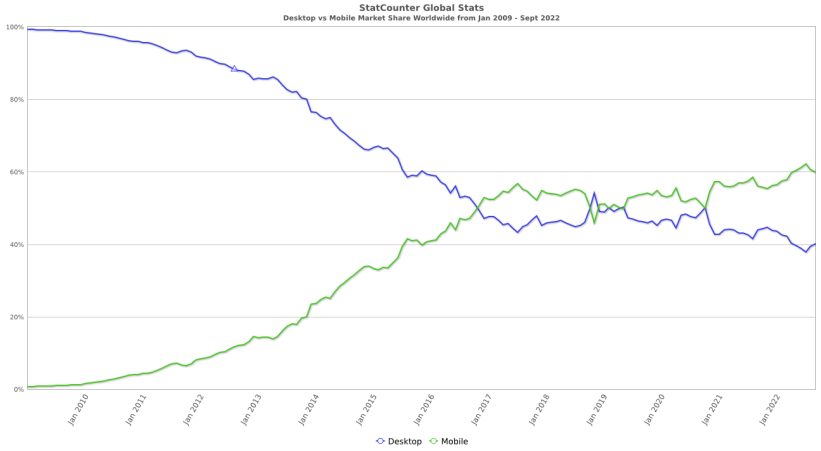
ESSE (Establishing Security) – IT Security Research Team
Research Group for Industrial Software (INSO)
https://establishing-security.at/

Agenda

- Motivation Mobile Security
- Android
  - Grundlagen & Allgemeines
  - Apps (Entwicklung, Verbreitung, Aufbau)
  - Threatmodell
  - Sicherheitsmodell
- iOS
  - Grundlagen & Allgemeines
  - Integritätssicherung
  - Apps (Entwicklung, Verbreitung, Aufbau)
  - Datenspeicher
- Ausblick, Literatur & Zusammenfassung

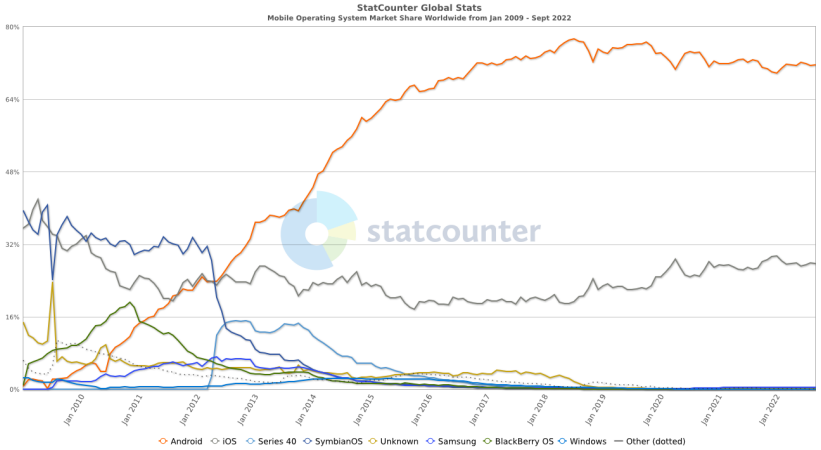
Motivation Mobile Security

Desktop vs. mobile Betriebssysteme



(Vergleiche `https://gs.statcounter.com/platform-market-share`)

Marketshare mobiler Betriebssysteme



(Vergleiche `https://gs.statcounter.com/os-market-share`)

Beispiele für sensible Daten auf mobilen Geräten

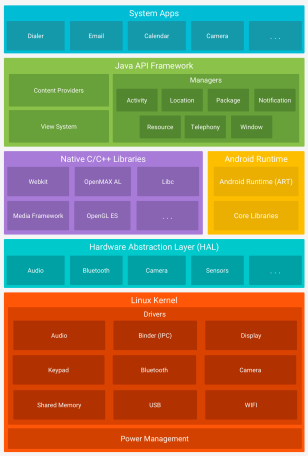
- SMS, E-Mail, Kontakte und Telefonie
- Standort, Kamera und Mikrofon
- Zugangsdaten
- 2FA Tokens
- Datamining (Fitness, persönliches Monitoring)
- *Internet of Things* Steuerung

Android: Grundlagen & Allgemeines

Allgemein

- basierend auf Linux
- weitere Security-Maßnahmen durch SELinux
- Open Source: `https://source.android.com`
- *aber:* oftmals proprietäre Erweiterungen durch Smartphone(komponenten)-Hersteller

Schematischer Aufbau von Android



(Vergleiche `https://developer.android.com/guide/platform`)

Entwicklung von Android und seinen Applikationen 1/2

- Verwendung unterschiedlicher Sprachen zur Entwicklung von Android und Android Apps
- IDE: Android Studio – `https://developer.android.com/studio`
- Entwicklung des Android-Systems zumeist in Systemprogrammiersprachen
  - C/C++
  - Rust – `https://play.rust-lang.org`
  - aber auch Java und Kotlin werden verwendet

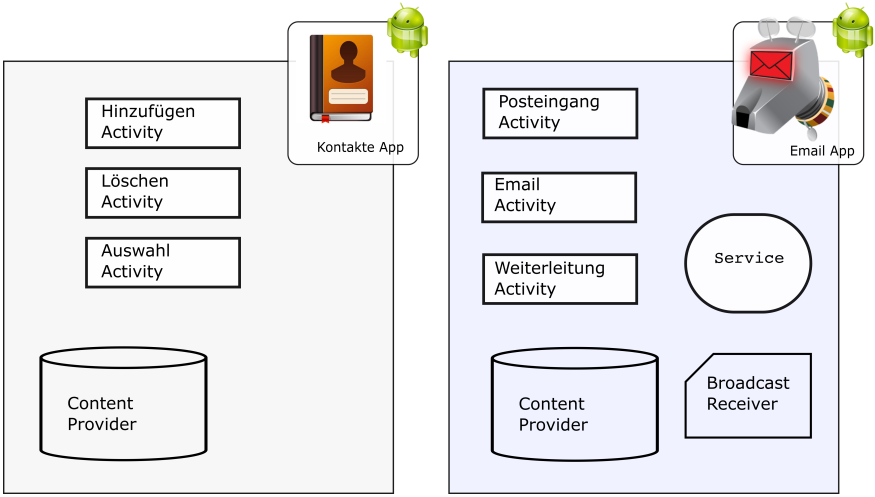
Entwicklung von Android und seinen Applikationen 2/2

Entwicklung von nativen Apps erfolgte zuerst in Java und dann Kotlin – `https://play.kotlinlang.org`

- mittlerweile existieren verschiedene Frameworks und Bibliotheken, welche die Entwicklung in anderen Programmiersprachen erlauben, z.B. React Native oder Flutter
- oftmals werden auch Shared Objects, die über das Java Native Interface (JNI) verfügbar sind, integriert

Android: Apps (Entwicklung, Verbreitung, Aufbau)

Komponenten-Übersicht

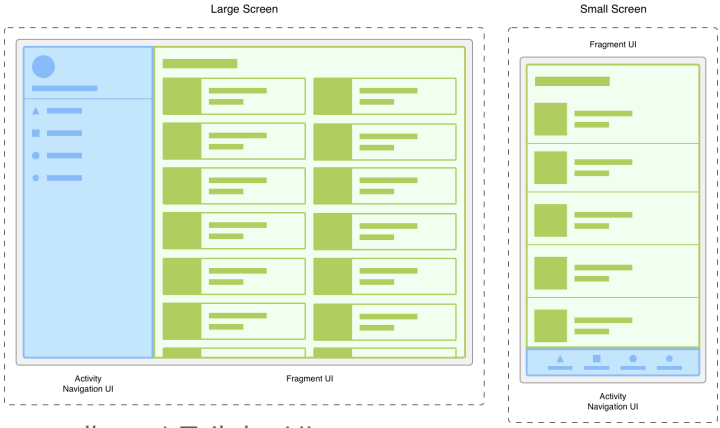


- App besteht aus Komponenten: `Activity`, `Service`, `BroadcastReceiver`, `ContentProvider`

Activity

- „Entry Point“ einer Android-Applikation
- beinhaltet Fragments und Layouts
- muss im Android-Manifest deklariert werden
- heute: meist einzige Activity, mehrere Fragments

Fragment



- (wiederverwendbarer) Teil der UI
- meist ein Fragment pro „Seite“

(Vergleiche `https://developer.android.com/guide/fragments`)

Service

- für langlebige Hintergrund-Operationen
- Musik-Player, Downloads etc.
- Vordergrund-Services
  - dauerhaft sichtbar für Nutzer:innen (Benachrichtigung)
  - Ausführung auch, wenn Nutzer:innen andere App verwenden
- Hintergrund-Services
  - „unsichtbar“ für Nutzer:innen
  - limitierte Funktionalität

Content Provider

- verwaltet Zugriff auf App-Daten
- ermöglicht Zugriff auf Daten anderer Apps
- ermöglicht es, Daten für andere Apps zugänglich zu machen

Broadcast Receiver

- ermöglicht Reaktion auf globale Events (z.B. Erhalt einer SMS)
- kann als weiterer „Entry Point“ einer App dienen
- z.B. Öffnen der App wenn Kamerataste gedrückt wurde

Android Package Format (APK)

- ZIP-Datei
- vollständige, installationsbereite Android-Applikation
- aus installierten Apps extrahierbar
- APKs können von unbekannten Quellen installiert werden (Sideloading)
  - USB
    - Android Debug Bridge
    - `https://developer.android.com/studio/command-line/adb`
    - `adb install <path_to_apk>`
  - Datei auf Smartphone (Download etc.)

Hauptinhalte einer APK

- `AndroidManifest.xml`: Deklariert App-Name, Komponenten etc.
- `lib/`: Libraries und kompilierter, plattformabhängiger Code
- `res/`: Ressources (z.B. Icons, Layouts etc.)
- `assets/`: Assets (z.B. Texturen)
- `classes.dex`: Im DEX-Format kompilierte Klassen

Dalvik EXecutable Format (DEX)

- beinhaltet Dalvik Byte Code
- konvertierbar in Smali (Human-Readable Representation)
  - smali/baksmali – `https://github.com/JesusFreke/smali`
  - Apktool – `https://ibotpeaches.github.io/Apktool`

Java Code:

```
int x = 42
```

Dalvik Byte Code:

```
13 00 2A 00
```

Smali Repräsentation:

```
const/16 v0, 42
```

Android Runtime (ART)

- führt DEX Byte Code aus
- seit Android 5 (Ersetzt Dalvik VM)
- Ahead-of-time (AOT) Kompilierung (bei Installation)
- sandboxed (Eine ART Instanz pro App)
- seit Android 7: „Hybrid“-Lösung AOT + JIT

Android: Threatmodell

Mögliche Angriffe

- Forensic Attacks
- Network-based Attacks
- Code Execution Attacks
- Web-based Attacks
- Physical Proximity Attacks (USB, NFC)
- ...

Was eine App nicht ermöglichen sollte

- Störung anderer Apps
- Datendiebstahl, Spionage
- unerwartete Kosten verursachen
- Denial of Service (z.B. Notruf)
- auf alle OS-Funktionen (z.B. Mikrofon) zugreifen können
- ...

Dekompilierung

- Byte Code vergleichsweise einfach dekompilierbar
- Dalvik Byte Code (`classes.dex`) kann in Java Byte Code (JAR-Datei) konvertiert werden

  ⇒ dex2jar – `https://github.com/pxb1988/dex2jar`
- resultierende JAR-Datei kann mit üblichen Java Decompilation Tools analysiert werden

  ⇒ jd-gui – `https://java-decompiler.github.io`
- Shortcut: DEX to Java Decompiler

  ⇒ jadx – `https://github.com/skylot/jadx`

Repackaging

- bekannte App verwenden
- disassemblieren und Payload einfügen
- reassemblieren und in (in)offiziellem Store veröffentlichen
- die *umverpackte* App enthält oft Malware mit verschiedenen Funktionen
  - Diebstahl von Informationen
  - Diebstahl von Zugangsdaten
  - Premium-Rate Anrufe und SMS
  - Spam via SMS und E-Mail
  - Erpressung des Opfers

Android: Sicherheitsmodell

Sandboxing

- Apps laufen in eigener VM isoliert durch OS
- jede App bekommt eigenen Linux User `a<number>_a<packagenumber>`
- Verzeichnis in `/data/user/<usernumber>/<packagename>`
- durch Linux-Berechtigungen geschützt
- Mandatory Access Control (SELinux)

Android Berechtigungen

- Berechtigungen steuern Zugriff auf Ressourcen
  - Kamera und Mikrofon
  - Kontakte und Telefonie
  - Location Services
  - ...
- explizites Einverständnis
- Einhaltung durch OS gegeben
- feingranularer und widerrufbar ab Android 6

Signierung von Applikationen

- APKs sind mit selbstsignierten Zertifikaten signiert
- Apps im Play Store können nur mit selbem Zertifikat aktualisiert werden
- allerdings: Code kann während Laufzeit nachgeladen werden
  - via HTTP(S): Man-in-the-Middle möglich
  - vom lokalen Dateisystem: Potenziell von anderen Apps veränderbar
  - maliziöse App mit selbem Package-Namen könnte bereits installiert sein

⇒ Signierung liefert keine Information über ausgeführten Code

Android App Bundle (AAB)

- Verwendung ab August 2021 verpflichtend
- App Bundle wird mit Upload Key signiert und hochgeladen
- Google signiert AAB mit eigenem Key und generiert APK
  - APKs sind zugeschnitten auf verschiedene Geräte (Bildschirmauflösungen, Architekturen, ...)
  - bei Verlust des Upload Keys kann neuer Key generiert werden

App Hardening

- Root/Jailbreak Detection
- Identifier Obfuscation (Proguard)
- String Obfuscation, Class Encryption, Junk Code Insertion, Call Graph Flattening
- Native Code vs. Byte Code Obfuscation

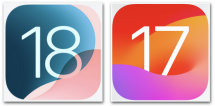
iOS: Grundlagen & Allgemeines

Vorbemerkungen

- Apples Software im Allgemeinen „Closed Source“
- ⇒ bzgl. Informationen *abhängig* von …
  - Apple
  - Researcher:innen, die „Reverse Engineering“ betreiben
- ⇒ Vorlesungseinheit basiert stark auf Apples Dokumentation & Papers von Researcher:innen (Details im Literaturverzeichnis)
- einige der vorgestellten Konzepte auch von anderen Betriebssystemen/Plattformen verwendet
- in dieser Vorlesung angeführte Informationen gelten auch für iPadOS (sofern nicht explizit Gegenteiliges angemerkt)

Update-Zyklus & Architektur von iOS

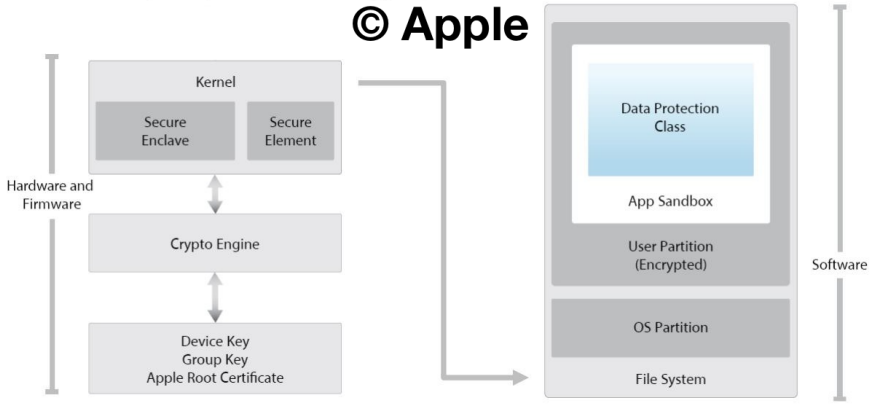
- jährlicher Versionssprung („große“ Features, Designänderungen)
- Point-Releases im Verlauf des Jahres (Security, kleinere Features)
- **Support:** 5-6 Jahre (iPhone 6S, XS, XR bspw. sogar 7),
  seit Juni 2024: Garantie für 5 Jahre,
  ältere Versionen bekommen u.U. ebenfalls noch Sicherheitsupdates
- großteils „Closed Source“, nur auf Apple-Geräten unterstützt
- basiert auf macOS (bzw. OS X) – siehe vorangegangene Einheit
- aktuelle Version (Herbst 2024): iOS 18



© Apple

iOS: Integritätssicherung

iOS Sicherheitsarchitektur

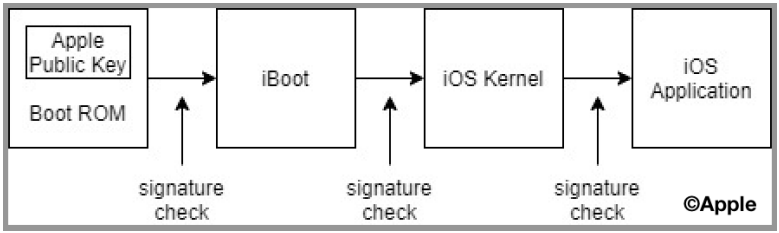


Secure Boot: Überblick

- jeder Schritt enthält **Integritätscheck**
  (kryptographisch signiert von Apple)

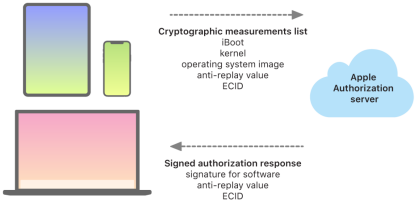
- „Chain of Trust“ (Kette des Vertrauens)

- **Ziel:** Lowest Level Software soll nicht modifiziert werden



Sichere Software Updates

- Signatur-Prozess verhindert, dass ältere Versionen installiert werden (Apple signiert alte OS-Versionen nicht mehr)
- Update über Finder/iTunes: volle OS-Kopie wird geladen, over-the-air Update: nur benötigte Komponenten werden geladen
- Data Volume ist während Updates unmounted
- Updates werden durch Exclusive Chip ID (ECID) personalisiert



How Apple devices interact with the Apple Authorization server.

© Apple

iOS: Apps (Entwicklung, Verbreitung, Aufbau)

iOS Apps – Entwicklung

- **iOS Software Development Kit (SDK)**: enthält Ressourcen, Technologien & Werkzeuge für iOS-App-Entwicklung – kostenlos
- iOS Apps können *nur auf macOS* entwickelt werden
- **Xcode:** kostenloser Projekt- & Schnittstellen-Builder
- seit 2015 empfiehlt Apple Verwendung von **Swift**, davor **Objective-C**

iOS Apps – Verbreitung

- **Apple App Store**: digitale Plattform für Verbreitung von iOS Apps
- App Store stellt Anforderungen an Apps (z.B. Sicherheit, Datenschutz) ⇒ Prüfung vor Veröffentlichung
- **Apple Developer Program**: Entwickler:innen müssen sich registrieren, um iOS Apps im App Store anbieten zu dürfen
- Apps müssen durch Entwickler:innen signiert sein, damit im App Store veröffentlicht werden können ⇒ *Zertifikatvalidierung* (über Apple Developer Program)
- Apple Developer *Enterprise* Program (für Unternehmen) ⇒ **Provisioning Profile**: ermöglicht Ausführung interner Apps auf Geräten von Mitarbeitenden

iOS Apps – Codesignierung

- stellt sicher, dass alle Apps von bekannten & genehmigten Quellen stammen & nicht manipuliert wurden
- gesamter ausführbarer Code muss mit von Apple ausgegebenem Zertifikat signiert sein
  - ab Werk vorhandene Apps (z.B. Mail, Safari)
    $\Rightarrow$ von Apple signiert
  - Apps von Dritten
    $\Rightarrow$ von Entwickler:in signiert
- „Chain of Trust“-Konzept auf Apps $\Rightarrow$ verhindert, dass Apps Dritter nicht signierten Code ausführen

iOS Apps – Sandboxing

- jede App Dritter läuft in eigener Sandbox
- Apps können keine von anderen Apps gespeicherten Informationen abrufen oder verändern
- ermöglicht Trennung von anderen Apps & OS
- Systemdateien & Ressourcen werden von den User:innen-Apps abgeschirmt
- Apps Dritter sowie meiste Systemdateien & Ressourcen von iOS als nicht privilegierte:r User:in „mobile“ ausgeführt
- gesamte Betriebssystempartition $\Rightarrow$ nur Lesezugriff

iOS App Store Package (IPA)

- **iOS App Store Package (*ipa-Datei)**: ZIP-Archiv, enthält Ressourcen, kompilierten Anwendungscode & Metadaten
- wird mit Entwickler:innenzertifikat signiert & im App Store veröffentlicht
- enthält Ordner „Payload“, der Anwendungsdaten enthält
- **„Application.app“-Datei**: enthält App-Binärdatei, Bundle-Ressourcen, `Embedded.mobileprovision`, `_CodeSignature`
- `Embedded.mobileprovision` ermöglicht Entwickler:innen, iOS-Anwendung ohne Xcode neu zu signieren
- `_CodeSignature` ermöglicht Überprüfung der Integrität
- ausführbare Datei im Bundle ist im „*Mach-O-Objektdateiformat*“

Bundle-ID

- = eindeutige Kennung für eine App
  - *Beispiel:* App Store: `com.apple.AppStore`
  - werden benötigt, um z.B. Capabilities zuzuweisen

```
                                        % frida-ps -Uai
 PID  Name           Identifier
----  -------------  ----------------------------------------------
5572  Ada            net.medx.Ada.production
5616  Einstellungen  com.apple.Preferences
5629  Filza          com.tigisoftware.Filza
5263  Kamera         com.apple.camera
5815  WhatsApp       net.whatsapp.WhatsApp
   -  Aktien         com.apple.stocks
   -  App Store      com.apple.AppStore
   -  Bücher         com.apple.iBooks
   -  Cydia          com.saurik.Cydia
   -  Dateien        com.apple.DocumentsApp
   -  Erinnerungen   com.apple.reminders
   -  FaceTime       com.apple.facetime
   -  Fotos          com.apple.mobileslideshow
   -  Health         com.apple.Health
   -  Home           com.apple.Home
   -  Kalender       com.apple.mobilecal
   -  Karten         com.apple.Maps
   -  Kompass        com.apple.compass
   -  Kontakte       com.apple.MobileAddressBook
   -  Kurzbefehle    com.apple.shortcuts
   -  Mail           com.apple.mobilemail
   -  Maßband        com.apple.measure
   -  Musik          com.apple.Music
   -  Nachrichten    com.apple.MobileSMS
   -  Notizen        com.apple.mobilenotes
```

iOS App-Verzeichnisse: „Bundle“, „Shared“

Bundle:

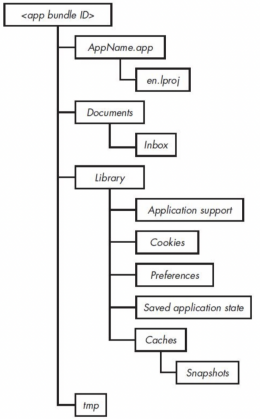
- enthält Verzeichnis für jede auf Gerät installierte App
- *Pfad*: `/private/var/containers/Bundle/Application/<UUID>`
- jedes App-Verzeichnis enthält Ordner `<App-Name>.app`, in dem Ressourcen, `Info.plist` & Binär-Dateien enthalten

Shared:

- für App-Gruppen: Apps eines Entwickler:innenaccounts können Inhalte, Schlüsselbundobjekte, Einstellungen gemeinsam verwenden
- *Pfad*: `/var/mobile/Containers/Shared/AppGroup/<App_Group_Identifier>`
- enthält Anwendungen zur gemeinsamen Nutzung von Daten

iOS App-Verzeichnisse: „Data“

- speichert App-Laufzeitdaten
- *Pfad*: `/var/mobile/Containers/Data/Application/<UUID>`
- enthält für jede installierte App Ordner mit Bundle-ID als Namen
- enthält Daten wie Einstellungen, Caches & Cookies



(Vergleiche David Thiel. IOS Application Security: The Definitive Guide for Hackers and Developers. San Francisco: No Starch Press, 2016.)

Eigenschaftslisten-Dateien (.plist)

- **Property List (plist)-Dateien** dienen zur Speicherung von Anwendungskonfigurationsdaten
- können binäres oder XML-Format haben
- = Wörterbuch, das hierarchische Schlüssel-Wert-Paare speichert
- können im Klartext abgerufen werden ⇒ sensible Informationen (z.B. Anmeldedaten) sollten nicht darin gespeichert werden

Info.plist

- muss immer im IPA vorhanden sein
- im Stammverzeichnis der App gespeichert
- beinhaltet Informationen wie z.B. Bundle-ID oder unterstützte iOS-Versionen
- beinhaltet Nutzungsbeschreibungstexte („purpose string“) für geschützte Ressourcen (z.B. Fotos, Standort)
  - in Eingabeaufforderung verwendet, wenn Erlaubnis erfragt
  - kein String vorhanden ⇒ Versuch schlägt fehl, App kann abstürzen
  - mehr als 20 geschützte Ressourcen, z.B. Kamera, Mikrofon, Dateien, Ordner, Netzwerk, NFC, Standort

Info.plist: Beispiel (XML)

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
  <key>NSLocationWhenInUseUsageDescription</key>
  <string>Your location is used to help you find health services near you.</string>
  <key>CFBundleDevelopmentRegion</key>
  <string>en</string>
  <key>CFBundleURLTypes</key>
  <array>
    <dict>
      <key>CFBundleURLSchemes</key>
      <array>
        <string>adahealth</string>
        <string>ada</string>
      </array>
      <key>CFBundleURLName</key>
      <string>com.ada.app</string>
    </dict>
  </array>

  <!-- [...] -->
</dict>
</plist>
```

Keychain (Schlüsselbund)

- kann verwendet werden, um sensible Daten (z.B. Passwörter, Anmeldedaten, Zertifikate) zu speichern
- als **SQLite-Datenbank** implementiert, im Dateisystem gespeichert
- Schlüsselbundobjekte werden verschlüsselt
- auf Datenbank kann nur über die API zugegriffen werden
- ein Schlüsselbund für alle Apps unter iOS – außerhalb der App-Sandbox gespeichert

iOS: Datenspeicher

Lokale Datenspeicher: SQLite

- Datei-basierte Datenbank (`.db`, `.sqlite`)
- SQLite-Engine benötigt keinen Server
- kann in Swift mit Hilfe eines Wrappers verwendet werden
- „Light-Version“ komplexer relationaler Datenbankmanagementsysteme wie MySQL

  $\Rightarrow$ weniger leistungsfähig
- verwendet „write-ahead log“ (WAL), um Änderungen in Datenbank zu erfassen

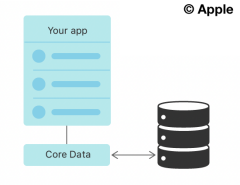
Lokale Datenspeicher: Core Data

- = Framework, häufig verwendet zur lokalen Speicherung von Daten
- kann Daten für Offline-Nutzung (permanent) speichern, temporär zwischenspeichern & Undo-Funktionalität in App bereitstellen
- Daten als Entity-Attribut-Modell dargestellt ⇒ z.B. in SQLite & XML serialisierbar
- verwaltet Objekt-Instanzen zur Laufzeit
- High-Level-Abstraktionsdarstellung ermöglicht direkte Kommunikation mit SQLite-Datenbanken
- schneller und einfacher Ansatz, um auch große Daten zu speichern, ohne direkt eine Datenbank zu verwalten

Core Data auslesen

Üblicherweise zu finden in:

- `Library/Application Support/[name].sqlite`
- `Documents/[name].sqlite`

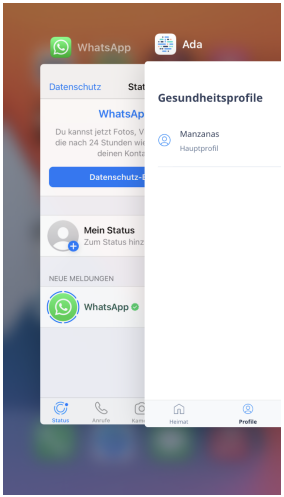


(Vergleiche `https://developer.apple.com/documentation/coredata`, abgerufen: 28. November 2024)

Lokaler Datenspeicher: Caches

- On-disk Cache für NSUrlRequest[1]:
  - kann Passwörter, sensitive Informationen (z.B. Kreditkarte) etc. beinhalten
- automatisch generierte Screenshots durch iOS:
  - wenn App in den Hintergrund geht, macht iOS Screenshot vom App-Fenster
  - kann sensible Informationen zeigen
  - können von Entwickler:in aber deaktiviert werden

Automatisch generierte Screenshots: Beispiel



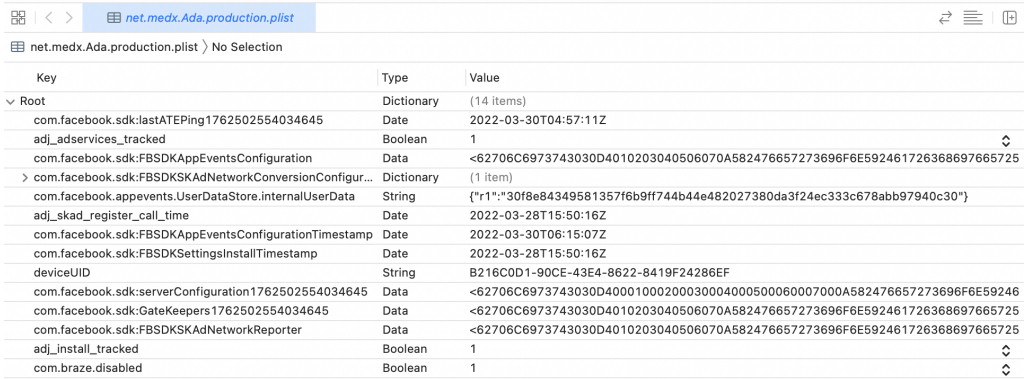
Lokaler Datenspeicher: URL Cache & Snapshots auslesen

- `Library/Caches/[Bundle ID]/Cache.db`
- größere Dateien unter `Library/Caches/[Bundle ID]/fsCachedData`
  - *receiver_data* Spalte in `Cache.db` $\Rightarrow$ *isDataOnFS = 1*
  - wenn *isDataOnFS = 0*, dann Daten in Zeile in `Cache.db`
- **Snapshots** unter `Library/SplashBoard/Snapshots/`

Lokaler Datenspeicher: User Defaults

- speichert Daten als Schlüssel-Wert-Paare
- Standardobjekt muss Eigenschaftslisten-Typ sein
- Lese- & Schreibvorgänge haben Auswirkungen auf Anwendungsleistung
  $\Rightarrow$ sollte nicht zum Speichern großer Daten verwendet werden
- in binärer Eigenschaftslisten-Datei im Einstellungsordner
  (`/Library/Preferences/…plist`) gespeichert
- nicht verschlüsselt, nicht durch Data Protection geschützt
  $\Rightarrow$ nicht empfehlenswert für sensiblen Daten & Anmeldedaten
- kann für Speichern von Einstellungsdaten verwendet werden

User Defaults: Beispiel (geöffnet in Xcode)



Key	Type	Value
Root	Dictionary	(14 items)
com.facebook.sdk:lastATEPing1762502554034645	Date	2022-03-30T04:57:11Z
adj_adservices_tracked	Boolean	1
com.facebook.sdk:FBSDKAppEventsConfiguration	Data	<62706C6973743030D4010203040506070A5824766572736F6E592461726368697665725
com.facebook.sdk:FBSDKSKAdNetworkConversionConfigur...	Dictionary	(1 item)
com.facebook.appevents.UserDataStore.internalUserData	String	{"r1":"30f8e84349581357f6b9ff744b44e482027380da3f24ec333c678abb97940c30"}
adj_skad_register_call_time	Date	2022-03-28T15:50:16Z
com.facebook.sdk:FBSDKAppEventsConfigurationTimestamp	Date	2022-03-30T06:15:07Z
com.facebook.sdk:FBSDKSettingsInstallTimestamp	Date	2022-03-28T15:50:16Z
deviceUID	String	B216C0D1-90CE-43E4-8622-8419F24286EF
com.facebook.sdk:serverConfiguration1762502554034645	Data	<62706C6973743030D4000100020003000400050006007000A582476657273696F6E59246
com.facebook.sdk:GateKeepers1762502554034645	Data	<62706C6973743030D4010203040506070A5824766572736F6E592461726368697665725
com.facebook.sdk:FBSDKSKAdNetworkReporter	Data	<62706C6973743030D4010203040506070A5824766572736F6E592461726368697665725
adj_install_tracked	Boolean	1
com.braze.disabled	Boolean	1

Ausblick: Weitere Themen ...

Es gibt noch viele weitere Themen, die heute keinen Platz gefunden haben ...

- Apple Pay
- Verbundene Devices
- Apple Security Research Devices
- Developer (Kits) Security
- Device Management (innerhalb von Organisationen)
- ...

⇒ LVAs (Mobile Security), Bachelor-, Diplom- & Projektarbeiten bei der ESSE! :)

Zusammenfassung

- Android basiert auf Linux, iOS auf macOS (bzw. OS X)
  – open vs. closed
- beide OS stellen Sandboxing und Berechtigungs-Mechanismen zur Verfügung
- APKs enthalten vollständige Android Applikationen, iOS Apps im ipa-Format
- Android: üblicherweise Java/Kotlin, iOS: Swift (früher: Objective-C)
- Byte Code vs. Machine Code
- zwei (relativ) restriktive App-Stores, wobei mittlerweile auch bei iOS Sideloading nativ unterstützt wird

Literaturverzeichnis 1/11

- Android Runtime (ART) and Dalvik: `https://source.android.com/devices/tech/dalvik`, abgerufen: 28. November 2024
- App Manifest Overview: `https://developer.android.com/guide/topics/manifest/manifest-intro`, abgerufen: 28. November 2024
- Dalvik Executable format: `https://source.android.com/devices/tech/dalvik/dex-format`, abgerufen: 28. November 2024
- Content provider basics: `https://developer.android.com/guide/topics/providers/content-provider-basics`, abgerufen: 28. November 2024
- Broadcasts overview: `https://developer.android.com/guide/components/broadcasts`, abgerufen: 28. November 2024

Literaturverzeichnis 2/11

- Elenkov (2014): Android Security Internals
- Drake, et al. (2014): Android Hacker's Handbook
- Chell (2015): The Mobile Application Hacker's Handbook

Literaturverzeichnis 3/11

- Apple Keynotes: https://podcasts.apple.com/us/podcast/apple-events-video/id275834665/, abgerufen: 28. November 2024
- Secure Enclave: https://support.apple.com/de-at/guide/security/sec59b0b31ff/web, abgerufen: 28. November 2024
- Find My: https://support.apple.com/en-us/HT204756, abgerufen: 28. November 2024
- Private Relay: https://developer.apple.com/videos/play/wwdc2021/10096, abgerufen: 28. November 2024
- Safari Privacy: https://www.apple.com/privacy/, abgerufen: 28. November 2024

Literaturverzeichnis 4/11

- Systemsicherheit: `https://support.apple.com/de-at/guide/security/sec114e4db04/1/web/1`, abgerufen: 28. November 2024
- Sicherheit bei Apps: `https://support.apple.com/de-at/guide/security/sec35dd877d0/1/web/1`, abgerufen: 28. November 2024
- Hardwaresicherheit – Überblick: `https://support.apple.com/de-at/guide/security/secf020d1074/1/web/1`, abgerufen: 28. November 2024
- Verschlüsselung & Datensicherheit – Überblick: `https://support.apple.com/de-at/guide/security/sece3bee0835/1/web/1`, abgerufen: 28. November 2024
- iCloud Sicherheit: `https://support.apple.com/en-us/HT202303`, abgerufen: 28. November 2024

Literaturverzeichnis 5/11

- Apple Dokumentation & Support: https://support.apple.com/, abgerufen: 28. November 2024
- Apple Platform Security (May 2021): https://manuals.info.apple.com/MANUALS/1000/MA1902/en_US/apple-platform-security-guide.pdf, abgerufen: 28. November 2024
- Korak, T., & Hoefler, M. (2014, September). On the effects of clock and power supply tampering on two microcontroller platforms. In 2014 Workshop on Fault Diagnosis and Tolerance in Cryptography (pp. 8-17). IEEE.: https://online.tugraz.at/tug_online/voe_main2.getvolltext?pCurrPk=79126, abgerufen: 28. November 2024

Literaturverzeichnis 6/11

- Data Protection: https://support.apple.com/de-at/guide/security/secf6276da8a/web, abgerufen: 28. November 2024
- Sealed Key Protection (SKP): https://support.apple.com/de-at/guide/security/secf5549a4f5/web, abgerufen: 28. November 2024
- Keychain: https://support.apple.com/de-at/guide/security/secb0694df1a/web, abgerufen: 28. November 2024
- Erweiterungen: https://support.apple.com/de-de/guide/security/secabd3504cd/web, abgerufen: 28. November 2024
- App-Gruppen: https://support.apple.com/de-de/guide/security/sec1a976c067/web, abgerufen: 28. November 2024

Literaturverzeichnis 7/11

- Sandboxing: https://support.apple.com/de-de/guide/security/sec15bfe098e/web, abgerufen: 28. November 2024
- Übersicht: https://support.apple.com/de-de/guide/security/secf49cad4db/web, abgerufen: 28. November 2024
- Datensicherheit bei Apple-Geräten: https://support.apple.com/de-at/guide/security/sece8608431d/1/web/1, abgerufen: 28. November 2024
- Datensicherheitsklassen: https://support.apple.com/de-at/guide/security/secb010e978a/web, abgerufen: 28. November 2024

Literaturverzeichnis 8/11

- Apple. Protected Resources. 2021. https://developer.apple.com/documentation/bundleresources/information_property_list/protected_resources, abgerufen: 28. November 2024
- Apple. Core Data. 2021. https://developer.apple.com/documentation/coredata, abgerufen: 28. November 2024
- Apple. NSUserDefaults. 2021. https://developer.apple.com/documentation/foundation/nsuserdefaults, abgerufen: 28. November 2024

Literaturverzeichnis 9/11

- Ahd Radwan and Samer Zein. „Model-Based Approach for Supporting Quick Caching at iOS Platform“. In: International Journal of Advanced Trends in Computer Science and Engineering 9 (Sept. 2020), pp. 4285–4294. doi: 10.30534/ijatcse/2020/17942020.
- Madalina Angelica Marin et al. „Proactive Secure Coding for iOS Applications“. In: 2019 18th RoEduNet Conference: Networking in Education and Research (RoEduNet). Galati, Romania: IEEE, 2019, pp. 1–5. doi: 10.1109/ROEDUNET.2019.8909672.
- Vijay Kumar Velu. Mobile Application Penetration Testing. Birmingham: Packt Publishing Ltd, 2016. isbn: 978-1-78588-337-8.
- David Thiel. IOS Application Security: The Definitive Guide for Hackers and Developers. San Francisco: No Starch Press, 2016.

Literaturverzeichnis 10/11

- Apple. Item Attribute Keys and Values. 2021. https://developer.apple.com/documentation/security/keychain_services/keychain_items/item_attribute_keys_and_values, abgerufen: 28. November 2024
- Apple. Keychain data protection. Feb. 18, 2021. https://support.apple.com/guide/security/keychain-data-protection-secb0694df1a/1/web/1, abgerufen: 28. November 2024
- Apple. Keychain Services. 2021. https://developer.apple.com/documentation/security/keychain_services, abgerufen: 28. November 2024
- OWASP. Data Storage on iOS. 2019. https://mobile-security.gitbook.io/mobile-security-testing-guide/ios-testing-guide/0x06d-testing-data-storage, abgerufen: 28. November 2024

Literaturverzeichnis 11/11

- Apple. Bundle IDs. 2021.
  `https://developer.apple.com/documentation/appstoreconnectapi/bundle_ids`, abgerufen: 28. November 2024
- Apple. Protecting the User's Privacy. 2021.
  `https://developer.apple.com/documentation/uikit/protecting_the_user_s_privacy`, abgerufen: 28. November 2024
- Konrad Kollnig, Anastasia Shuba, Max Van Kleek, Reuben Binns, und Nigel Shadbolt. Goodbye Tracking? Impact of IOS App Tracking Transparency and Privacy Labels.
  In *2022 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '22, Seite 508–520, New York, NY, USA, 2022. Association for Computing Machinery.
  ISBN 9781450393522.
  doi: 10.1145/3531146.3533116

Vielen Dank!

https://establishing-security.at/